**APLIKASI GERBANG LOGIKA BERBASIS ANDROID SEBAGAI SIMULASI IMPLEMENTASI MATA KULIAH LOGIKA INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS MUHAMMADIYAH JAKARTA (FT-UMJ)**

**SKRIPSI**

**Sebagai Syarat Dalam Menyelesaikan Program Studi Strata Satu (S-1)**

**Program Studi Teknik Informatika**

****

**Disusun Oleh:**

**NAMA : YEKTO PRIYANDHANI**

**NIM : 2016470114**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MUHAMMADIYAH JAKARTA**

**2021**

**UNIVERSITAS MUHAMMADIYAH JAKARTA FAKULTAS**

**TEKNIK-PRODI TEKNIK INFORMATIKA**

# LEMBAR PERSETUJUAN

**APLIKASI GERBANG LOGIKA BERBASIS ANDROID SEBAGAI SIMULASI IMPLEMENTASI MATA KULIAH LOGIKA INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS MUHAMMADIYAH JAKARTA (FT-UMJ)**

NAMA : YEKTO PRIYANDHANI

NIM : 2016470114

PROGRAM STUDI : TEKNIK INFORMATIKA

Skripsi ini telah disetujui pada tanggal 28 Juli 2021

Oleh

Pembimbing Utama

Popy Meilina S.T., M.KOM

**UNIVERSITAS MUHAMMADIYAH JAKARTA FAKULTAS**

**TEKNIK-PRODI TEKNIK INFORMATIKA**

# TANDA BUKTI PERSETUJUAN PEMBIMBING SKRIPSI

Pada Semester Genap

Tahun Akademik 2020/2021

Yang bertanda tangan dibawah ini Pembimbing Skripsi menyetujui bahwa:

Nama : Yekto Priyandhani

NIM : 2016470114

Judul TA

Aplikasi Gerbang Logika Berbasis Android Sebagai Simulasi Implementasi Mata Kuliah Logika Informatika Fakultas Teknik Universitas Muhammadiyah Jakarta (FT-UMJ)

Dimulai bulan, tahun : Maret, 2021

Selesai bulan, tahun : Agustus, 2021

Untuk ikut serta Ujian Sidang Strata Satu (S1) yang diselenggarakan oleh Program Studi Teknik Informatika Universitas Muhammadiyah Jakarta.

Jakarta, 28 Juli 2021

Pembimbing Utama

Popy Meilina S.T., M.KOM

**UNIVERSITAS MUHAMMADIYAH JAKARTA FAKULTAS**

**TEKNIK-PRODI TEKNIK INFORMATIKA**

# LEMBAR PENGESAHAN

**APLIKASI SIMULASI GERBANG LOGIKA BERBASIS ANDROID SEBAGAI SIMULASI IMPLEMENTASI MATA KULIAH LOGIKA INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS MUHAMMADIYAH JAKARTA (FT-UMJ)**

NAMA : YEKTO PRIYANDHANI

NIM : 2016470114

PROGRAM STUDI : TEKNIK INFORMATIKA

Skripsi ini telah diuji pada tanggal 4 Agustus 2021

**Oleh Penguji**

1. Nurvelly Rosanti M.Kom : ………………………………….

2. Yana Adharani, M.Kom : ………………………………….

3. Popy Meilina, M.Kom : ………………………………….

**UNIVERSITAS MUHAMMADIYAH JAKARTA FAKULTAS**

**TEKNIK-PRODI TEKNIK INFORMATIKA**

# LEMBAR PERNYATAAN

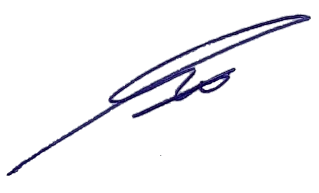
Bersama ini saya menyatakan bahwa isi yang terkandung dalam Skripsi ini, dengan Judul :

APLIKASI SIMULASI GERBANG LOGIKA BERBASIS ANDROID SEBAGAI SIMULASI IMPLEMENTASI MATA KULIAH LOGIKA INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS MUHAMMADIYAH JAKARTA (FT-UMJ)

Adalah murni merupakan hasil penelitian dan pemikiran saya sendiri.

Demikian pernyataan ini saya buat dan siap menerima konsekuensi apapun di masa yang akan datang apabila ternyata Skripsi ini merupakan salinan ataupun contoh karya-karya yang telah dibuat / diterbitkan sebelum tanggal Skripsi ini.

Jakarta

Penulis

(Yekto Priyandhani)

**UNIVERSITAS MUHAMMADIYAH JAKARTA FAKULTAS**

**TEKNIK-PRODI TEKNIK INFORMATIKA**

# SURAT PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Yang bertanda tangan dibawah ini, saya:

Nama : Yekto Priyandhani

NIM : 2016470114

Program Studi : Teknik Informatika

Jenjang : Strata Satu (S1)

Jenis Karya : Skripsi

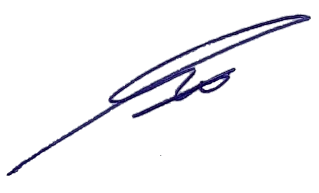
Demi pengembangan ilmu pengetahuan, dengan ini menyetujui untuk memberikan ijin kepada pihak Program Studi Teknik Informatika FT-UMJ **Hak Bebas Royalti Non-Eksklusif (Non-exclusive Royalti-Free Right)** atas karya ilmiah saya yang berjudul : “**Aplikasi Gerbang Logika Berbasi Android Sebagai Simulasi Implementasi Mata Kuliah Logika Informatika Fakultas Teknik Universitas Muhammadiyah Jakarta (FT-UMJ)**” beserta perangkat yang diperlukan (apabila ada).

Dengan **Hak Bebas Royalti Non-Eksklusif** ini pihak FT-UMJ berhak menyimpan, mengalih-media atau bentuk-kan, mengelolanya dalam pangkalan data (database), mendistribusikannya dan menampilkan atau mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari Saya selama tetap mencantumkan nama kami sebagai penulis/pencipta karya ilmiah tersebut.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak FT- UMJ, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini.

Demikian pernyataan ini saya buat dengan sebenarnya.

Jakarta,

 Yang menyatakan,

Yekto Priyandhani

**UNIVERSITAS MUHAMMADIYAH JAKARTA FAKULTAS**

**TEKNIK-PRODI TEKNIK INFORMATIKA**

# DAFTAR PRESENSI BIMBINGAN SKRIPSI

**APLIKASI SIMULASI GERBANG LOGIKA BERBASIS ANDROID SEBAGAI SIMULASI IMPLEMENTASI MATA KULIAH LOGIKA INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS MUHAMMADIYAH JAKARTA (FT-UMJ)**

Nama : Yekto Priyandhani

NPM : 2016470114

Program Studi : Teknik Informatika

Dosen Pendamping : Popy Meilina, M.KOM

|  |  |  |  |
| --- | --- | --- | --- |
| No | Tanggal | Catatan Dosen Pembimbing | Paraf |
| 1 | 18 Maret 2021 | Pengajuan BAB 1   * Revisi identifikasi masalah * Revisi rumusan masalah * Revisi Batasan masalah * Revisi tujuan masalah |  |
| 2 | 29 Maret 2021 | Pengajuan BAB 1 yang sudah direvisi  Pengajuan BAB 2   * Diawali dengan Logika Komputer * Menghilangkan Sub-bab Aplikasi * Revisi Kutipan * Penggunaan Header Tabel |  |
| 3 | 20 April 2021 | Pengajuan BAB 2 yang sudah direvisi   * Revisi Kutipan   Pengajuan BAB 3   * Revisi Activity diagram system berjalan * Revisi Prosedur Berjalan |  |
| 4 | 26 April 2021 | Pengajuan BAB 3   * Revisi Use Case Diagram, Activity Diagram Sequence Diagram karena penggunaan include pada Use Case tidak benar. * Soal ditampilkan acak |  |
| 5 | 27 April 2021 | Pengajuan BAB 3 yang sudah direvisi |  |
| 6 | 31 Mei 2021 | Pembahasan arahan revisi Seminar Skripsi |  |
| 7 | 3 Juli 2021 | Pengajuan hasil Revisi berdasarkan arahan dari dosen penguji.   * Level soal random : mudah, normal, sulit. * Soal kategorikal berdasarkan operator. * Bank soal mudah 18, mudah dan sulit minimal 30. * Kategorikal menampilkan 10 soal dengan distirbusi 2 soal mudah, 4 soal normal, dan 4 soal sulit. * Penentuan shuffle soal * Perincian output * Letak kelistirkan * Demo aplikasi |  |
| 8 | 19 Juli 2021 | * Perbaikan bug aplikasi. * Pengajuan BAB 4 dan BAB 5 * Revisi BAB 4 Pembahasan Hasil dan BAB 5 Kesimpulan * Melengkapi judul gambar dan table. |  |
| 9 | 27 Juli 2021 | Pengajuan hasil revisi BAB 4 dan BAB 5   * Melengkapi kesimpulan pada BAB 5 |  |
| 10 | 28 Juli 2021 | Pengajuan Draft Skripsi Final |  |

Dosen Pembimbing

(Popy Meilina, M.Kom)

# ABSTRACT

*Informatics Logic is one of the courses in Informatics Engineering at the Faculty of Engineering, University of Muhammadiyah Jakarta (FT-UMJ). During the learning process students are expected to be able to understand the basic concepts of logic to the operation of logic, especially logic on computers. Logic on the computer has a truth value where if it is true then the value is 1 (TRUE) and if it is false then the value is 0 (FALSE). To understand the questions and answer questions with more than two inputs and more than one operator, students have to make a lot of truth tables and more reasoning is needed. Therefore, the researchers tried to implement the learning outcomes by making an Android-based Logic Gate Simulation Application in the form of a game. With this application, users can understand and answer the question directly and attract interest because it is presented in the form of a game. Application provides operator usage information (And, Or, Nand, Nor, Xor, and Xnor). The application also provides question selection based on the number of inputs or the type of operator used in the problem. To answer the question, the user only needs to choose one of the three answer options provided. If the answer is correct then the light is on and if it is wrong the light will be off. For each correct answer will be given a value of ten and zero for wrong answers. After the game ends the application will display the score.*

*Keywords: Informatic Logic, Logic Gate, game.*

# ABSTRAK

Logika Informatika adalah salah satu mata kuliah Teknik Informatika di Fakultas Teknik Universitas Muhammadiyah Jakarta (FT-UMJ). Selama proses pembelajaran mahasiswa diharapkan mampu memahami konsep dasar tentang logika sampai pengoperasian logika terutama logika pada komputer. Logika pada computer memiliki nilai kebenaran dimana jika benar maka nilainya 1 (TRUE) dan jika salah maka nilainya 0 (FALSE). Untuk memahami soal serta menjawab soal dengan *input* lebih dari dua dan *operator* lebih dari satu mahasiswa harus membuat table kebenaran yang banyak dan dibutuhkan penalaran yang lebih. Oleh karena itu peneliti mencoba mengimplementasikan hasil pembelajaran dengan membuat sebuah Aplikasi Simulasi Gerbang Logika berbasis android dalam bentuk *game*. Dengan adanya aplikasi ini pengguna dapat memahami serta menjawab soal tersebut secara langsung dan menarik minat karena disajikan dalam bentuk *game*. Aplikasi menyediakan informasi penggunaan operator (And, Or, Nand, Nor, Xor, dan Xnor). Aplikasi juga menyediakan pemilihan soal berdasarkan jumlah *input* atau jenis operator yang digunakan pada soal. Untuk menjawab soal, pengguna hanya tinggal memilih satu dari tiga pilihan jawaban yang disediakan. Jika jawaban benar maka lampu hidup dan jika salah lampu akan mati. Untuk setiap jawaban benar akan diberi nilai sepuluh dan nol untuk jawaban salah. Setelah permainan berakhir aplikasi akan menampilkan perolehan nilai.

Kata Kunci: Logika Informatika, Gerbang Logika, *game*.

# KATA PENGANTAR

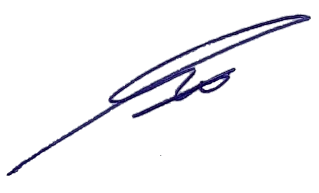
Alhamdulillah serta puji syukur kehadirat Allah S.W.T. yang telah memberikan rahmat serta karunia sehat yang tidak terbatas sehingga penulis dapat menyelesaikan skripsi dengan judul “Aplikasi Gerbang Logika Berbasis Android Sebagai Simulasi Implementasi Mata Kuliah Logika Informatika Fakultas Teknik Universitas Muhammadiyah Jakarta (FT-UMJ)”.

Proposal ini disusun memenuhi sebagaian dar syarat-syarat guna mencapai gelar Sarjana Sistem Informatika di Universitas Muhammadiyah Jakarta. Penulisan Skripsi ini tidak akan terwujud tanpa adanya dukungan berupa pengarahan, bimbingan, dan Kerjasama semua pihak yang turut membantu dalam proses penyelesaian skripsi ini, Untuk itu, penulis ingin menyampaikan ucapan terima kasih kepada:

1. Bapak Irfan Purnawan, ST., M.CHEM.ENG. selaku Dekan Fakultas Teknik Universitas Muhammadiyah Jakarta.
2. Ibu Popy Meilina, S.T., M. Kom selaku dosen pembimbing dan Ketua Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Jakarta.
3. Ayah, ibu, serta adik-adiku yang selalu memberikan *support*.
4. Saudara Fajar Ramadhan selaku rekan tim *developer*.
5. Teman-teman seperjuangan yaitu Asep, Jadi, Munir, Yulia, Lina, Benne, Imam, Adit, dan Yulian.

Penulis menyadari bahwa skripsi ini masih dapat kekurangan dan keterbatasan serta ilmu pengetahuan yang dimiliki oleh penulis. Oleh karena itu, kritik dan saran dibutuhkan untuk menambah wawasan dan berpikir bagi pembaca.

Jakarta 28 Juli 2021



Yekto Priyandhani

# DAFTAR ISI

[LEMBAR PERSETUJUAN i](#_Toc80857285)

[TANDA BUKTI PERSETUJUAN PEMBIMBING SKRIPSI ii](#_Toc80857286)

[LEMBAR PENGESAHAN iii](#_Toc80857287)

[LEMBAR PERNYATAAN iv](#_Toc80857288)

[SURAT PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS v](#_Toc80857289)

[DAFTAR PRESENSI BIMBINGAN SKRIPSI vi](#_Toc80857290)

[ABSTRACT ix](#_Toc80857291)

[ABSTRAK x](#_Toc80857292)

[KATA PENGANTAR xi](#_Toc80857293)

[DAFTAR ISI xii](#_Toc80857294)

[DAFTAR TABEL xv](#_Toc80857295)

[DAFTAR GAMBAR xvi](#_Toc80857296)

[DAFTAR LAMPIRAN xviii](#_Toc80857297)

[BAB 1 1](#_Toc80857298)

[1.1 Latar Belakang Masalah 1](#_Toc80857299)

[1.2 Identifikasi Masalah 2](#_Toc80857300)

[1.3 Rumusan Masalah 2](#_Toc80857301)

[1.4 Batasan Masalah 3](#_Toc80857302)

[1.5 Tujuan Penelitian 3](#_Toc80857303)

[1.6 Metodologi Penelitian 3](#_Toc80857304)

[BAB II 5](#_Toc80857305)

[2.1 Sejarah dan Pengertian Logika Informatika 5](#_Toc80857306)

[2.2 Logika dan Komputer 6](#_Toc80857307)

[2.3 Waterfall Modell 8](#_Toc80857308)

[2.4 Unified Modeling Language (UML) 10](#_Toc80857309)

[2.4.1 Use Case Diagram 11](#_Toc80857310)

[2.4.2 Activity Diagram 14](#_Toc80857311)

[2.4.3 Sequence Diagram 15](#_Toc80857312)

[2.5 Flowchart 17](#_Toc80857313)

[2.6 Hierarchy Plus Input Output (HIPO) 18](#_Toc80857314)

[2.7 Pengujian Perangkat Lunak 19](#_Toc80857315)

[2.7.1 Black Box Testing 20](#_Toc80857316)

[BAB III 21](#_Toc80857317)

[3.1 Tinjauan Organisasi 21](#_Toc80857318)

[3.1.1 Sejarah Organisasi 21](#_Toc80857319)

[3.1.2 Visi dan Misi 22](#_Toc80857320)

[3.1.3 Struktur Organisasi 23](#_Toc80857321)

[3.2 Analisis Sistem Berjalan 23](#_Toc80857322)

[3.2.1 Prosedur Sistem Berjalan 23](#_Toc80857323)

[3.2.2 Activity Diagram Sistem Berjalan 24](#_Toc80857324)

[3.2.3 Permasalahan Sistem Berjalan 25](#_Toc80857325)

[3.3 Sistem Usulan 25](#_Toc80857326)

[3.3.1 Prosedur Sistem Usulan 25](#_Toc80857327)

[3.3.2 Metode Pengembangan Perangkat Lunak 27](#_Toc80857328)

[3.3.3 Perancangan UML Sistem Usulan 29](#_Toc80857329)

[3.3.3.1 *Use Case* Diagram 29](#_Toc80857330)

[3.3.3.2 *Activity* *Diagram* 32](#_Toc80857331)

[3.3.3.2.1 *Activity* *Diagram* *Tutorial* 32](#_Toc80857332)

[3.3.3.2.2 *Activity Diagram* Mulai Bermain 33](#_Toc80857333)

[3.3.3.3 Sequence Diagram 33](#_Toc80857334)

[3.3.3.3.1 Sequence Diagram Tutorial 34](#_Toc80857335)

[3.3.3.3.2 *Sequence Diagram* Mulai Bermain 34](#_Toc80857336)

[3.3.4 Perancangan Aplikasi 35](#_Toc80857337)

[3.3.4.1 HIPO Aplikasi 35](#_Toc80857338)

[3.3.4.2 Flowchart Aplikasi 36](#_Toc80857339)

[3.3.4.3 Perancangan Antarmuka / Interface 37](#_Toc80857340)

[3.3.4.3.1 Rancangan Tampilan Halaman Utama 37](#_Toc80857341)

[3.3.4.3.2 Rancangan Tampilan Halaman Pemilihan Kategori 37](#_Toc80857342)

[3.3.4.3.3 Rancangan Tampilan Halaman Tutorial 39](#_Toc80857343)

[3.3.4.3.4 Rancangan Tampilan Halaman Contoh Soal Mudah 39](#_Toc80857344)

[3.3.4.3.5 Rancangan Tampilan Halaman Contoh Soal Sedang 41](#_Toc80857345)

[3.3.4.3.6 Rancangan Tampilan Halaman Contoh Soal Sulit 42](#_Toc80857346)

[3.3.4.3.7 Rancangan Tampilan Halaman Jawaban Benar 43](#_Toc80857347)

[3.3.4.3.8 Rancangan Tampilan Halaman Jawaban Salah 44](#_Toc80857348)

[3.3.4.3.9 Rancangan Tampilan Halaman *Score* 45](#_Toc80857349)

[3.3.5 Simulasi 45](#_Toc80857350)

[3.3.6 Aturan Permain 47](#_Toc80857351)

[3.3.6.1 Latihan (*Tutorial*) 47](#_Toc80857352)

[3.3.6.2 Mulai 47](#_Toc80857353)

[3.3.7 Pemanggilan Acak Pada Model Soal 49](#_Toc80857354)

[BAB IV 50](#_Toc80857355)

[4.1 Tampilan Aplikasi Gerbang Logika 50](#_Toc80857356)

[4.1.1 Tampilah Halaman Utama 50](#_Toc80857357)

[4.1.2 Tampilan Halaman Tutorial 51](#_Toc80857358)

[4.1.3 Tampilan Halaman Kategori 56](#_Toc80857359)

[4.1.4 Tampilan Halaman Soal 63](#_Toc80857360)

[4.1.5 Tampilan Halaman *Score* 74](#_Toc80857361)

[4.2 Pengujian Aplikasi 77](#_Toc80857362)

[4.2.1 Pengujian Prosedur Latihan (*Tutorial*) 77](#_Toc80857363)

[4.2.2 Pengujian Tombol Mulai 79](#_Toc80857364)

[4.2.3 Pengujian *Spinner* 80](#_Toc80857365)

[4.2.4 Pengujian Tombol Mulai pada *Spinner* 82](#_Toc80857366)

[4.2.5 Pengujian Tombol Submit 85](#_Toc80857367)

[4.2.6 Pengujian Tombol *Continue* 88](#_Toc80857368)

[4.2.7 Pengujian Tombol Exit 88](#_Toc80857369)

[4.2.8 Pengujian Tombol *Back* pada Soal 88](#_Toc80857370)

[4.2.9 Pengujian Akurasi Jawaban 89](#_Toc80857371)

[4.3 Pembahasan Hasil 92](#_Toc80857372)

[BAB V 94](#_Toc80857373)

[5.1 Kesimpulan 94](#_Toc80857374)

[5.2 Saran 95](#_Toc80857375)

[DAFTAR PUSTAKA 96](#_Toc80857376)

# DAFTAR TABEL

[Tabel 2.1 Simbol – Simbol *Use Case Diagram* (Sukamto & Shalahudin, 2015) 11](#_Toc78323145)

[Tabel 2.2 Simbol pada *Activity Diagram* (Sukamto & Shalahudin, 2015) 15](#_Toc78323146)

[Tabel 2.3 Simbol pada *Sequence* *Diagram* (Sukamto & Shalahudin, 2015) 16](#_Toc78323147)

[Tabel 3.1 Kebutuhan Perangkat Keras……………………………………………27](#_Toc78323159)

[Tabel 3. 2 Tabel Kebutuhan Perangkat Lunak 27](#_Toc78323160)

[Tabel 3.3 Kebutuhan Fungsional Mahasiswa 28](#_Toc78323161)

[Tabel 3. 4 Spesifikasi Use Case Melakukan Tutorial 30](#_Toc78323162)

[Tabel 3. 5 Spesifikasi Use Case Mulai Bermain 30](#_Toc78323163)

[Tabel 3. 6 Deskripsi HIPO Aplikasi 35](#_Toc78323164)

[Tabel 4. 1 Kode Tampilan Halaman Utama ……………………………………50](#_Toc80292244)

[Tabel 4. 2 Kode Tampilan Halaman Tutorial 52](#_Toc80292245)

[Tabel 4. 3 Kode Tampilan Halaman Kategori 57](#_Toc80292246)

[Tabel 4. 4 Kode Tampilan Halaman Soal 63](#_Toc80292247)

[Tabel 4. 5 Kode Tampilan Halaman Score 75](#_Toc80292248)

[Tabel 4. 6 Pengujian Prosedur Latihan 78](#_Toc80292249)

[Tabel 4. 7 Pengujian Tombol Mulai 79](#_Toc80292250)

[Tabel 4. 8 Pengujian Spinner 80](#_Toc80292251)

[Tabel 4. 9 Pengujian Tombol Mulai pada Spinner (Benar) 82](#_Toc80292252)

[Tabel 4. 10 Pengujian Tombol Mulai pada Spinner (Salah) 84](#_Toc80292253)

[Tabel 4. 11 Pengujian Tombol Submit (Benar) 86](#_Toc80292254)

[Tabel 4. 12 Pengujian Tombol Submit (Salah) 87](#_Toc80292255)

[Tabel 4. 13 Pengujian Tombol Continue 88](#_Toc80292256)

[Tabel 4. 14 Pengujian Tombol Exit 88](#_Toc80292257)

[Tabel 4. 15 Pengujian Tombol Back pada Soal 88](#_Toc80292258)

[Tabel 4. 16 Pengujian Soal Kategorikal OR 90](#_Toc80292259)

[Tabel 4. 17 Pengujian Soal Kategorikal NOR 91](#_Toc80292260)

[Tabel 4. 18 Akurasi Jawaban 92](#_Toc80292261)

# DAFTAR GAMBAR

[Gambar 2. 1 Gerbang AND 6](#_Toc78323460)

[Gambar 2. 2 Gerbang NOT 6](#_Toc78323461)

[Gambar 2. 3 Gerbang OR 7](#_Toc78323462)

[Gambar 2. 4 Gerbang NAND 7](#_Toc78323463)

[Gambar 2. 5 Gerbang NOR 7](#_Toc78323464)

[Gambar 2. 6 Gerbang XOR 8](#_Toc78323465)

[Gambar 2. 7 Gerbang XNOR 8](#_Toc78323466)

[Gambar 2. 8 Metode Waterfall (Pressman, 2015) 9](#_Toc78323467)

[Gambar 2.9 Contoh Diagaram VTOC - HIPO (Tedjo, 2017) 19](#_Toc78323468)

[Gambar 3. 1 Struktur Organisasi Universitas Muhammadiyah Jakarta (Hanifah, 2018) ……………………………………………………………………………23](#_Toc79169411)

[Gambar 3. 2 Activity Diagram Sistem Berjalan 24](#_Toc79169412)

[Gambar 3. 3 Use Case Diagram Usulan 30](#_Toc79169413)

[Gambar 3. 4 Activity Diagram Tutorial 32](#_Toc79169414)

[Gambar 3. 5 Activity Diagram Mulai Bermain 33](#_Toc79169415)

[Gambar 3. 6 Sequence Diagram Tutorial 34](#_Toc79169416)

[Gambar 3. 7 Sequence Diagram Mulai Bermain 34](#_Toc79169417)

[Gambar 3. 8 Hipo Aplikasi 35](#_Toc79169418)

[Gambar 3. 9 Flowchart Aplikasi 36](#_Toc79169419)

[Gambar 3. 10 Rancangan Tampilan Halaman Utama 37](#_Toc79169420)

[Gambar 3. 11 Rancangan Tampilan Halaman Pemilihan Kategori 38](#_Toc79169421)

[Gambar 3. 12 Rancangan Tampilan Halaman Tutorial 39](#_Toc79169422)

[Gambar 3. 13 Rancangan Tampilan Halaman Contoh Soal Mudah 40](#_Toc79169423)

[Gambar 3. 14 Tampilan Halaman Contoh Soal Sedang 41](#_Toc79169424)

[Gambar 3. 15 Rancangan Tampilan Halaman Contoh Soal Sulit 42](#_Toc79169425)

[Gambar 3. 16 Rancangan Tampilan Halaman Jawaban Benar 43](#_Toc79169426)

[Gambar 3. 17 Rancangan Tampilan Halaman Jawaban Salah 44](#_Toc79169427)

[Gambar 3. 18 Rancangan Tampilan Halaman Score 45](#_Toc79169428)

[Gambar 3. 19 Gambar Saklar pada Aplikasi 46](#_Toc79169429)

[Gambar 3. 20 Gambar Lampu Mati dan Hidup pada Aplikasi 46](#_Toc79169430)

[Gambar 3. 21 Gambar Konduktor pada Aplikasi 46](#_Toc79169431)

[Gambar 3. 22 Gambar Massa pada Aplikasi 47](#_Toc79169432)

[Gambar 4. 1 Tampilan Halaman Utama ……………………………………50](#_Toc80292262)

[Gambar 4. 2 Tampilan Halaman Tutorial 52](#_Toc80292263)

[Gambar 4. 3 Tampilan Halaman Kategori 57](#_Toc80292264)

[Gambar 4. 4 Tampilan Halaman Soal 63](#_Toc80292265)

[Gambar 4. 5Tampilan Halaman Score 75](#_Toc80292266)

[Gambar 4. 6 Pengujian Prosedur Latihan 78](#_Toc80292267)

[Gambar 4. 7 Pengujian Tombol Mulai 80](#_Toc80292268)

[Gambar 4. 8 Pengujian Spinner Dificulty 81](#_Toc80292269)

[Gambar 4. 9 Pengujian Spinner Kategorikal 82](#_Toc80292270)

[Gambar 4. 10 Pengujian Tombol Mulai pada Spinner (Benar) 84](#_Toc80292271)

[Gambar 4. 11 Pengujian Tombol Mulai pada Spinner (Salah) 85](#_Toc80292272)

[Gambar 4. 12 Pengujian Tombol Submit (Benar) 86](#_Toc80292273)

[Gambar 4. 13 Pengujian Tombol Submit (Salah) 87](#_Toc80292274)

[Gambar 4. 14 Pengujian Tombol Back pada Soal 89](#_Toc80292275)

[Gambar 4. 15 Hasil Pengujian Soal Kategorikal OR 90](#_Toc80292276)

[Gambar 4. 16 Hasil Pengujian Soal Kategorikal NOR 91](#_Toc80292277)

# DAFTAR LAMPIRAN

[Lampiran 1 Kode Random 97](#_Toc78383824)

[Lampiran 2 Pemanggilan Model 98](#_Toc78383825)

[Lampiran 3 Bank soal dan Model 99](#_Toc78383826)

[Lampiran 4 Paging Lampu Hidup dan Mati 100](#_Toc78383827)

# BAB 1

**PENDAHULUAN**

## 1.1 Latar Belakang Masalah

Logika Informatika adalah salah satu mata kuliah Teknik Informatika di Fakultas Teknik Universitas Muhammadiyah Jakarta (FT-UMJ). Mata kuliah tersebut berisi tentang kegiatan yang berhubungan dengan penggunaan akal untuk menghasilkan suatu penalaran dengan kebenaran yang dapat dibuktikan secara matematis. Walaupun tanpa perhitungan dengan matemastis atau dengan statistik, tetapi dapat diuji dan masuk akal akan kebenarannya (Rezaldy, Hayadi, & Ropianto, 2021).

Dalam proses pembelajarannya, dosen pengampu yang berkompeten memberi pemahaman atau pengajaran ke mahasiswa dengan memberikan materi dan menjelaskannya. Selama proses pembelajaran mahasiswa diharapkan mampu memahami konsep logika informatika dimulai dari pengetahuan dasar tentang logika sampai pengoperasian logika terutama logika pada komputer.

Logika pada computer memiliki nilai kebenaran dimana jika benar maka nilainya 1 (TRUE) dan jika salah maka nilainya 0 (FALSE) serta memiliki operator logika yaitu AND, OR, NAND, NOR, XOR, dan XNOR. Semua aturan pengoperasian mengikuti aturan yang ada berdasarkan operator yang digunakan. Dalam kasus dua *input* dan satu *operator* kita dapat melihatnya pada referensi atau materi dasar kuliah Logika Informatika yang digunakan, akan tetapi jika *input* lebih dari dua, mahasiswa harus membuat table kebenaran yang banyak. Begitu juga dengan operator lebih dari dua membuat mahasiswa harus menggunakan penalaran lebih. Berdasarkan data hasil wawancara yang dilakukan peneliti kepada mahasiswa Teknik Informatika Universitas Muhammadiyah Jakarta bahwa mahasiswa mengalami kesulitan dalam menjawab soal dengan kasus input lebih dari dua dan operator lebih dari satu. Oleh sebab itu peneliti mengimplementasikan hasil belajar mata kuliah Logika Informatika dengan membuat sebuah Aplikasi *Game* Simulasi Gerbang Logika berbasis android.

Aplikasi Simulasi Gerbang Logika adalah aplikasi *game* berbasis android yang dibuat menggunakan metode pengembangan *software waterfall* dan *tools* berupa Unified Modeling Language (UML). Cara kerja aplikasi ini menerapkan metode *Aljabar Boolean.* Tujuan aplikasi ini dibuat sebagai pengimplementasian hasil belajar mata kuliah Logika Informatika. Dengan adanya aplikasi ini, pengguna aplikasi diharapkan mampu mengetahui hasil output dari pengoperasian logika berdasarkan nilai *input* beserta *operator* yang digunakan dan juga menambah minat pada mata kuliah Logika Informatika. Pada aplikasi ini pengimplementasiannya menggunakan sebuah lampu dan *input* digambarkan sebagai saklar pada aplikasi. Pensimulasian ini bertujuan mempermudah agar pengguna dapat memahami proses logika pada aplikasi. Nilai 1 (true) jika saklar dalam keadaan On dan nilai 0 (false) jika saklar dalam keadaan Off. Dimana jika *input* dimasukkan dan dioperasikan oleh *operator* jika hasilnya (*output*) benar maka lampu akan menyala dan jika hasilnya salah maka lampu tidak akan menyala.

Berdasarkan latar belakang tersebut maka, penulis memberi judul dalam penulisan ini **“APLIKASI SIMULASI GERBANG LOGIKA BERBASIS ANDROID SEBAGAI IMPLEMENTASI SIMULASI MATA KULIAH LOGIKA INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS MUHAMMADIYAH JAKARTA (FT-UMJ)”**.

## 1.2 Identifikasi Masalah

Berdasarkan gambaran dalam latar belakang diatas, maka dapat dirumuskan beberapa permasalahan yang akan dibahas sebagai berikut :

1. Untuk *input* lebih dari dua membutuhkan pembuatan table kebenaran yang banyak untuk mengetahui jawabannya*.*
2. Untuk *operator* lebih dari satu membutuhkan pengoperasian yang panjang dan penalaran yang lebih.

## 1.3 Rumusan Masalah

1. Bagaimana cara menerapkan metode Aljabar Boolean?
2. Bagaimana cara membuat Aplikasi Simulasi Gerbang Logika dengan mengimplementasikan cara kerja saklar lampu dalam menghidupkan lampu dua sampai empat input?
3. Bagaimana cara mengimplementasikan satu sampai tiga operator?

## 1.4 Batasan Masalah

1. Implementasi berdasarkan hasil belajar mata kuliah Logika Informatika Fakultas Teknik Muhammadiyah Jakarta terutama pada materi Gerbang Logika.
2. Jumlah input yang digunakan adalah dua sampai empat input.
3. Jumlah operator yang digunakan adalah satu sampai tiga operator.
4. Jenis operator yang digunakan (AND, OR, NAND, NOR, XOR, dan XNOR).
5. Tools yang digunakan adalah UML dan *text editor*.
6. Penerapan konsep kelistrikan menggunakan konsep dasar kelistrikan.

## 1.5 Tujuan Penelitian

Penelitian ini bertujuan untuk menciptakan Aplikasi Gerbang Logika yang berbasis *Operating System* Android dengan empat *input* yang mensimulasikan cara kerja saklar dalam menghidupkan lampu menggunakan prinsip logika dasar (AND, OR, NAND, NOR, XOR, dan XNOR) sebagai penerapan mata kuliah Logika Informatika. Aplikasi ini diharapkan dapat dijadikan bahan ajar tambah mata kuliah Logika Informatika pada Fakultas Teknik Muhammadiyah Jakarta.

## 1.6 Metodologi Penelitian

1. Pengumpulan data.
2. Metode Observasi

Observasi dilakukan dengan cara pengamatan selama 4 bulan secara langsung pada mata kuliah Logika Informatika Fakultas Teknik Universitas Muhammadiyah Jakarta untuk memperoleh data yang akan dijadikan bahan penulisan ilmiah Tugas Akhir.

1. Metode *Study* Kepustakaan.
2. Pengumpulan data yang dilakukan dengan wawancara dengan pihak yang terkait. Di penulisan ini Ibu Retnani Latifa sebagai dosen pengampu mata kuliah Logika Informatika Fakultas Teknik Universitas Muhammadiyah Jakarta dan mahasiswa Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Jakarta.
3. Analisis Data

Pada tahap ini, penulis menganalisis data yang akan digunakan sebagai dasar prinsip kerja pada Aplikasi Gerbang Logika yang akan dibuat.

1. Perancangan

Pada tahap ini data-data yang telah dikumpulkan dan dianalisis lalu dilakukan perancangan system model yang akan digunakan dalam pembuatan aplikasi. *Tools* yang digunakan *Unified Modeling Language* (UML) seperti *usecase diagram, activity diagram, Hierarchy plus Input-Output* (HIPO), dan *user interface* (UI) *Design Tools* menggunakan *Balsamiq*

1. Pembuatan Aplikasi

Pada tahap ini, dilakukan pembuatan aplikasi, setelah tahap perancangan system selesai. Aplikasi Gerbang Logika ini merupakan aplikasi berbasis android dengan menggunakan bahasa pemrograman Java.

1. Pengujian Aplikasi

Proses pengujian dalam penulisan ini yaitu dengan menggunakan pengujian *black box testing* dengan mengevaluasi dari sisi fungsional berdasarkan *input* dan *output*.

# BAB II

**TINJAUAN PUSTAKA**

## 2.1 Sejarah dan Pengertian Logika Informatika

Rasional berasal dari bahasa Yunani, menjadi spesifik Logos yang berarti kata-kata atau renungan yang dikomunikasikan dalam kata-kata, pemikiran atau yang berarti perenungan. Para ahli sepakat bahwa Rationale adalah investigasi aturan untuk menilai perselisihan dengan mencari tahu perselisihan mana yang substansial dan mengenali perselisihan besar dan buruk. Pada awalnya, rasional dikonsentrasikan sebagai bagian dari teori atau penalaran. Orang dapat mengembangkan informasi karena mereka memiliki kemampuan bahasa dan berpikir. (Rezaldy, Hayadi, & Ropianto, 2021)

Untuk memiliki pilihan untuk membuat kesimpulan yang tepat, diperlukan kemampuan berpikir. Kapasitas untuk menalar adalah kapasitas untuk mencapai kesimpulan yang tepat dari bukti atau kenyataan yang ada, dan seperti yang ditunjukkan oleh prinsip-prinsip tertentu. Rasional adalah bagian dari teori dan dapat dianggap sebagai bagian dari aritmatika. Rasional diurutkan sebagai aritmatika murni karena matematika adalah rasional yang tersusun.

Bagaimanapun, sejak tahun 1800-an pemikiran telah terkonsentrasi di bidang matematika dan saat ini juga di bidang rekayasa perangkat lunak, karena pemikiran juga mempengaruhi rekayasa perangkat lunak di bidang (peralatan) dan (pemrograman). Alasan di sini disebut alasan gambar karena merenungkan upaya untuk secara resmi mewakili upaya. Konsekuensinya, alasan tersebut disebut juga dengan alasan formal (formal rationale). Aristoteles dengan cepat memperhatikan, mengeksplorasi, dan mencatat hukum-hukum rasional formal, khususnya jenis pemikiran yang disebut logika yang terdiri dari beberapa premis dan satu ujung. Dasar pemikiran yang diciptakan oleh Aristoteles disebut juga dengan pemikiran gaya lama atau pemikiran Aristotelian.

## 2.2 Logika dan Komputer

Rekayasa kerangka kerja PC terbuat dari sirkuit rasional 1 (valid) dan 0 (palsu) yang digabungkan dengan berbagai pintu rasional AND, NOR, NOT, XOR, XNOR dan NAND. BUKAN pintu, DAN pintu masuk dan juga pintu masuk adalah tiga pintu utama dari pintu masuk dasar yang penting. Pintu lain yang merupakan gabungan dari tiga jalan masuk utama, khususnya Jalan Masuk NAND, Jalan Masuk NOR, Jalan Masuk XOR dan Jalan Masuk XNOR (Sugiartowo & Ambo, 2018).

1. Gerbang AND

Setidaknya ada dua saluran info dan saluran hasil di pintu masuk AND yang merupakan salah satu pintu rasional penting. Hasil ganda bergantung pada status informasi dan kapasitasnya pada pintu masuk AND. Kondisi hasil akan menjadi alasan 1 jika semua jalur informasi adalah alasan 1 yang merupakan standar fungsi dari pintu masuk AND. Bagaimanapun hasilnya akan menjadi alasan 0.



**Gambar 2. 1 Gerbang AND**

1. Gerbang NOT

Pintu NOT secara teratur disebut pintu masuk inverter. Pintu masuk alasan paling mudah untuk diingat adalah pintu NOT. Hanya memiliki satu jalur informasi dan satu jalur hasil adalah atribut pintu NOT. Nilai wajar yang meniadakan kondisi cerdas di saluran informasi akan selalu dibuat oleh pintu masuk NOT. Jika saluran informasi adalah rasional 1, saluran hasil akan menjadi rasional 0 dan sebaliknya, dengan asumsi saluran informasi adalah rasional 0, saluran hasil akan menjadi rasional 1.



**Gambar 2. 2 Gerbang NOT**

1. Gerbang OR

Pintu OR adalah salah satu jalan masuk dasar pemikiran yang memiliki setidaknya 2 saluran informasi dan saluran hasil. Meskipun jumlah saluran informasi yang dimiliki oleh pintu OR, sebenarnya memiliki pedoman kerja yang sama di mana kondisi hasil akan menjadi alasan 1 jika satu atau seluruh saluran informasi adalah alasan 1. Terlebih lagi, hasilnya adalah alasan 0.



**Gambar 2. 3 Gerbang OR**

1. Gerbang NAND

Pintu NAND adalah augmentasi dari pintu masuk AND. Pintu ini benar-benar merupakan pintu masuk DAN di mana pintu NOT diperkenalkan.



**Gambar 2. 4 Gerbang NAND**

1. Gerbang NOR

Pintu NOR adalah augmentasi dari pintu masuk OR. Peningkatan ini melalui pengenalan pintu NOT pada hasil dari pintu masuk OR. Gambar 4 menunjukkan perpaduan ini di samping gambar untuk pintu NOR. Karena pada dasarnya pintu OR yang hasilnya dibalik, tabel realitas adalah sesuatu yang bertentangan dengan tabel realitas dari pintu masuk OR.



**Gambar 2. 5 Gerbang NOR**

1. Gerbang XOR

Pintu XOR mewakili Selective OR dimana jika informasi tersebut memiliki alasan yang sama, hasilnya akan menjadi alasan 0 dan sebaliknya dengan asumsi informasi tersebut adalah alasan yang berbeda, hasilnya akan menjadi alasan 1.



**Gambar 2. 6 Gerbang XOR**

1. Gerbang XNOR

Pintu XNOR mewakili Selective NOT OR adalah sesuatu yang bertentangan dengan pintu masuk XOR dimana jika informasi adalah alasan yang sama, hasilnya akan menjadi alasan 1 dan sebaliknya dengan asumsi informasi adalah alasan yang beragam, hasilnya akan menjadi alasan 0.

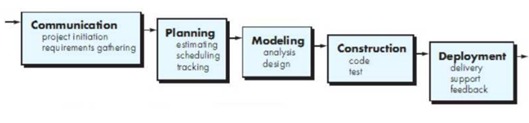


**Gambar 2. 7 Gerbang XNOR**

Program PC (*Personal* *Computer*) berjalan pada struktur pemikiran yang baik dari jawaban untuk suatu masalah dengan bantuan Asumsi… Kemudian, pada saat itu… LAINNYA, UNTUK… UNTUK… MELAKUKAN, SAAT, KASUS… DARI bagian-bagian program.

## 2.3 Waterfall Modell

Menurut (Pressman, 2015) model cascade adalah kemajuan pemrograman menggunakan model gaya lama yang metodis dan berurutan. Nama model ini benar-benar "Model Berturut-turut Lurus". Model ini sering disinggung sebagai "siklus hidup teladan" atau teknik kaskade. Model ini mendapat tempat dengan model pemrograman komputer konvensional dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970-an dengan 7 (tujuh) tahapan dan mengalami perbaikan serta perubahan hingga menjadi 5 (lima) tahapan sehingga model ini yang paling banyak dipakai dalam *Software Engineering* (SE). Metode ini mengambil pendekatan yang sistematis dan berurutan. Disebut *Waterfall* karena prosesnya bertahap.



**Gambar 2. 8 Metode Waterfall** (Pressman, 2015)

* + - 1. *Communication (Project Initiation & Requirements Gathering)*

Sebelum memulai pekerjaan khusus, sangat penting untuk berbicara dengan klien untuk memahami dan mencapai tujuan yang ingin dicapai. Hasil dari korespondensi ini adalah pernyataan usaha, misalnya, menyelidiki masalah yang dialami dan mengumpulkan informasi mendasar, serta membantu menentukan fitur dan elemen produk. Berbagai informasi tambahan juga dapat diambil dari buku harian, artikel, dan internet.

* + - 1. *Planning (Estimating, Scheduling, Tracking)*

Tahap selanjutnya adalah tahap penyusunan yang menjelaskan penilaian terhadap usaha khusus yang akan dilakukan, bahaya yang dapat terjadi, aset yang diharapkan untuk membuat kerangka, item pekerjaan yang akan diserahkan, perencanaan pekerjaan yang akan diselesaikan, dan tindak lanjut. dari ukuran kerja kerangka kerja.

* + - 1. *Modeling (Analysis & Design)*

Tahap ini adalah tahap rencana dan demonstrasi dari rekayasa kerangka kerja yang berpusat di sekitar rencana struktur informasi, model pemrograman, tampilan antarmuka, dan perhitungan program. Tujuannya adalah untuk lebih mudah memahami pandangan berkali-kali dari apa yang akan selesai.

* + - 1. *Construction (Code & Test)*

Tahap pengembangan ini adalah cara untuk membuat interpretasi dari struktur rencana menjadi kode atau struktur/bahasa yang dapat diuraikan oleh mesin. Setelah pengkodean selesai, dilakukan percobaan pada framework dan selanjutnya kode yang telah dibuat. Tujuannya adalah untuk menemukan kesalahan yang mungkin terjadi untuk diperbaiki kemudian.

* + - 1. *Deployment (Delivery, Support, Feedback)*

Tahap *Arrangement/Deployment* adalah tahap pelaksanaan pemrograman kepada klien, pemeliharaan pemrograman intermiten, perbaikan pemrograman, penilaian pemrograman, dan pengembangan pemrograman tergantung pada kritik yang diberikan dengan tujuan agar framework dapat terus berjalan dan berkreasi sesuai dengan kapasitasnya.

**Keuntungan Metode *Waterfall***

1. Sifat kerangka berikutnya umumnya sangat baik. Ini karena langkah eksekusi progresif dengan tujuan tidak hanya membidik pada tahapan tertentu.
2. Arsip perbaikan kerangka kerja yang sangat terkoordinasi, karena setiap tahap akan diselesaikan sepenuhnya sebelum melanjutkan ke tahap berikutnya. Jadi setiap stage atau stage akan memiliki record tertentu*.* (Saputra & Agustin, 2012)

**Kelemahan Metode *Waterfall***

1. Diperlukan administrasi yang baik, karena interaksi perbaikan tidak mungkin dilakukan berulang-ulang sebelum terjadinya suatu item.
2. Sedikit kesalahan akan menjadi masalah yang sangat besar jika tidak segera diketahui dalam perbaikan sistem.
3. Klien akan berpikir bahwa sulit untuk mengungkapkan kebutuhan mereka secara tegas sehingga mereka tidak dapat mengatasi kerentanan terhadap awal perbaikan. (Saputra & Agustin, 2012)

## 2.4 Unified Modeling Language (UML)

*Unified modelling language* (UML) merupakan sebuah “bahasa” yang sudah menjadi suatu standar pada bidang industri untuk visualisasi, merancang dan dokumentasikan system pada perangkat lunak. UML telah menawarkan sebuah standar untuk melakukan perancangan model sebuah sistem. UML juga digunakan untuk membuat model semua jenis aplikasi perangkat lunak, yang di mana pada aplikasi tersebut telah berjalan pada perangkat keras, suatu sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun (Yasin, 2012).

UML mendefinisikan notasi dan *syntax/*semantik. Notasi pada UML adalah suatu sekumpulan dalam bentuk khusus yang menggambarkan berbagai diagram piranti lunak. Setiap bentuk yang memiliki makna tertentu, dan UML pada *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut bisa dikombinasikan (Yasin, 2012).

Tujuan UML diantaranya adalah:

1. Model sistem (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi objek.
2. Menyediakan bahasa pemodelan yang tidak terpengaruh oleh berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktik terbaik yang termasuk dalam pemodelan.

Adapun jenis – jenis diagram *Unified modelling language* (UML), yaitu :

### 2.4.1 Use Case Diagram

*Use Case diagram* Merupakan model perilaku sistem informasi yang akan dibuat. *Use case* Menjelaskan interaksi antara satu atau lebih partisipan dan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk menentukan fungsi dalam sistem informasi dan orang yang memiliki akses ke fungsi tersebut. (Sukamto & Shalahudin, 2015)

Syarat penamaan pada *use case* Apakah namanya sesederhana dan semudah mungkin. Terdapat dua hal pokok pada *use case.* Pengertian apa yang disebut aktor dan *use case*.

1. Aktor merupakan manusia, proses atau sistem lain yang berinteraksi dengan sistem informasi selain sistem informasi yang akan dibuat, sehingga meskipun simbol partisipan adalah avatar seseorang, partisipan tersebut belum tentu orang.
2. *Use case* merupakan fungsi yang terdapat pada sistem sebagai modul untuk bertukar informasi antar aktor. (Sukamto & Shalahudin, 2015).

Selanjutnya adalah simbol-simbol yang terdapat pada diagram *Use Case* pada tabel 2.1 :

**Tabel 2.1 Simbol – Simbol *Use Case Diagram*** (Sukamto & Shalahudin, 2015)

| **Simbol** | **Deskripsi** | |
| --- | --- | --- |
| *Use case* | Fungsi yang terdapat pada sistem merupakan satu kesatuan untuk bertukar informasi antar unit atau partisipan. Seringkali diungkapkan oleh kata kerja di awal frase nama *use case*. | |
| Aktor / actor | Manusia, proses, atau sistem luar yang berinteraksi dengan sistem informasi akan dibuat di luar sistem informasi yang akan dirancang, sehingga meskipun simbol aktor adalah gambaran karakter, aktor tersebut belum pasti karakter. Biasanya menggunakan kata benda di awal frasa nama actor. | |
| Asosiasi / *association* | Interaksi antara aktor dan *use case* yang bekerja sama pada *usecase* atau *usecase* yang memiliki ikatan dengan aktor*.* | |
| Ekstensi / *extend* | Hubungan *usecase* tambahan di sebuah *use case* dimana *use case* yang ditambahkan bersifat *independent* walaupun tanpa *usecase* tambahan. Seringkali *usecase* tambahan memiliki nama depan yang identik dengan *usecase* yang ditambahkan, missal :    Arah menuju pada *use case* yang ditambahkan. Seringkali *usecase* induk identic atau sama dengan *usecase extend*-nya. | |
| Generalisasi / *generalization* | | Relasi general serta spesial (umum-khusus) pada dua buah *use case* yaitu fungsi yang lebih umum dari lainnya, contoh:    Arah menuju pada user yang general (umum). |
| Menggunakan / *include* | | Realasi *use case* tambahan pada sebuah *use case* di mana penambahan membutuhkan *use case* tersebut untuk melaksanakan tugasnya atau syarat pengeksekusian *use case* tersebut.  terdapat dua hal pokok berkaitan dengan *incude* pada *use case*:  *Include* yaitu pemanggilan *use case* |
|  | | berulang.   1. saat eksekusi *use case* tambahan, contoh pada kasus ini:     *Include* yaitu pemeriksaan berulang oleh *use case* yang ditambahkan apakah *use case* yang ditambah telah berjalan sebelum *use case* tambahan dieksekusi, contoh kasus:     1. selanjutnya pendapat berdasarkan pertimbangan dan penjelasan yang diperlukan, salah satu metode di atas atau keduanya dapat digunakan. |

### 2.4.2 Activity Diagram

*Activity diagram* mempresentasikan *workflow* (aliran kerja) aktivitas dalam sistem atau proses bisnis atau menu pada perangkat lunak. Perlu dicatat di sini bahwa diagram aktivitas menggambarkan aktivitas sistem, bukan perilaku peserta, sehingga sistem dapat melakukan aktivitas (Sukamto & Shalahudin, 2015).

Diagram aktivitas seringkali digunakan untuk mempresentasikan hal berikut :

1. Desain proses bisnis, di mana semua urutan *activity* yang dipaparkan adalah penentuan proses bisnis system.
2. Tahapan atau penggolongan tampilan dalam sistem / *User* interface setiap kegiatan dinilai memiliki desain antar muka tampilan.
3. Desain uji yang meyakini bahwa seluruh *activity* perlu diuji, dan pengujian membutuhkan penentuan kasus pengujian.
4. Desain menu yang terdapat di perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram *Activity Diagram* tertera pada tabel 2.2 :

**Tabel 2.2 Simbol pada *Activity Diagram*** (Sukamto & Shalahudin, 2015)

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Status awal  Description: Activity5 | Status awal aktivitas sistem, diagram aktivitas memiliki keadaan awal. |
| Aktivitas  Description: Activity | Aktivitas yang dilakukan sistem, aktivitas biasanya dimulai dengan kata kerja. |
| *Transition*  Description: Activity1 | Notasi jalur untuk menampilkan aliran kontrol *activity* ke *activity*. |
| Percabangan/ decision  Description: Activity2 | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| Penggabungan / join  Description: Activity3 | Asosiasi penggabungan dimana gabungkan lebih dari satu aktivitas menjadi satu. |
| Status akhir  Description: Activity6 | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir. |
| Swimlane  Description: Activity4 atau Description: Activity4 | Memisahkan bisnis yang bertanggung jawab atas aktivitas yang terjadi. |

### 2.4.3 Sequence Diagram

Diagram urutan(*Sequence*) mempresentasikan interaksi antar objek *internal* dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* biasanya digunakan untuk mendeskripsikan skenario atau serangkaian langkah yang dilakukan dalam menanggapi suatu situasi event untuk menghasilkan *output* tertentu .

Diagram *sequence* menjelaskan perilaku objek pada *use case* dengan menggambarkan kehidupan suatu objek dan *message* yang dikirimkan dan diterima antar objek. Jadi untuk menggambar diagram *sequence* maka harus mengetahui *use case* dan metode milik kelas yang dipakai untuk objek. Membuat diagram *Sequence* harus mengacu pada *use case* (Sukamto & Shalahudin, 2015).

Berikut adalah simbol yang terdapat di diagram *Sequence* tertera pada tabel 2.3 :

**Tabel 2.3 Simbol pada *Sequence* *Diagram*** (Sukamto & Shalahudin, 2015)

| **Simbol** | **Deskripsi** |
| --- | --- |
| Aktor    atau    tanpa waktu aktif | Orang-orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi akan dibuat di luar sistem informasi yang akan dibuat, sehingga meskipun simbol aktor adalah gambaran dari suatu karakter, aktor tersebut belum tentu karakter. Biasanya adalah kata benda sebelum nama actor. |
| Garis hidup / lifeline | Menyatakan kehidupan suatu hal |
| Objek | Menyatakan *object* yang menginteraksikan pesan |
| Waktu aktif | Nyatakan bahwa *object* tersebut aktif serta interaktif, dan segala hal yang berkaitan dengan kegiatan ini berada dalam sesi.    maka cek status *login* dan *open* dilakukan di dalam metode login Aktor tidak memiliki waktu aktif. |
| Pesan tipe create | Nyatakan bahwa satu objek membuat objek lain, arah panah menunjuk ke objek yang sedang dibuat. |
| Pesan tipe send | Tunjukkan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lain, dan panah menunjuk ke objek yang akan dikirim. |
| Pesan tipe call | Nyatakan bahwa sebuah objek memanggil operasi / metode pada objek lain atau dirinya sendiri,    arah panah menunjuk ke objek dengan operasi / metode, karena memanggil operasi / metode, maka yang disebut operasi / metode harus ada pada diagram kelas sesuai kelas objek interaktif. |
| Pesan tipe return | Tunjukkan bahwa objek yang melakukan operasi atau metode mengembalikan objek tertentu, dan panah menunjuk ke objek yang menerima nilai pengembalian. |
| Pesan tipe destroy | Nyatakan bahwa suatu benda mengakhiri umur benda lain, dan panah menunjuk ke ujung benda tersebut, sebaiknya jika benda itu ada create maka ada destroy. |

## 2.5 Flowchart

*Flowchart* adalah diagram proses yang terdiri dari Diagram alir dihasilkan dengan mendeskripsikan langkah-langkah untuk memecahkan masalah atau mendeskripsikan secara grafis langkah-langkah dan urutan proses program (Ladjamudin, 2013).

Sistem *flowchart* menentukan urutan pemrosesan dalam sistem dengan menampilkan perangkat media masukan, proses pengolahan data dan jenis media penyimpanan. *Flowchart* terstruktur dengan simbol, yang digunakan seperti alat untuk presentasikan proses pada program*.*

**Tabel 2.4 *Flowcart Symbols*** (Ladjamudin, 2013)

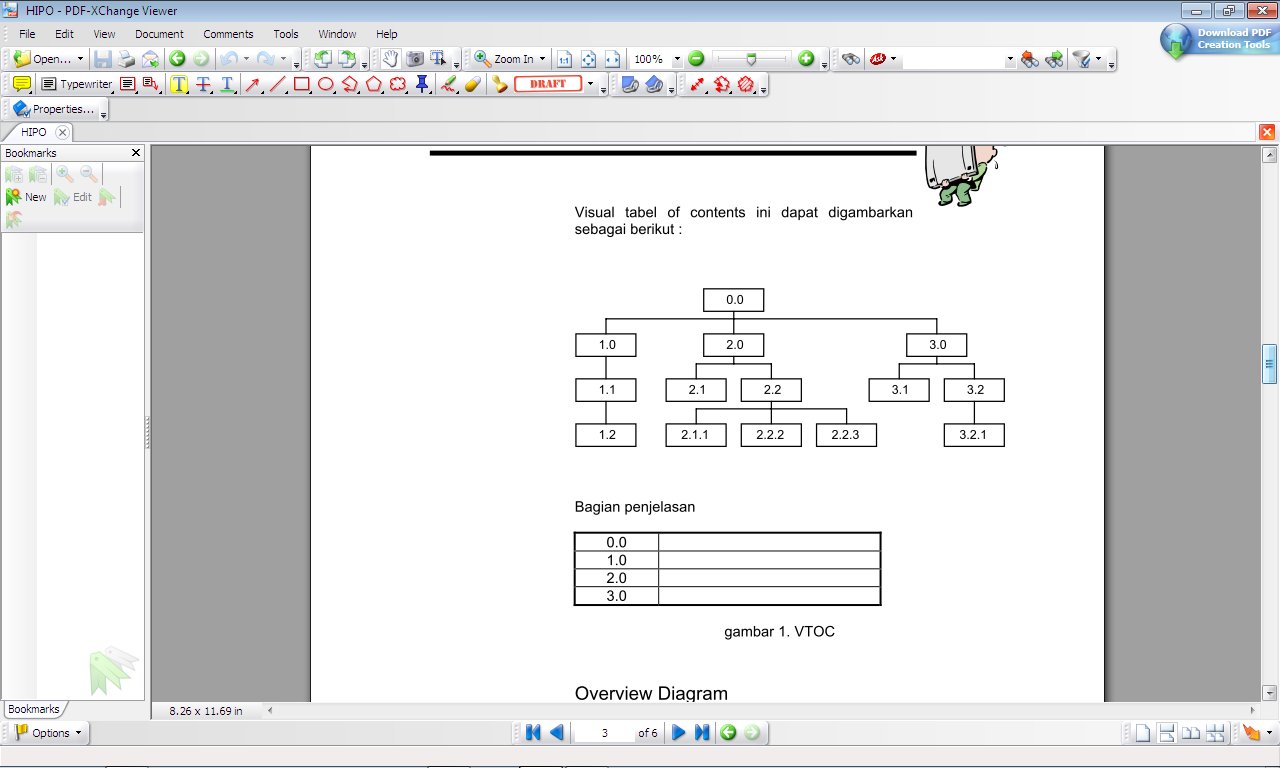
|  |  |  |
| --- | --- | --- |
| **N o** | **S i m b o l** | **Keterangan** |
| 1 |  | **D o k u m e n**  Menunjukan dokumen berupa dokumen input dan output  untuk proses manual dan proses mekanis atau berbasis komputer. |
| 2 |  | **Proses Manual**  Menunjukan proses yang dilakukan secara manual. |
| 3 |  | **Simbol Proses Terkomputerisasi**  Menunjukan kegiatan proses yang dilakukan secara komputerisasi. |
| 4 |  | **File Harddisk**  Menunjukan media penyimpanan data / informasi file tentang proses berbasis komputer. File dapat disimpan di lemari file, map dll. |
| 5 |  | **Pengarsipan (*Offline Storage*)**  Menunjukan Informasi file dalam penyimpanan data non-komputer / pemrosesan manual. |
| 6 |  | **Penghubung**  Menunjukan Alur dokumen pada halaman alur dokumen yang sama terputus atau terpisah. |
| 7 |  | **Arah Alir Dokumen**  Menunjukan Arah aliran dokumen antara bagian-bagian sistem yang relevan. Bisa dari luar sistem atau dari luar ke sistem dan antar bagian luar sistem. |

## 2.6 Hierarchy Plus Input Output (HIPO)

Tedjo menjelaskan bahwa *Hierarchy plus Input-Proses-Output* (HIPO) Ini adalah alat dokumentasi program. sekarang juga, HIPO ini juga banyak digunakan sebagai alat desain dan teknologi dokumen dalam siklus pengembangan sistem. HIPO berdasarkan fungsinya, yaitu setiap modul dalam sistem dijelaskan oleh fungsi utamanya (Tedjo, 2017) sama seperti DFD, Deskripsi di tingkat HIPO Pertama-tama jelaskan fungsi utama, lalu bagi fungsi utama menjadi level yang lebih rendah. sehingga, HIPO Alat dan teknologi dokumentasi yang digunakan sebagai fungsi program, tujuan utamanya adalah sebagai berikut :

1. Menyediakan struktur yang memungkinkan pemahaman tentang fungsi sistem.
2. Untuk menjelaskan fungsi yang akan dilakukan oleh program daripada menentukan pernyataan program untuk melakukan fungsi ini.
3. Memberikan gambaran visual tentang input yang akan digunakan dan output yang akan dihasilkan oleh setiap fungsi di setiap level diagram.
4. Menghasilkan keluaran yang benar dan memenuhi kebutuhan pengguna.

Salah satu jenis HIPO adalah Daftar Isi Visual/ *Visual Tabel of Contents* (VTOC), terdiri dari satu atau lebih grafik hierarki. VTOC menelaskan keseluruhan program HIPO baik ringkasan terperinci dan terstruktur. Di gambar ini, nama dan nomor program HIPO diidentifikasikan. Diagram struktur paket dan hubungan fungsional juga diidentifikasi dalam bentuk struktur hierarki. Deskripsi setiap fungsi termasuk dalam bagian deskripsi diagram ini. VTOC ini dapat digambarkan sebagai berikut :



Gambar 2.9 Contoh Diagaram VTOC - HIPO (Tedjo, 2017)

## 2.7 Pengujian Perangkat Lunak

*Software* pengujian merupakan teknik penting untuk perbaikan dan pengukuran kualitas sistem perangkat lunak. Pengujian juga memberikan pandangan mengenai perangkat lunak secara objektif dan independen, yang bermanfaat dalam operasional bisnis untuk memahami tingkat resiko pada implementasinya.

Untuk pengujian yang lengkap maka suatu perangkat lunak harus diuji dengan *white box testing* dan *black box testing*, tetapi pada penelitian ini hanya menggunakan pengujian *black box testing* (Pressman, 2015).

### 2.7.1 Black Box Testing

Menurut Pressman dalam bukunya Rekayasa Perangkat Lunak : (Pressman, 2015) “Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Pengujian *black box* memungkinkan para insinyur perangkat lunak untuk mendapatkan sekumpulan kondisi input yang dapat memanfaatkan sepenuhnya semua persyaratan fungsional program”

Pengujian *Black box* bukan merupakan *alternative* dari teknik *white box*, tetapi ini adalah metode tambahan yang mungkin dapat menemukan kategori kesalahan, bukan metode *white box*. Percobaan *Black box* berusaha mencari kesalahan dalam kategori sebagai berikut : (Pressman, 2015)

1. Fungsi salah atau hilang.
2. Kesalahan pada *interface*.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan dalam kinerja.
5. Inisialisasi dan kesalahan terminasi.

Pengujian *black box* cenderung diaplikasikan selama tahap akhir pengujian, karena pengujian *black box* memperhatikan struktur *control* maka perhatian berfokus pada domain informasi.

# BAB III

**METODOLOGI PENETIAN**

## 3.1 Tinjauan Organisasi

Pada tinjauan organisasi, peneliti memberikan sedikit informasi tentang sejarah dan struktur organisasi tempat dilakukannya penelitian. Adapun tempat penelitian yang dilakukan adalah di Fakultas Teknik Universitas Muhammadiyah Jakarta.

### 3.1.1 Sejarah Organisasi

Salah satu pilihan sidang Panel Muktamar yang digelar di Kalongan Utara adalah membangun Tenaga Hukum dan Teori di Padang Panjang. Staf ini resmi dibuka pada tanggal 3 Rabi'ul menjelang akhir tahun 1375 H, bertepatan dengan tanggal 18 November 1955, yang pada tahun 1956 kemudian dipindahkan ke Jakarta, dengan nama baru Sekolah Pendidik (PTPG). Apalagi PTPG sudah dirintis pada 18 November 1975.

Pada tahun 1958, PTPG muhammadiyah diubah menjadi Personil Penyiapan Instruktur dan Persekolahan (FKIP) dan berada di bawah Perguruan Tinggi Muhammadiyah Jakarta (UMJ), dengan pimpinan utama perguruan tinggi tersebut adalah dr.H. Ali Akbar, Sementara itu sebagai petinggi FKIP terpilih RH. Mubangit ronodihardjo. Pada tanggal 21 September 1961 dibuka Tenaga Kerja Bantuan Sosial Pemerintah (FKS) yang dimulai oleh Pendeta Silaturahmi Bapak Mulyadi Djojomartono. Anggota senior utama FKS adalah Prof. Mr. H sumantri praptokuso yang pada saat itu menjabat sebagai sekretaris jenderal dinas sosial.

Pada tahun 1962, tenaga Tarbiyah dibuka, dan pada tahun 1963 dibuka 3 (tiga) sumber daya, yaitu Tenaga Hukum, Tenaga Perancangan dan Tenaga Kerja Bidang Keuangan. Selain itu, pada tanggal 19 Juni 1963, UMJ dikukuhkan pendiriannya melalui Akta Akuntan Publik Raden Soerojo Wongsowidjojo di Jakarta. Hingga saat ini, UMJ memiliki 9 (Sembilan) Sumber Daya dengan 43 Proyek Investigasi, antara lain Tenaga Ahli Teori Sosial Politik, Tenaga Hukum, Tenaga Bidang Keuangan, Tenaga Perancangan, Tenaga Keagamaan Islam, Tenaga Agribisnis, Tenaga Kerja Kedokteran, dan Kesejahteraan, Tenaga Kependidikan, dan Tenaga Keperawatan dan Sekolah Pascasarjana, yang memiliki beberapa proyek ahli, mengingat Atasan Hukum, Ahli Ujian Islam, Ahli Administrasi, Atasan Ilmu Regulasi, Ahli Pembukuan, Ahli dalam Kesejahteraan Umum, dan Ahli Keperawatan. Saat ini UMJ sedang membuka program doktor di bidang Agama, Eksekutif dan Bisnis, seperti Hukum (Universitas Muhammadiyah Jakarta, 2021).

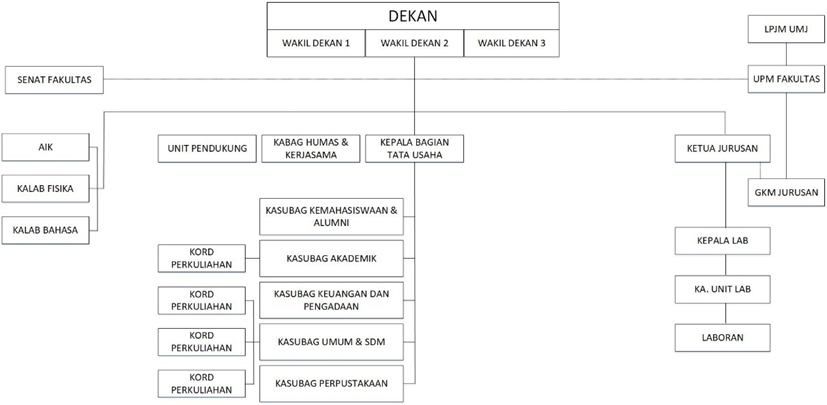
### 3.1.2 Visi dan Misi

Berikut adalah visi dan misi dari Fakultas Teknik Universitas Muhammadiyah Jakarta (Universitas Muhammadiyah Jakarta, 2021) :

1. Visi
2. Terkemuka memiliki arti dan tujuan menjadi unggulan tingkat nasioanl
3. Islami mempunyai arti kepribadian yang berakhlak dan
4. Modern mempunyai arti berbasis pada system informasi teknologi.
5. Misi
6. Menyelenggarakan proses pembelajaran yang memiliki keunggulan dibidang Ilmu Teknik.
7. Menyelenggarakan penelitian yang inovatif dan pengabdian masyarakat yang dapat meningkatkan kesejahteraan manusia.
8. Mengembangkan sumber daya manusia (dosen dan tenaga kependidikan) yang berkualitas.
9. Menjalin kerjasama yang produktif dan berkelanjutan dengan lembaga pendidikan, pemerintah dan dunia industry.
10. Menyelenggarakan pembinaan civitas akademika dalam kehidupan Islami.
11. Menanamkan sikap profesionalisme dan etos kewirausahaan.

### 3.1.3 Struktur Organisasi

Berikut adalah struktur organisasi Fakultas Teknik Universitas Muhammadiyah Jakarta (Hanifah, 2018):



**Gambar 3. 1 Struktur Organisasi Universitas Muhammadiyah Jakarta** (Hanifah, 2018)

## 3.2 Analisis Sistem Berjalan

Analisis sistem yang berjalan akan digambarkan menggunakan *activity diagram.* Berikut adalah alur kejadian dari sistem yang saat ini berjalan pada Fakultas Teknik Universitas Muhammadiyah Jakarta dalam prosedur pembelajaran Logika Informatika.

### 3.2.1 Prosedur Sistem Berjalan

Prosedur system berjalan pada pembelajaran mata kuliah Logika Informatika di Fakultas Teknik Universitas Muhammadiyah Jakarta memiliki dua prosedur yaitu prosedur pembahasan materi dan prosedur pengerjaan soal. Berikut ini prosedur system pembelajaran gerbang logika:

* + - 1. Prosedur Pembahasan

Dosen pengampu mata kuliah Logika Informatika membahas materi kepada mahasiswa. Pembahasan dimulai dari pengertian Logika sampai jenis-jenis operator yang digunakan. Operator yang digunakan dimulai dari AND, OR, NAND, NOR, XOR, dan XNOR. Sejak saat itu, dosen yang bertanggung jawab atas kursus juga memberikan contoh pertanyaan dan cara mengatasinya.

* + - 1. Prosedur evaluasi pembelajaran

Evaluasi pembelajaran berupa tugas yang diberikan dosen. Pada prosedur ini, mahasiswa mengerjakan soal dan menjawabnya berdasarkan soal yang diberikan. Hasil dari proses ini berupa nilai yang menjadi tolak ukur pemahaman mahasiswa terhadap materi yang diberikan.

### 3.2.2 Activity Diagram Sistem Berjalan

Berikut adalah *activity diagram* system berjalan pembelajaran Logika Informatika di Fakultas Teknik Universitas Muhammadiyah Jakarta :

**Gambar 3. 2 Activity Diagram Sistem Berjalan**

Gambar 3.2 diatas menjelaskan bagaimana proses pembelajaran Logika Informatika di Fakultas Teknik Universitas Muhammadiyah Jakarta yaitu dosen memberikan pengajaran kepada mahasiswa berdasarkan modul pembelajaran. Selanjutnya dosen akan memberikan *test* kepada mahasiswa. Test berupa soal yang memiliki tingkat kesulitan variatif. Selanjutnya mahasiswa mengerjakan test tersebut. Setelah mengerjakan test, mahasiswa akan memberikan test yang sudah dikerjakan kepada dosen. Dosen mengkoreksi hasil test dan memberi nilai berdasarkan jumlah jawaban benar dan salah. Nilai tersebut diberikan kepada mahasiswa agar mahasiswa dapat mengetahui hasil test tersebut.

### 3.2.3 Permasalahan Sistem Berjalan

Permasalahan yang terjadi pada system berjalan proses pembelajaran mata kuliah Logika Informatika di Fakultas Teknik Universitas Muhammadiyah Jakarta:

* + - 1. Untuk *input* lebih dari dua membutuhkan pembuatan tabel kebenaran yang banyak untuk mengetahui jawabannya.
      2. Untuk *operator* lebih dari satu membutuhkan pengoperasian yang panjang dan penalaran yang lebih.
      3. Test memiliki tingkat kesulitan variatif pada tiap soalnya.
      4. Dosen mengkoreksi jawaban secara manual.
      5. Dosen membutuhkan waktu untuk mengkoreksi.

## 3.3 Sistem Usulan

Didalam system usulan ini penulis akan membuat sebuah aplikasi game Gerbang Logika berbasis android. Pada aplikasi terdapat satu *actor* yaitu Mahasiswa. Metode *waterfall* digunakan pada pengembangan perangkat lunak pada system usulan ini.

### 3.3.1 Prosedur Sistem Usulan

* + - 1. Prosedur Tutorial

Aktor Mahasiswa dapat melakukan prosedur tutorial. Pada prosedur tutorial, aktor Mahasiswa dapat mempelajari materi gerbang logika secara mandiri.

* + - 1. Prosedur Bermain

Aktor Mahasiswa dapat melakukan prosedur bermain dengan cara menekan tombol Mulai pada aplikasi setelah itu memilih salah satu kategori dan kemudian aplikasi akan menampilkan testberdasarkan kategori yang dipilih.

Kategori soal dibagi menjadi dua, yaitu *random* dan kategorikal. Pada pilihan ***random***terdapat tiga pilihan yaitu mudah*,* sedang*,* dan sulit*.* Soal akan ditampilkan berdasarkan tiga pilihan tersebut dengan penggunaan *operator* pada rangkaian secara acak.

Mudah : jumlah *input* yang digunakan adalah dua dan jumlah operator yang digunakan adalah satu untuk setiap rangkaian pada soal. Soal yang ditampilkan adalah sebanyak sepuluh.

Sedang : jumlah *input* yang digunakan adalah tiga dan jumlah operator yang digunakan adalah satu sampai dua untuk setiap rangkaian pada soal. Soal yang ditampilkan adalah sebanyak sepuluh.

Sulit : jumlah *input* yang digunakan adalah empat dan jumlah *operator* yang digunakan adalah satu sampai tiga untuk setiap rangkaian pada soal. Soal yang ditampilkan adalah sebanyak sepuluh.

Pada pilihan **kategorikal** terdapat enam jenis soal berdasarkan *operator* yang digunakan pada rangkaian setiap soal yaitu AND, OR, NAND, NOR, XNOR, dan XOR. Masing-masing pilihan akan menampilkan sepuluh soal yang terdiri dari dua soal mudah*,* empat soal sedang*,* dan empat soal sulit. System akan memilih secara acak soal tersebut.

Jika telah menyelesaikan soal maka system akan menampilkan halaman score yang berisi jenis kategori yang dipilih, *score* yang diperoleh serta tombol *continue* dan *exit*. Perolehan *score* dihitung berdasarkan jumlah jawaban yang benar dan salah. Untuk jawaban benar akan diberi nilai sepuluh dan nol untuk jawaban yang salah. Tombol *continue* untuk menuju halaman kategori befungsi jika ingin melanjutkan permainan. Tombol *exit* untuk menuju halaman utama berfungsi untuk menyudahi permainan.

### 3.3.2 Metode Pengembangan Perangkat Lunak

Metode yang digunakan untuk pengembangan perangkat lunak (*System Development Life Cycle*) pada penelitian ini adalah Metode *Waterfall.* Berikut adalah proses dari metode *waterfall* :

1. Communication (Project Initiation & Requirements Gathering)

Pada tahap ini, pencarian informasi mengenai system yang akan dibuat menggunakan ditemukan kebutuhan pengguna. Berikut adalah analisis kebutuhan system

Kebutuhan non-fungsional (kebutuhan User)

1. Kebutuhan Perangkat Keras

Pembangunan dan pengujian aplikasi menggunakan perangkat keras pada table 3.1:

**Tabel 3.1 Kebutuhan Perangkat Keras**

|  |  |
| --- | --- |
| **Nama Komponen** | **Keterangan** |
| Versi Android | 8.0 |
| RAM | 2 GB |
| Penyimpanan Internal | 32 GB |

1. Kebutuhan Perangkat Lunak

Adapun kebutuhan perangkat lunak untuk membangun aplikasi dapat dilihat pada table 3.2:

**Tabel 3. 2 Tabel Kebutuhan Perangkat Lunak**

|  |
| --- |
| *PlayStore* yang mendukung, sebagai alat untuk mengunduh aplikasi. |
| *Android Studio,* digunakan sebagai *editor text* dalam pengkodean. |
| *Balsamiq Mockups,* digunakan untuk merancang *interface /* antar muka aplikasi. |
| *Star UML,* Digunakan untuk merancang UML dari aplikasi yang akan dibangun. |

1. Kebutuhan Aplikasi

* Aplikasi dapat memberikan pemahaman tentang materi gerbang logika.
* Aplikasi dapat menyajikan sebuah *game* gerbang logika
* Aplikasi dapat memberikan pilihan tingkat kesulitan *game.*

1. Planning (Estimating, Scheduling, Tracking)

Pada tahap ini menjelaskan spesifikasi kebutuhan *user* mengenai system yang akan dibuat. Berikut adalah analisis kebutuhan system:

Kebutuhan Fungsional

Selanutnya yaitu penggunaan kebutuhan fungsional untuk membangun aplikasi pada table 3.3

**Tabel 3.3 Kebutuhan Fungsional Mahasiswa**

|  |  |
| --- | --- |
| *Actor* | Keterangan Sistem |
| Mahasiswa | Mahasiswa dapat menggunakan fitur *tutorial.* |
| Mahasiswa dapat memilih tingkat kesulitan *game.* |
| Mahasiswa dapat memainkan *game.* |
| Mahasiswa dapat melihat *score*. |

1. Modelling (Analysis & Design)

*Modeling* merupakan tahapan perancangan sistem yang di dalamnya dilakukan pemodelan sistem. Pada tahap ini pemodelan pada sistem yang akan dibuat adalah menggunakan *Unified Modeling Language* (UML). UML yang digunakan tersebut adalah *Use Case Diagram*, *Activity Diagram* dan *Sequence Diagram*. Selain itu, *Hierarchy Plus Input Process Output* (HIPO) dan *Balsamiq Mockups* digunakan untuk pemodelan aplikasi*.*

1. Construction (Code & Test)

Pada sesi ini akan dilakukan pembangunan aplikasi yang diinginkan berdasarkan dengan fungsinya menggunakan bahasa pemorograman *javascript* dikarenakan Bahasa pemrograman tersebut yang paling mudah untuk dipelajari.

1. Deployment (Delivery, Support, Feedback)

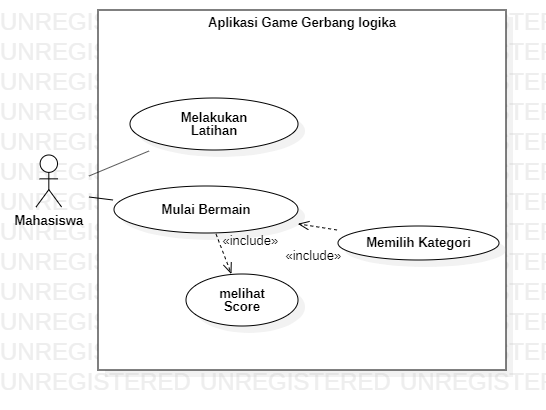
Setelah tahap *coding* dilakukan, selanjutnya adalah tahap implementasi ke *customer* serta pengetesan dari komponen aplikasi yang dibuat. Pada tahap ini akan dilakukan pengecekan kesesuaian fungsi dari komponen aplikasi yang dibuat. Penelitian ini sendiri menggunakan teknik pengujian *Black Box.*

### 3.3.3 Perancangan UML Sistem Usulan

Berikut adalah prosedur system usulan dan perancangan system usulan dengan menggunakan beberapa diagram UML (*Unified Modelling Language*) yaitu *use case diagram, activity diagram,* dan *sequence diagram.*

#### 3.3.3.1 *Use Case* Diagram

Pada use case diagram terdapat satu aktor yaitu Mahasiswa. Mahasiswa dapat melakukan Tutorial dan Mulai Bermain*.* Berikut adalah rancangan *use case diagram* system usulan.



**Gambar 3. 3 Use Case Diagram Usulan**

Aktor Mahasiswa dapat melakukan Latihan atau *tutorial* dengan cara memilih latihan pada aplikasi. Pada Tutorial, aktor Mahasiswa dapat mempelajari cara memainkan Aplikasi *Game* Gerbang Logika berikut dengan penjelasannya. Mahasiswa dapat memulai permainan dengan memilih tombol Start kemudian pilih salah satu kategori yang akan dimainkan.Permainan yang disajikan berupa soal *puzzle* melengkapi sebuah rangkaian gerbang logika. Mahasiswa ditantang untuk melengkapi rangkaian dengan cara memilih satu jawaban benar. Jika jawaban benar maka lampu akan hidup dan jika jawaban salah maka lampu akan mati. Jika sudah menyelesaikan permainan maka aplikasi akan menampilkan *Score* atau nilai berdasarkan jawaban benar dan salah.

**Tabel 3. 4 Spesifikasi Use Case Melakukan Tutorial**

|  |  |
| --- | --- |
| *Use Case Name* | Melakukan Tutorial |
| *Actor* | Mahasiswa |
| *Description* | Untuk mengetahui cara bermain |
| *Pre-Condition* | Aktor membuka Aplikasi |
| *Flow of Event* | 1. Aktor membuka Aplikasi 2. Pilih menu Latihan |

Pada tabel 3.4 terdapat *use case* melakukan tutorial dan aktor yaitu mahasiswa. Tujuan melakukan tutorial adalah untuk mengetahui cara bermain. Untuk melakukan tutorial mahasiswa harus membuka aplikasi terlebih dahulu lalu pilih menu Tutorial pada aplikasi.

**Tabel 3. 5 Spesifikasi Use Case Mulai Bermain**

|  |  |
| --- | --- |
| *Use Case Name* | Pilih Tingkat Kesulitan |
| *Actor* | Mahasiswa |
| *Description* | Memulai permainan |
| *Pre-Condition* | Aktor membuka Aplikasi |
| *Flow of Event* | 1. Aktor membuka Aplikasi 2. Pilih menu *Mulai* 3. System akan menampilkan pilihan *dificulty* dan kategorikal*.* 4. Aktor memilih salah satu. 5. System menampilkan soal bedasarkan kategori yang dipilih 6. System akan menampilkan *score* diakhir permainan |

Pada tabel 3.5 terdapat *use case* mulai bermain dan aktor mahasiswa. Tujuan fungsi ini agar aktor dapat bermain dan menyesuaikan kategori yang diinginkan dengan membuka aplikasi kemudian klik menu Start kemudian system akan menyajikan kategori *game* yaitu *random* dan kategorikal. Jika pilih *random* maka system akan mennyajikan pilihan mudah*,* sedang*,* dan sulit. Jika pilih kategorikal maka system akan menampilkan pilihan AND, OR, NAND, NOR, XOR, dan XNOR. Aktor memilih salah satu. Kemudian mahasiswa megerjakan soal. Setelah itu diakhir permainan mahasiswa dapat melihat *score.*

#### 3.3.3.2 *Activity* *Diagram*

Berikut adalah *activity* diagram untuk menggambarkan *work flow* pada sistem atau proses bisnis atau menu pada system usulan.

##### 3.3.3.2.1 *Activity* *Diagram* *Tutorial*

Sistem usulan *Tutorial* dapat kita lihat pada gambar 3.4 adalah sebagai berikut:



**Gambar 3. 4 Activity Diagram Tutorial**

Gambar 3.4 menjelaskan *activity diagram Tutorial.* Mahasiswa membuka aplikasi *game* gerbang logika dan system akan menampilkan halaman utama. Setelah itu halaman tutorial tampil setelah mahasiswa memilih Tutorial atau latihan pada system.

##### 3.3.3.2.2 *Activity Diagram* Mulai Bermain

****

**Gambar 3. 5 Activity Diagram Mulai Bermain**

Gambar 3.5 menjelaskan *activity diagram* bermain. Diawali dengan mahasiswa membuka aplikasi kemudian system akan menampilkan halaman utama. Setelah itu pilih *button Start* kemudian system akan menampilkan pilihan *random* dan kategorikal. Jika pilih *Random* maka akan terdapat pilihan mudah*,* sedang*,* dan sulit*.* Jika pilih Kategorikal maka akan terdapat pilihan AND, OR, NAND, NOR, XOR, dan XNOR. Mahasiswa memilih salah satu.Saat permaian sudah berakhir maka system akan menampilkan *Score.*

#### 3.3.3.3 Sequence Diagram

Berikut adalah *sequence diagram* pada aplikasi *game* gerbang logika. Terdapat dua *sequence diagram,* yaitu *sequence diagram* tutorial dan *sequence diagram* mulai bermain.

##### 3.3.3.3.1 Sequence Diagram Tutorial

****

**Gambar 3. 6 Sequence Diagram Tutorial**

Mahasiswa membuka aplikasi kemudian system akan menampilkan halaman utama. Kemudian mahasiswa pilih *button tutorial* dan system akan memanggil Halaman Tutorial untuk ditampilkan.

##### 3.3.3.3.2 *Sequence Diagram* Mulai Bermain

****

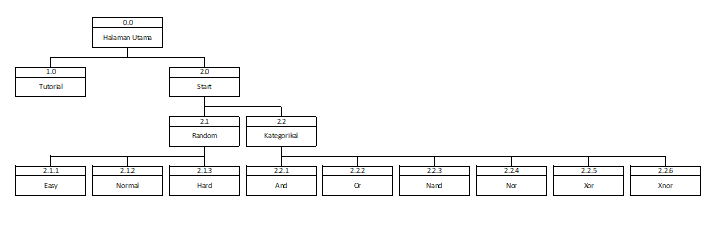
**Gambar 3. 7 Sequence Diagram Mulai Bermain**

Mahasiswa membuka aplikasi kemudian system akan menampilkan Halaman Utama. Untuk memulai permainan, mahasiswa pilih *button Start* dan system akan memanggil Halaman Kategori untuk ditampilkan. Pada halaman Kategori terdapat pilihan *Random* dan Kategorikal. Pada pilihan *random* terdapat pilihan mudah*,* sedang*,* dan sulit. Pada pilihan Kategorikal terdapat pilihan berdasarkan operator yang digunakan pada soal yaitu And, Or, Nand, Nor, Xor, Xnor.Kemudian mahasiswa hanya bisa memilih salah satu dari pilihan tersebut. Setelah memilih kemudian system akan menampilkan soal. setelah menjawab semua soal maka system akan menampilkan halaman *score*.

### 3.3.4 Perancangan Aplikasi

Berikut adalah perancangan aplikasi meliputi diagram HIPO, *Flowchart,* dan perancangan antarmuka / *Interface*.

#### 3.3.4.1 HIPO Aplikasi



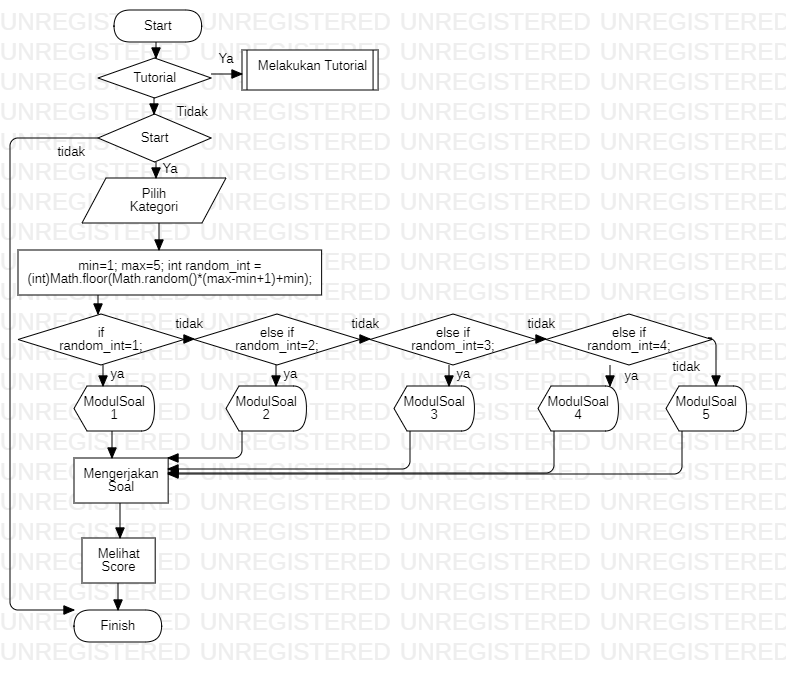
**Gambar 3. 8 Hipo Aplikasi**

Gambar 3.13 menjelaskan menu yang dapat diakses mahasiswa yang akan dideskripsikan oleh table dibawah ini:

**Tabel 3. 6 Deskripsi HIPO Aplikasi**

| **Level** | **Keterangan** |
| --- | --- |
| Level 0.0 | Halaman Utama, halaman yang diakses oleh mahasiswa pertama kali saat membuka aplikasi. |
| Level 1.0 | Menu Tutorial, menampilkan cara bermain *game.* |
| Level 2.0 | Menu Start, awal dimulai permainan. Terdapat sub-menu Kesulitandan Kategorikal*.* |
| Level 2.1 | Menu Kesulitan*,* terdapat pilihan *easy, normal,* dan *hard.* |
| Level 2.2 | Menu Kategorikal*,* terdapat pilihan And, Or, Nand, Nor, Xor, dan Xnor*.* |

#### 3.3.4.2 Flowchart Aplikasi



**Gambar 3. 9 Flowchart Aplikasi**

Pada gambar 3.9 dijelaskan alur dari Aplikasi *Game* Gerbang Logika. Diawali dengan Tutorial, jika mahasiswa memilih Tutorial maka mahasiswa akan melakukan Tutorial. Jika tidak maka mahasiswa memilih tombol Start untuk memulai permainan. Kemudian memilih salah satu kategori. Setiap kategori terdiri dari **lima modul soal**. Pemilihan modul menggunakan algoritma *Random* dimana pendeklarasian nilai *min* dan *max* diawal yang kemudian dioperasikan dengan**random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min)***.* System akan memilih bilangan acak dari nol sampai bilangan maksimal berdasarkan **(max-min+1)+min** terlebih dahulu. Setelah didapat kemudian bilangan tersebut dibulatkan yang kemudian dimasukkan kedalam variabel **random\_int**. Hasil tersebut akan dicocokan dengan angka pada modul soal untuk dipanggil.Setelah itu maka mahasiswa dapat bermain *game. Game* berupa soal dengan tingkat kesulitan berdasarkan pilihan sebelumnya. Setelah bermain akan muncul hasil atau *score*.

#### 3.3.4.3 Perancangan Antarmuka / Interface

Berikut ini adalah rancangan antarmuka / *interface* pada aplikasiGerbang Logika.

##### 3.3.4.3.1 Rancangan Tampilan Halaman Utama



**Gambar 3. 10 Rancangan Tampilan Halaman Utama**

Pada halaman utama aplikasi terdapat terdapat tulisan Game Gerbang Logika dengan button *Tutorial* dan *start.*

##### 3.3.4.3.2 Rancangan Tampilan Halaman Pemilihan Kategori

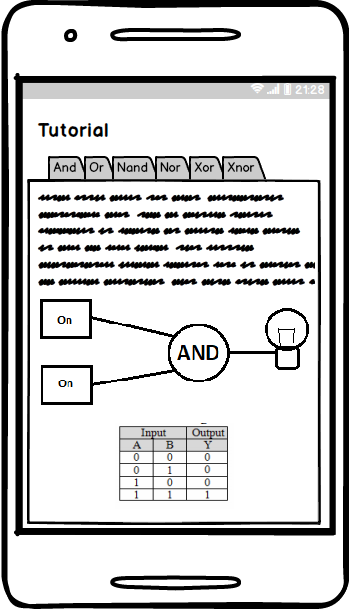
Setelah menekan tombol Start maka akan tampil menu pemilihan kategori. Berikut adalah rancangan halaman tampilan pemilihan kategori pada aplikasi gerbang logika.



**Gambar 3. 11 Rancangan Tampilan Halaman Pemilihan Kategori**

Terdapat pilihan *Random* dan Kategorikal. Jika pilih *Random* maka system akan menampilkan pilihan mudah*,* sedang*,* sulitsecara *drop down.* Jika pilih Kategorikal maka system akan menmapilkan And, Or, Nand, Nor, Xor, dan Xnor secara *drop down*. Untuk dapat bermain, mahasiswa harus memilih salah satu.

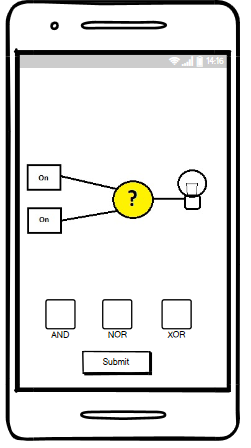
##### 3.3.4.3.3 Rancangan Tampilan Halaman Tutorial



**Gambar 3. 12 Rancangan Tampilan Halaman Tutorial**

Pada rancangan tampilan tutorial ini system akan menampilkan cara bermain *game* Gerbang Logika. Akan ditampilkan table dan rangkaian sederhana (dua *input* dan satu *operator*) berdasarkan tiap *operator.* Mahasiswa dapat memainkan input dengan cara meng-klik input. Jika hasil dari pengoperasian tersebut adalah 1 maka lampu akan hidup jika tidak maka lampu mati.

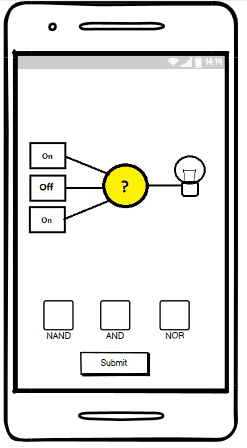
##### 3.3.4.3.4 Rancangan Tampilan Halaman Contoh Soal Mudah



**Gambar 3. 13 Rancangan Tampilan Halaman Contoh Soal Mudah**

Contoh Soal Mudahberupa rangkaian yang terdiri dari dua *input* dan satu *operator*. Mahasiswa harus melengkapi rangkaian tersebut dengan memilih jawaban yang tepat berdasarkan pilihan jawaban yang disediakan. Kemudian tekan tombol *submit*.

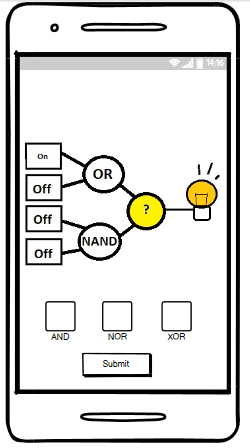
##### 3.3.4.3.5 Rancangan Tampilan Halaman Contoh Soal Sedang



**Gambar 3. 14 Tampilan Halaman Contoh Soal Sedang**

Contoh Soal Sedangberupa rangkaian yang terdiri dari tiga *input* dan satu sampai tiga *operator*. Mahasiswa harus melengkapi rangkaian tersebut dengan memilih jawaban yang tepat berdasarkan pilihan jawaban yang disediakan. Kemudian tekan tombol *submit*.

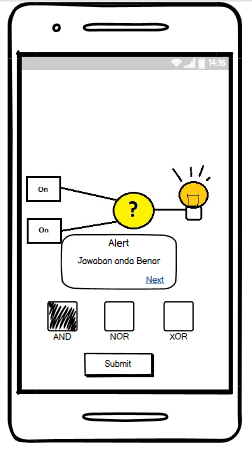
##### 3.3.4.3.6 Rancangan Tampilan Halaman Contoh Soal Sulit



**Gambar 3. 15 Rancangan Tampilan Halaman Contoh Soal Sulit**

Contoh Soal Sulitberupa rangkaian yang terdiri dari empat *input* dan satu sampai tiga *operator*. Mahasiswa harus melengkapi rangkaian tersebut dengan menentukan jawaban yang tepat pada pilihan jawaban tersebut pada layar. Kemudian tekan tombol *submit*.

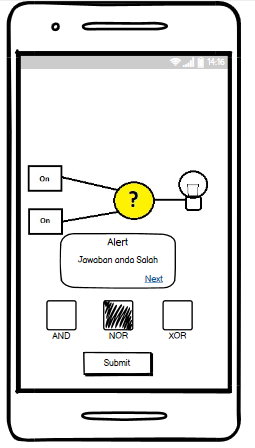
##### 3.3.4.3.7 Rancangan Tampilan Halaman Jawaban Benar



**Gambar 3. 16 Rancangan Tampilan Halaman Jawaban Benar**

Jika jawaban benar maka lampu akan Hidup dan muncul pop-up “Jawaban anda Benar” dan klik next untuk menuju soal berikutnya.

##### 3.3.4.3.8 Rancangan Tampilan Halaman Jawaban Salah



**Gambar 3. 17 Rancangan Tampilan Halaman Jawaban Salah**

Jika jawaban benar maka lampu akan Hidup dan muncul pop-up “Jawaban anda Salah” dan klik next untuk menuju soal berikutnya.

##### 3.3.4.3.9 Rancangan Tampilan Halaman *Score*



**Gambar 3. 18 Rancangan Tampilan Halaman Score**

Jika soal terakhir sudah dikerjakan maka akan tampil halaman *score*. Halaman akan memunculkan *score* berdasarkan jawaban benar dan salah. Terdapat juga *button Continue* dan *Exit*. Jika pilih *Continue* maka akan tampil Halaman Pemilihan Tingkat Kesulitan. Jika pilih *exit* maka system akan mengembalikan ke Halaman Utama.

### 3.3.5 Simulasi

Simulasi yang digunakan berupa rangkaian kelistrikan dasar yang terdiri dari saklar, beban (lampu), massa, dan konduktor. Penerapan simulasi ini bertujuan untuk mempermudah pengguna dalam memainkan permainan. Berikut adalah penggunaan simulasi pada rangkaian yang terdapat pada aplikasi:

* Saklar

Saklar pada rangkaian digambarkan dalam bentuk *button on/off.* Jika *button On* maka saklar tersebut akan mengalirkan arus listrik dan jika *Off* maka tidak ada arus listrik yang mengalir. Berikut adalah bentuk saklar pada aplikasi:



**Gambar 3. 19 Gambar Saklar pada Aplikasi**

* Beban

Beban pada rangkaian dilambangkan menjadi sebuah lampu. Lampu akan Hidup jika diberikan arus listrik dan mati jika tidak ada arus yang mengalir melewatinya.



**Gambar 3. 20 Gambar Lampu Mati dan Hidup pada Aplikasi**

* Konduktor

Konduktor akan mengalirkan arus listrik yang mengalir. Pada aplikasi konduktor digambarkan dengan garis berwarna hitam.



**Gambar 3. 21 Gambar Konduktor pada Aplikasi**

* Massa

Massa atau yang biasa disebut *ground* digambarkan dengan beberapa garis *horizontal* yang berbaris kebawah.



**Gambar 3. 22 Gambar Massa pada Aplikasi**

### 3.3.6 Aturan Permain

Permainan dimulai dari *user* membuka aplikasi, system akan menampilkan halamana utama. Pada halaman utama terdapat pilihan Mulai dan Latihan. Direkomendasikan untuk melakukan latihan terlebih dahulu dengan cara memilih pilihan Latihan. Berikut adalah aturan bermainnya:

#### 3.3.6.1 Latihan (*Tutorial*)

Untuk melakukan Latihan, *user* dapat menekan tombol Latihan. Setelah menekan tombol Latihan system akan menampilkan halaman *tutorial* (latihan). Pada halaman tersebut akan muncul pilihan berdasarkan *operator* logika yang terdapat dalam permainan yaitu logika And, Or, Nand, Nor, Xor, dan Xnor yang jika kita klik akan muncul pengertian, symbol, serta table kebenaran dari jenis *operator* yang kita pilih. Tujuan dari aturan ini dibuat agar *user* memahami aturan-aturan *operator* logika yang akan digunakan pada permainan. Jika *user* sudah memahaminya maka selanjutnya *user* dapat keluar dari *tutorial* dengan menekan tombol *back* pada *device* kemudian pilih Mulai.

#### 3.3.6.2 Mulai

Setelah pilih Mulai maka system akan menampilkan halaman kategori. Didalam halaman kategori terdapat dua kategori penggolongan yaitu *difficulty* (berdasarkan jumlah operator yang digunakan) dengan tombol Mulai-nya dan kategorikal (berdasarkan jenis *operator* yang digunakan) dengan tombol Mulai-nya sendiri. Untuk *difficulty* terdapat pilihan mudah yang tiap soalnya terdiri dari satu *operator* dengan dua *input,* sedangyang tiap soalnya terdiri satu sampai tiga *operator* dengan tiga *input,* serta sulityang tiap soalnya terdiri dari satu sampai tiga *operator* dengan empat *input*. Dan untuk kategorikal terdiri dari And, Or, Nand, Nor, Xor, Xnor dengan komposisi soal terdiri dari dua soal mudah*,* empat soal sedang*,* dan empat soal sulit*.*

Untuk setiap kategori terdiri dari lima *modul* yang dimana tiap *modul* terdiri dari sepuluh soal. Soal yang diambil berasal dari ***bank* soal**. Pada *bank* soal terdapat total **sembilan puluh soal** yang terdiri dari delapan belas soal mudah (And, Or, Nand, Nor, Xor, dan Xnor masing-masing tiga soal), tiga puluh enam soal sedang(And, Or, Nand, Nor, Xor, dan Xnor masing-masing enam soal), dan tiga puluh enam soal sulit(And, Or, Nand, Nor, Xor, dan Xnor masing-masing enam soal). Dan pemanggilan acak dilakukan pada pemanggilan *modul-*nya. Pada saat modul dipanggil maka system akan menampilkan halaman soal.

Pada halaman soal terdiri dari nomor urut soal (satu sampai sepuluh), gambar rangkaian, tiga pilihan jawaban beserta kunci jawabannya dan tombo *Submit*. Untuk tiap soal sudah *include* dengan kunci jawabannya masing-masing. Jika *user* meng-*input* jawaban maka saat di *submit* system akan mencocokkan dengan kunci jawabannya. Jika jawaban salah, maka lampu pada rangkaian akan tetap mati kemudian system akan menampilkan *pop-up* yang menandakan bahwa jawaban yang dipilih adalah salah dan juga memberi tahu jawaban yang benar. Jika jawaban benar, maka lampu pada rangkaian akan hidup kemudian system akan menampilkan *pop-up* yang menandakan bahwa jawaban yang dipilih adalah benar. Semua jawaban yang benar akan diberi nilai sepuluh serta jawaban yang salah akan diberi nilai nol.

Jika *user* sudah menjawab sepuluh soal maka system akan menampilkan halaman *score.* Pada halaman score terdapat score yang banyaknya berdasarkan jumlah jawaban yang benar dengan nilai maksimal adalah seratus. Selain itu juga terdapat informasi tentang kategori yang dipilih serta tombol *continue* dan *exit*. Tombol *continue* berfungsi untuk kembali ke halaman kategori jika *user* ingin memainkan permainan lagi dan tombol *exit* berfungsi untuk menyudahi permainan menuju ke halaman utama setelah itu tekan tombol *back* pada *device* maka akan muncul *alert* berupa peringatan untuk keluar aplikasi. Kemudian tekan *yes* jika iya dan tekan *no* jika ingin membatalkan.

### 3.3.7 Pemanggilan Acak Pada Model Soal

Pemanggilan acak pada model soal menggunakan algoritma random yaitu **random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min)**. Pertama tentukan nilai min sama dengan satu dan max sesuai dengan jumlah model yang akan kita acak. Setelah itu algoritma ini akan menggunakan nilai min max tersebut untuk dioperasikan. Setelah min max dioperasikan kemudian menghasilkan angka yang menjadi batas pengacakan angka. **Math.random** akan mengambil nilai acak dari satu sampai batas angka yang tadi diperoleh. Setelah didapat, selanjutnya akan di bulatkan oleh **math.floor** dan kemudian di integerkan. Hasil dari pengacakan tadi akan dimasukan dan menjadi nilai dari **random\_int**.

# BAB IV

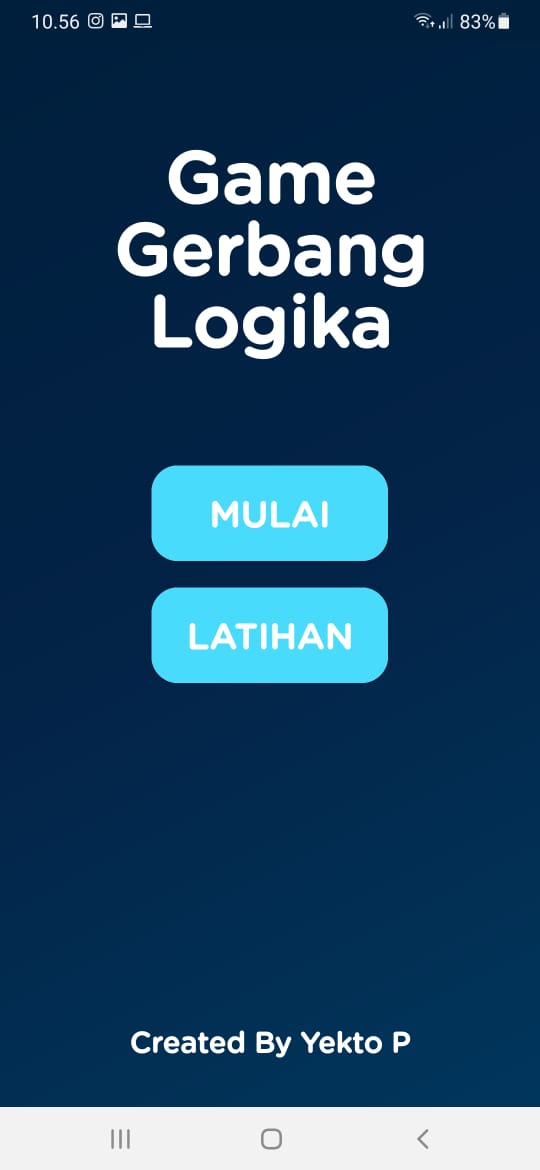
**HASIL DAN PEMBAHASAN**

## 4.1 Tampilan Aplikasi Gerbang Logika

Berikut ini adalah tampilan dan pengkodean hasil dari aplikasi Gerbang Logika:

### 4.1.1 Tampilah Halaman Utama

Berikut adalah tampilan halaman utama pada aplikasi *game* Gerbang Logika pada gambar 4.1 sebagai berikut:



**Gambar 4. 1 Tampilan Halaman Utama**

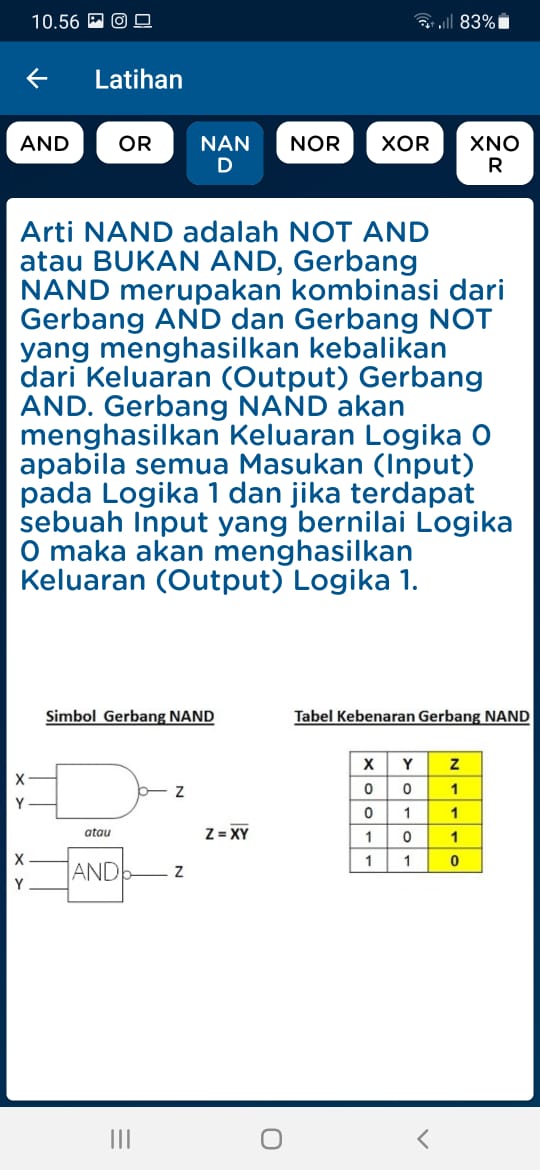
Adapun kode program pada halaman utama sebagai berikut:

**Tabel 4. 1 Kode Tampilan Halaman Utama**

|  |
| --- |
| package com.destiny.gamegerbanglogika;  import androidx.appcompat.app.AppCompatActivity;  import android.content.Intent;  import android.os.Bundle;  import android.view.View;  import android.widget.ImageView;  import android.widget.LinearLayout;  import android.widget.TextView;  import com.destiny.gamegerbanglogika.Method.Method;  import com.destiny.gamegerbanglogika.Mulai.StartActivity;  import java.text.DateFormat;  import java.text.SimpleDateFormat;  import java.util.Calendar;  import java.util.Date;  public class HomeActivity extends AppCompatActivity {  LinearLayout Latihan,Mulai;  ImageView ivHeader;  TextView tvHeader,tvTgl;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_home);  ivHeader = findViewById(R.id.ivHeader);  tvHeader = findViewById(R.id.tvHeader);  Latihan = findViewById(R.id.linearLatihan);  Mulai = findViewById(R.id.linearMulai);  Latihan.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Intent intent =new Intent(HomeActivity.this,LatihanActivity.class);  startActivity(intent);  }  });  Mulai.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Intent intent =new Intent(HomeActivity.this, StartActivity.class);  startActivity(intent);  }  });  }  } |

### 4.1.2 Tampilan Halaman Tutorial

Berikut adalah tampilan halaman *tutorial* pada aplikasi *game* Gerbang Logika.



**Gambar 4. 2 Tampilan Halaman Tutorial**

Adapun kode program pada halaman tutorial sebagai berikut

**Tabel 4. 2 Kode Tampilan Halaman Tutorial**

|  |
| --- |
| package com.destiny.gamegerbanglogika;  import androidx.appcompat.app.AppCompatActivity;  import androidx.fragment.app.Fragment;  import androidx.fragment.app.FragmentManager;  import androidx.fragment.app.FragmentTransaction;  import android.graphics.Color;  import android.os.Bundle;  import android.view.View;  import android.widget.LinearLayout;  import android.widget.TextView;  import com.destiny.gamegerbanglogika.Tabs.AndFragment;  import com.destiny.gamegerbanglogika.Tabs.NandFragment;  import com.destiny.gamegerbanglogika.Tabs.NorFragment;  import com.destiny.gamegerbanglogika.Tabs.OrFragment;  import com.destiny.gamegerbanglogika.Tabs.XnorFragment;  import com.destiny.gamegerbanglogika.Tabs.XorFragment;  public class LatihanActivity extends AppCompatActivity {  LinearLayout LAND,LOR,LNAND,LNOR,LXOR,LXNOR;  TextView AND,OR,NAND,NOR,XOR,XNOR;  Fragment fragment;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_latihan);  Declaration();  AND();  OnCLick();  }  private void OnCLick(){  LAND.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  AND();  }  });  LOR.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  OR();  }  });  LNAND.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  NAND();  }  });  LNOR.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  NOR();  }  });  LXOR.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  XOR();  }  });  LXNOR.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  XNOR();  }  });  }  private void Declaration(){  LAND = findViewById(R.id.linearAnd);  LOR = findViewById(R.id.linearOr);  LNAND = findViewById(R.id.linearNand);  LNOR = findViewById(R.id.linearNor);  LXOR = findViewById(R.id.linearXor);  LXNOR = findViewById(R.id.linearXnor);  AND = findViewById(R.id.tvAnd);  OR = findViewById(R.id.tvOr);  AND = findViewById(R.id.tvAnd);  NAND = findViewById(R.id.tvNand);  NOR = findViewById(R.id.tvNor);  XOR = findViewById(R.id.tvXor);  XNOR = findViewById(R.id.tvXnor);  }  private void ChangeFragment(Fragment fragment){  if(fragment != null){  FragmentManager fragmentManager = getSupportFragmentManager();  FragmentTransaction ft = fragmentManager.beginTransaction();  ft.replace(R.id.Container,fragment);  ft.commit();  }  }  private void Default(){  LAND.setBackgroundResource(R.color.colorWhite);  LOR.setBackgroundResource(R.color.colorWhite);  LNAND.setBackgroundResource(R.color.colorWhite);  LNOR.setBackgroundResource(R.color.colorWhite);  LXOR.setBackgroundResource(R.color.colorWhite);  LXNOR.setBackgroundResource(R.color.colorWhite);  AND.setTextColor(Color.rgb(0,0,0));  OR.setTextColor(Color.rgb(0,0,0));  NAND.setTextColor(Color.rgb(0,0,0));  NOR.setTextColor(Color.rgb(0,0,0));  XOR.setTextColor(Color.rgb(0,0,0));  XNOR.setTextColor(Color.rgb(0,0,0));  }  private void AND(){  Default();  LAND.setBackgroundResource(R.color.colorPrimary);  AND.setTextColor(Color.rgb(255,255,255));  fragment = new AndFragment();  ChangeFragment(fragment);  }  private void OR(){  Default();  LOR.setBackgroundResource(R.color.colorPrimary);  OR.setTextColor(Color.rgb(255,255,255));  fragment = new OrFragment();  ChangeFragment(fragment);  }  private void NAND(){  Default();  LNAND.setBackgroundResource(R.color.colorPrimary);  NAND.setTextColor(Color.rgb(255,255,255));  fragment = new NandFragment();  ChangeFragment(fragment);  }  private void NOR(){  Default();  LNOR.setBackgroundResource(R.color.colorPrimary);  NOR.setTextColor(Color.rgb(255,255,255));  fragment = new NorFragment();  ChangeFragment(fragment);  }  private void XOR(){  Default();  LXOR.setBackgroundResource(R.color.colorPrimary);  XOR.setTextColor(Color.rgb(255,255,255));  fragment = new XorFragment();  ChangeFragment(fragment);  }  private void XNOR(){  Default();  LXNOR.setBackgroundResource(R.color.colorPrimary);  XNOR.setTextColor(Color.rgb(255,255,255));  fragment = new XnorFragment();  ChangeFragment(fragment);  }  } |

### 4.1.3 Tampilan Halaman Kategori

Berikut adalah tampilan halaman kategori pada aplikasi *game* Gerbang Logika.



**Gambar 4. 3 Tampilan Halaman Kategori**

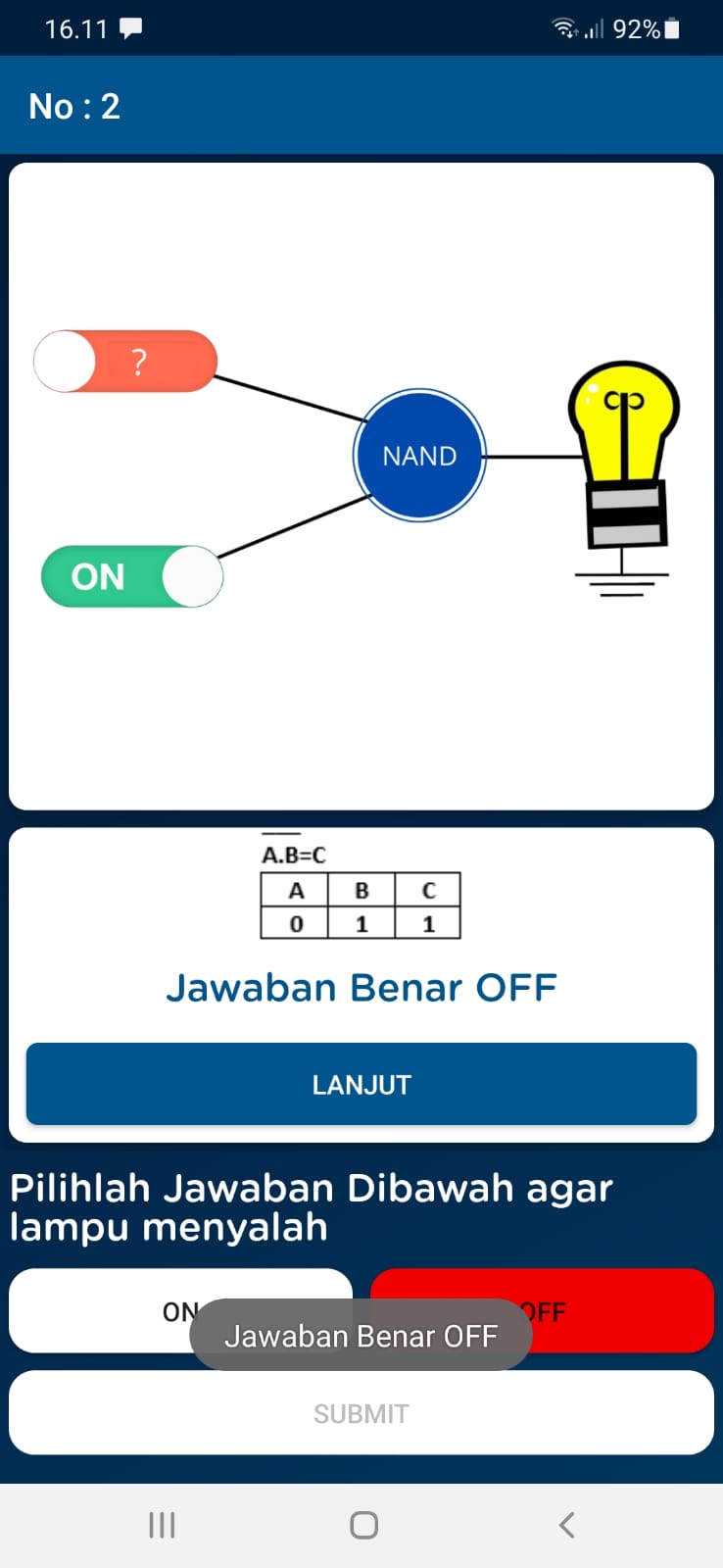
Adapun kode program pada halaman kategori sebagai berikut.

**Tabel 4. 3 Kode Tampilan Halaman Kategori**

|  |
| --- |
| package com.destiny.gamegerbanglogika.Mulai;  import androidx.appcompat.app.AppCompatActivity;  import android.content.Intent;  import android.os.Bundle;  import android.view.View;  import android.widget.ArrayAdapter;  import android.widget.LinearLayout;  import android.widget.Spinner;  import android.widget.Toast;  import com.destiny.gamegerbanglogika.R;  public class StartActivity extends AppCompatActivity {  Spinner dificulty,kategori;  LinearLayout LDificulty,LKategori;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_start);  dificulty = findViewById(R.id.spinnerDificulty);  kategori = findViewById(R.id.spinnerKategori);  LDificulty = findViewById(R.id.linearDificulty);  LKategori = findViewById(R.id.linearCategory);  ArrayAdapter adapter = ArrayAdapter.createFromResource(this,  R.array.dificulty, R.layout.spinner\_item);  ArrayAdapter adapter2 = ArrayAdapter.createFromResource(this,  R.array.kategorikal, R.layout.spinner\_item);  dificulty.setAdapter(adapter);  kategori.setAdapter(adapter2);  LDificulty.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if (dificulty.getSelectedItem().toString().equals("Easy")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","Easy");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else if (dificulty.getSelectedItem().toString().equals("Normal")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","Normal");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else if (dificulty.getSelectedItem().toString().equals("Hard")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","Hard");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else{  Toast.makeText(StartActivity.this, "Harap Pilih Dificulty", Toast.LENGTH\_SHORT).show();  }  }  });  LKategori.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if (kategori.getSelectedItem().toString().equals("AND")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","AND");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else if (kategori.getSelectedItem().toString().equals("OR")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","OR");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else if (kategori.getSelectedItem().toString().equals("NAND")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","NAND");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else if (kategori.getSelectedItem().toString().equals("NOR")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","NOR");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else if (kategori.getSelectedItem().toString().equals("XOR")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","XOR");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else if (kategori.getSelectedItem().toString().equals("XNOR")){  Intent goInput = new Intent(StartActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(0));  goInput.putExtra("SCORE",String.valueOf(0));  goInput.putExtra("KATEGORI","XNOR");  int min = 1;  int max = 5;  int random\_int = (int)Math.floor(Math.random()\*(max-min+1)+min);  goInput.putExtra("RAND",String.valueOf(random\_int));  startActivities(new Intent[]{goInput});  }else{  Toast.makeText(StartActivity.this, "Harap Pilih Kategori", Toast.LENGTH\_SHORT).show();  }  }  });  }  } |

### 4.1.4 Tampilan Halaman Soal

Berikut adalah tampilan halaman soal pada aplikasi *game* Gerbang Logika.



**Gambar 4. 4 Tampilan Halaman Soal**

Setelah jawaban di input maka akan muncul *true table* dari rangkaian untuk membuat lampu hidup serta pemberitahuan bahwa jawaban yang dipilih adalah Benar atau Salah.

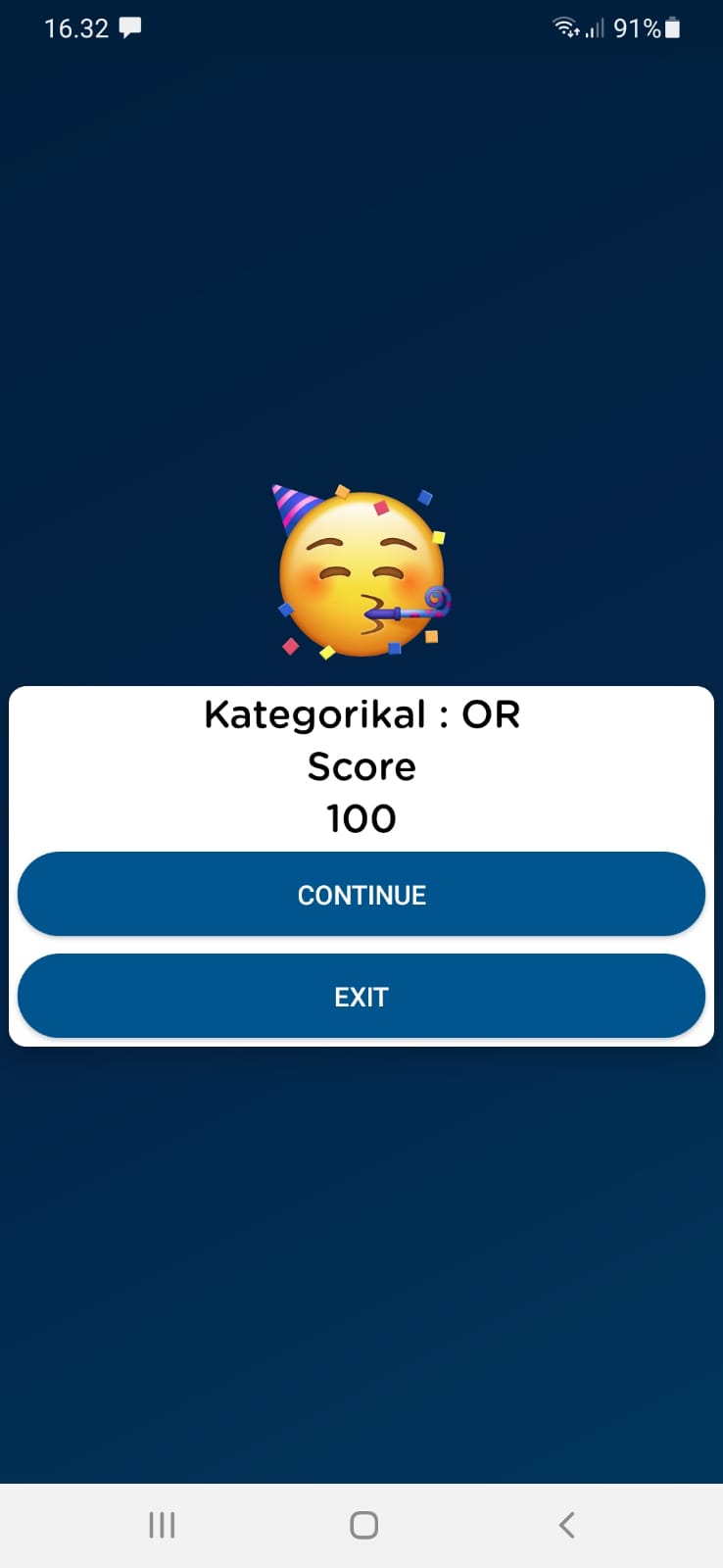
Adapun kode program program pada halaman soal sebagai berikut.

**Tabel 4. 4 Kode Tampilan Halaman Soal**

|  |
| --- |
| package com.destiny.gamegerbanglogika.Mulai;  import androidx.appcompat.app.AppCompatActivity;  import android.app.AlertDialog;  import android.content.DialogInterface;  import android.content.Intent;  import android.os.Bundle;  import android.os.Handler;  import android.view.View;  import android.widget.Button;  import android.widget.ImageView;  import android.widget.Toast;  import com.destiny.gamegerbanglogika.Model.DataModel;  import com.destiny.gamegerbanglogika.Model.Dificulty.Easy.EasyModel;  import com.destiny.gamegerbanglogika.Model.Dificulty.Easy.EasyModel1;  import com.destiny.gamegerbanglogika.Model.Dificulty.Easy.EasyModel2;  import com.destiny.gamegerbanglogika.Model.Dificulty.Easy.EasyModel3;  import com.destiny.gamegerbanglogika.Model.Dificulty.Easy.EasyModel4;  import com.destiny.gamegerbanglogika.Model.Dificulty.Easy.EasyModel5;  import com.destiny.gamegerbanglogika.Model.Dificulty.Hard.HardModel1;  import com.destiny.gamegerbanglogika.Model.Dificulty.Hard.HardModel2;  import com.destiny.gamegerbanglogika.Model.Dificulty.Hard.HardModel3;  import com.destiny.gamegerbanglogika.Model.Dificulty.Hard.HardModel4;  import com.destiny.gamegerbanglogika.Model.Dificulty.Hard.HardModel5;  import com.destiny.gamegerbanglogika.Model.Dificulty.Medium.NormalModel1;  import com.destiny.gamegerbanglogika.Model.Dificulty.Medium.NormalModel2;  import com.destiny.gamegerbanglogika.Model.Dificulty.Medium.NormalModel3;  import com.destiny.gamegerbanglogika.Model.Dificulty.Medium.NormalModel4;  import com.destiny.gamegerbanglogika.Model.Dificulty.Medium.NormalModel5;  import com.destiny.gamegerbanglogika.Model.Kategorikal.AND.AndModel1;  import com.destiny.gamegerbanglogika.Model.Kategorikal.AND.AndModel2;  import com.destiny.gamegerbanglogika.Model.Kategorikal.AND.AndModel3;  import com.destiny.gamegerbanglogika.Model.Kategorikal.AND.AndModel4;  import com.destiny.gamegerbanglogika.Model.Kategorikal.AND.AndModel5;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NAND.NandModel1;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NAND.NandModel2;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NAND.NandModel3;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NAND.NandModel4;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NAND.NandModel5;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NOR.NorModel1;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NOR.NorModel2;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NOR.NorModel3;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NOR.NorModel4;  import com.destiny.gamegerbanglogika.Model.Kategorikal.NOR.NorModel5;  import com.destiny.gamegerbanglogika.Model.Kategorikal.OR.OrModel1;  import com.destiny.gamegerbanglogika.Model.Kategorikal.OR.OrModel2;  import com.destiny.gamegerbanglogika.Model.Kategorikal.OR.OrModel3;  import com.destiny.gamegerbanglogika.Model.Kategorikal.OR.OrModel4;  import com.destiny.gamegerbanglogika.Model.Kategorikal.OR.OrModel5;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XNOR.XnorModel1;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XNOR.XnorModel2;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XNOR.XnorModel3;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XNOR.XnorModel4;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XNOR.XnorModel5;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XOR.XorModel1;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XOR.XorModel2;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XOR.XorModel3;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XOR.XorModel4;  import com.destiny.gamegerbanglogika.Model.Kategorikal.XOR.XorModel5;  import com.destiny.gamegerbanglogika.R;  import java.util.ArrayList;  public class SoalActivity extends AppCompatActivity {  Button Jawaban1,Jawaban2,Jawaban3,Submit;  ImageView Gambar;  private ArrayList<DataModel> pList = new ArrayList<>();  String Jawaban = "0";  String Kategori,Random;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_soal);  Intent data = getIntent();  String No = data.getStringExtra("NO");  final String Score = data.getStringExtra("SCORE");  final String Kategori = data.getStringExtra("KATEGORI");  final String Random = data.getStringExtra("RAND");  final int no = Integer.parseInt(No);  Jawaban1 = findViewById(R.id.btnJawaban1);  Jawaban2 = findViewById(R.id.btnJawaban2);  Jawaban3 = findViewById(R.id.btnJawaban3);  Submit = findViewById(R.id.btnSubmit);  Gambar = findViewById(R.id.ivGambar);  if (Kategori.equals("Easy")){  if (Random.equals("1")){  pList.addAll(EasyModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(EasyModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(EasyModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(EasyModel4.getListData());  }else{  pList.addAll(EasyModel5.getListData());  }  }else if (Kategori.equals("Normal")){  if (Random.equals("1")){  pList.addAll(NormalModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(NormalModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(NormalModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(NormalModel4.getListData());  }else{  pList.addAll(NormalModel5.getListData());  }  }else if (Kategori.equals("Hard")){  if (Random.equals("1")){  pList.addAll(HardModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(HardModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(HardModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(HardModel4.getListData());  }else{  pList.addAll(HardModel5.getListData());  }  }else if (Kategori.equals("AND")){  if (Random.equals("1")){  pList.addAll(AndModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(AndModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(AndModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(AndModel4.getListData());  }else{  pList.addAll(AndModel5.getListData());  }  }else if (Kategori.equals("NAND")){  if (Random.equals("1")){  pList.addAll(NandModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(NandModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(NandModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(NandModel4.getListData());  }else{  pList.addAll(NandModel5.getListData());  }  }else if (Kategori.equals("NOR")){  if (Random.equals("1")){  pList.addAll(NorModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(NorModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(NorModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(NorModel4.getListData());  }else{  pList.addAll(NorModel5.getListData());  }  }else if (Kategori.equals("OR")){  if (Random.equals("1")){  pList.addAll(OrModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(OrModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(OrModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(OrModel4.getListData());  }else{  pList.addAll(OrModel5.getListData());  }  }else if (Kategori.equals("XNOR")){  if (Random.equals("1")){  pList.addAll(XnorModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(XnorModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(XnorModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(XnorModel4.getListData());  }else{  pList.addAll(XnorModel5.getListData());  }  }else if (Kategori.equals("XOR")){  if (Random.equals("1")){  pList.addAll(XorModel1.getListData());  }else if (Random.equals("2")){  pList.addAll(XorModel2.getListData());  }else if (Random.equals("3")){  pList.addAll(XorModel3.getListData());  }else if (Random.equals("4")){  pList.addAll(XorModel4.getListData());  }else{  pList.addAll(XorModel5.getListData());  }  }  getSupportActionBar().setTitle("No : "+pList.get(no).getNo());  Jawaban1.setText(pList.get(no).getJawaban1());  Jawaban2.setText(pList.get(no).getJawaban2());  Jawaban3.setText(pList.get(no).getJawaban3());  Gambar.setImageResource(Integer.parseInt(pList.get(no).getGambar()));  final Handler handler = new Handler();  Jawaban1.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  one(pList.get(no).getJawaban1());  }  });  Jawaban2.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  two(pList.get(no).getJawaban2());  }  });  Jawaban3.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  three(pList.get(no).getJawaban3());  }  });  Submit.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if (Jawaban.equals(pList.get(no).getJawaban())){  Gambar.setImageResource(Integer.parseInt(pList.get(no).getGambar2()));  Toast.makeText(SoalActivity.this, "Jawaban Benar "+Jawaban, Toast.LENGTH\_SHORT).show();  handler.postDelayed(new Runnable() {  public void run() {  if (no >= pList.size()-1){  Intent goInput = new Intent(SoalActivity.this, ScoreActivity.class);  goInput.putExtra("SCORE",String.valueOf(Integer.parseInt(Score)+10));  goInput.putExtra("KATEGORI",Kategori);  startActivity(goInput);  }else{  Intent goInput = new Intent(SoalActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(no+1));  goInput.putExtra("SCORE",String.valueOf(Integer.parseInt(Score)+10));  goInput.putExtra("KATEGORI",Kategori);  goInput.putExtra("RAND",Random);  startActivities(new Intent[]{goInput});  }  }  }, 3000); //3000 L  }else{  Toast.makeText(SoalActivity.this, "Jawaban Salah "+Jawaban+" Yang Benar adalah "+pList.get(no).getJawaban(), Toast.LENGTH\_SHORT).show();  Gambar.setImageResource(Integer.parseInt(pList.get(no).getGambar()));  handler.postDelayed(new Runnable() {  public void run() {  if (no >= pList.size()-1){  Intent goInput = new Intent(SoalActivity.this, ScoreActivity.class);  goInput.putExtra("SCORE",String.valueOf(Integer.parseInt(Score)));  goInput.putExtra("KATEGORI",Kategori);  startActivity(goInput);  }else{  Intent goInput = new Intent(SoalActivity.this, SoalActivity.class);  goInput.putExtra("NO",String.valueOf(no+1));  goInput.putExtra("SCORE",String.valueOf(Integer.parseInt(Score)));  goInput.putExtra("KATEGORI",Kategori);  goInput.putExtra("RAND",Random);  startActivities(new Intent[]{goInput});  }  }  }, 3000); //3000 L  }  }  });  }  private void Default(){  Jawaban1.setBackgroundResource(R.drawable.button\_home\_white);  Jawaban2.setBackgroundResource(R.drawable.button\_home\_white);  Jawaban3.setBackgroundResource(R.drawable.button\_home\_white);  }  private void one(String Jaw){  Default();  Jawaban1.setBackgroundResource(R.drawable.button\_red\_rounded);  Jawaban = Jaw;  }  private void two(String Jaw){  Default();  Jawaban2.setBackgroundResource(R.drawable.button\_red\_rounded);  Jawaban = Jaw;  }  private void three(String Jaw){  Default();  Jawaban3.setBackgroundResource(R.drawable.button\_red\_rounded);  Jawaban = Jaw;  }  @Override  public void onBackPressed() {  AlertDialog.Builder builder = new AlertDialog.Builder(SoalActivity.this);  // Set a title for alert dialog  builder.setTitle("Pemberitahuan");  // Ask the final question  builder.setMessage("Jika Kembali Sekarang Jawaban anda akan Ter-Reset");  // Set the alert dialog yes button click listener  builder.setPositiveButton("Iya", new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  // Do something when user clicked the Yes button  Intent intent = new Intent(SoalActivity.this,StartActivity.class);  startActivity(intent);  finishAffinity();  }  });  // Set the alert dialog no button click listener  builder.setNegativeButton("Tidak", new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  // Do something when No button clicked  }  });  AlertDialog dialog = builder.create();  // Display the alert dialog on interface  dialog.show();  }  } |

### 4.1.5 Tampilan Halaman *Score*

Berikut adalah tampilan halaman *score* pada aplikasi *game* Gerbang Logika.



**Gambar 4. 5Tampilan Halaman Score**

Halaman *score* menampilkan jenis kategori yang dipilih dan *score* yang diperoleh. Terdapat *button Contiue* yang saat di klik akan kembali menuju ke halaman kategori serta *button Exit* yang jika di klik akan kembali pada halaman utama.

Adapun kode program pada halaman *score* sebagai berikut.

**Tabel 4. 5 Kode Tampilan Halaman Score**

|  |
| --- |
| package com.destiny.gamegerbanglogika.Mulai;  import androidx.appcompat.app.AppCompatActivity;  import android.content.Intent;  import android.os.Bundle;  import android.view.View;  import android.widget.Button;  import android.widget.TextView;  import com.destiny.gamegerbanglogika.HomeActivity;  import com.destiny.gamegerbanglogika.R;  public class ScoreActivity extends AppCompatActivity {  TextView Dificulty,Score;  Button Continue,Stop;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_score);  Dificulty = findViewById(R.id.tvDificulty);  Score = findViewById(R.id.tvScore);  Continue = findViewById(R.id.btnContinue);  Stop = findViewById(R.id.btnExit);  Intent data = getIntent();  final String score = data.getStringExtra("SCORE");  final String Kategori = data.getStringExtra("KATEGORI");  if (Kategori.equals("Easy")){  Dificulty.setText("Kesulitan : "+Kategori);  }else if (Kategori.equals("Normal")){  Dificulty.setText("Kesulitan : "+Kategori);  }else if (Kategori.equals("Hard")){  Dificulty.setText("Kesulitan : "+Kategori);  }else if (Kategori.equals("AND")){  Dificulty.setText("Kategorikal : "+Kategori);  }else if (Kategori.equals("NAND")){  Dificulty.setText("Kategorikal : "+Kategori);  }else if (Kategori.equals("NOR")){  Dificulty.setText("Kategorikal : "+Kategori);  }else if (Kategori.equals("OR")){  Dificulty.setText("Kategorikal : "+Kategori);  }else if (Kategori.equals("XNOR")){  Dificulty.setText("Kategorikal : "+Kategori);  }else if (Kategori.equals("XOR")){  Dificulty.setText("Kategorikal : "+Kategori);  }  Score.setText(score);  Continue.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Intent intent = new Intent(ScoreActivity.this,StartActivity.class);  startActivity(intent);  }  });  Stop.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Intent intent = new Intent(ScoreActivity.this, HomeActivity.class);  startActivity(intent);  }  });  }  @Override  public void onBackPressed() {  }  } |

## 4.2 Pengujian Aplikasi

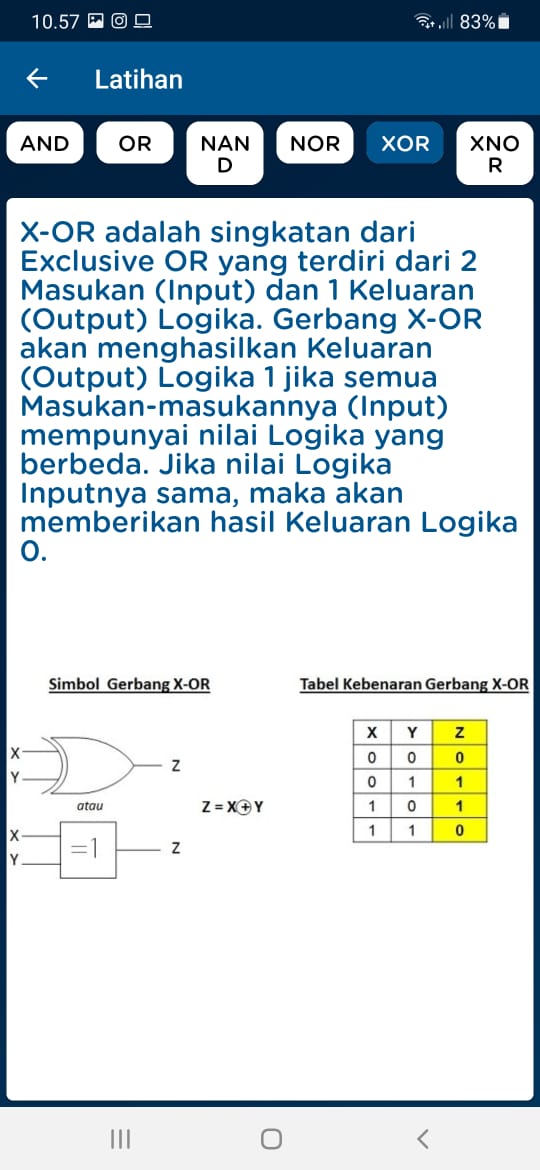
Penggunaan metode *black-box* dalam pengujian diperoleh hasil sebagai berikut:

### 4.2.1 Pengujian Prosedur Latihan (*Tutorial*)

Pada halaman utama terdapat tombol Latihan untuk melakukan latihan yang jika di klik akan menampilkan penjelasan dari *operator* yang digunakan. Berikut adalah table hasi ujinya.

**Tabel 4. 6 Pengujian Prosedur Latihan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pengujian dan Hasil Uji (data benar)** | | | | |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button* Latihan | *OnClick* | Menampilkan table informasi mengenai operator logika (AND, OR, NAND, NOR, XOR, XNOR) | Sesuai yang diharapkan | Berhasil |



**Gambar 4. 6 Pengujian Prosedur Latihan**

Gambar 4.6 adalah hasil uji coba prosedur tutorial. Jika berhasil maka system akan menampilkan informasi tentang penggunaan operator gerbang logika lengkap dengan pengertian, symbol, dan table kebenarannya.

### 4.2.2 Pengujian Tombol Mulai

Pada halaman utama terdapat tombol Mulai. Jika di klik akan menuju halaman kategori untuk memilih kategori yang akan dipilih. Berikut adalah table hasil uji.

**Tabel 4. 7 Pengujian Tombol Mulai**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pengujian dan Hasil Uji (data benar)** | | | | |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button* Mulai | *OnClick* | Menampilkan halaman kategori. | Sesuai yang diharapkan | Berhasil |



**Gambar 4. 7 Pengujian Tombol Mulai**

Gambar x adalah hasil uji coba tombol Mulai. Jika ditekan maka system akan menampilkan pilihan kategori soal.

### 4.2.3 Pengujian *Spinner*

Pada halaman kategoridilakukan pengujian *spinner* untuk memastikan bahwa *user* dapat memilih kategori soal yang akan dipilih. Terdapat dua *spinner* yaitu ***spinner difficulty***yang berisi mudah*,* sedang*,* sulitdan ***spinner* kategorikal** yang berisi and, or, nand, nor, xor, dan xnor. Dapat kita lihat table hasil uji pada table 4.8.

**Tabel 4. 8 Pengujian Spinner**

| **Pengujian dan Hasil Uji (data benar)** | | | | |
| --- | --- | --- | --- | --- |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Spinner Dificulty* | *OnClick* | *Spinner* menampilkan pilihan mudah*,* sedang*,* dan sulit. | Sesuai yang diharapkan | Berhasil |
| *Spinner* Kategorikal | *OnClick* | *Spinner* menampilkan pilihan and, or, nand, nor, xor, dan xnor | Sesuai yang diharapkan | Berhasil |



**Gambar 4. 8 Pengujian Spinner Dificulty**



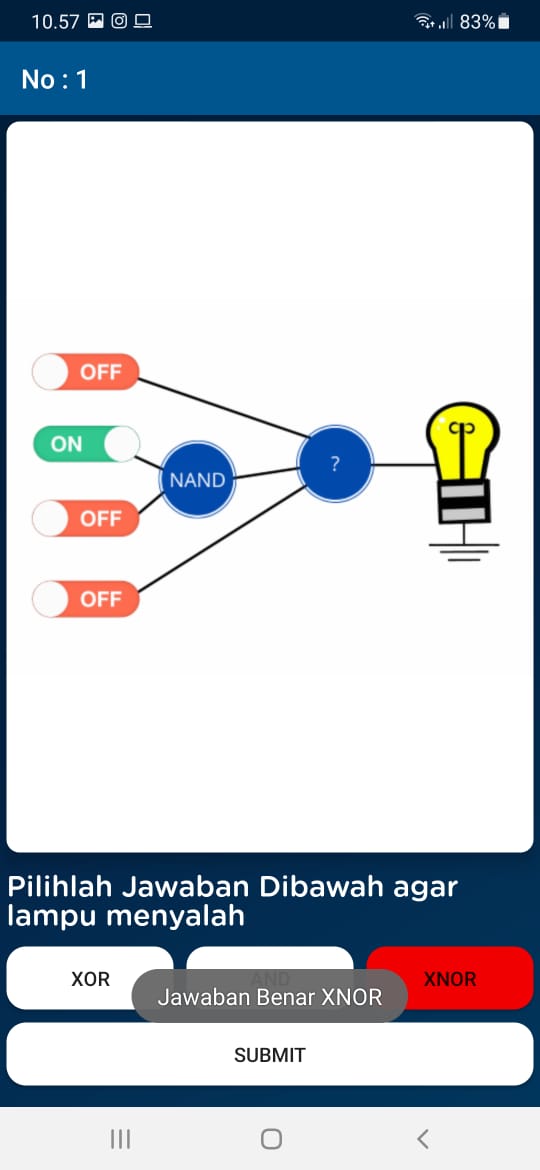
**Gambar 4. 9 Pengujian Spinner Kategorikal**

### 4.2.4 Pengujian Tombol Mulai pada *Spinner*

Setelah *user* memilih kategori pada *spinner* selanjutnya adalah tekan tombol Mulai untuk menuju soal berdasarkan kategori yang dipilih. Berikut adalah table hasil uji.

**Tabel 4. 9 Pengujian Tombol Mulai pada Spinner (Benar)**

| **Pengujian dan Hasil Uji (data benar)** | | | | |
| --- | --- | --- | --- | --- |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button* Mulai pada *Spinner* | *user* memilih kategori pada *spinner* | Menampilkan soal berdasarkan kategori yang dipilih | Sesuai yang diharapkan | Berhasil |



**Gambar 4. 10 Pengujian Tombol Mulai pada Spinner (Benar)**

Gambar 4.10 merupakan hasil uji coba dari salah satu tombol Mulai pada *spinner*. Jika ditekan maka system akan menampilkan soal berdasarkan jenis kategori yang dipilih.

**Tabel 4. 10 Pengujian Tombol Mulai pada Spinner (Salah)**

| **Pengujian dan Hasil Uji (data salah)** | | | | |
| --- | --- | --- | --- | --- |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button* Mulai pada *Spinner* | *user* memilih kategori pada *spinner* | Menampilkan *alert* bahwa harus memilih kategori terlebih dahulu | Sesuai yang diharapkan | Berhasil |



**Gambar 4. 11 Pengujian Tombol Mulai pada Spinner (Salah)**

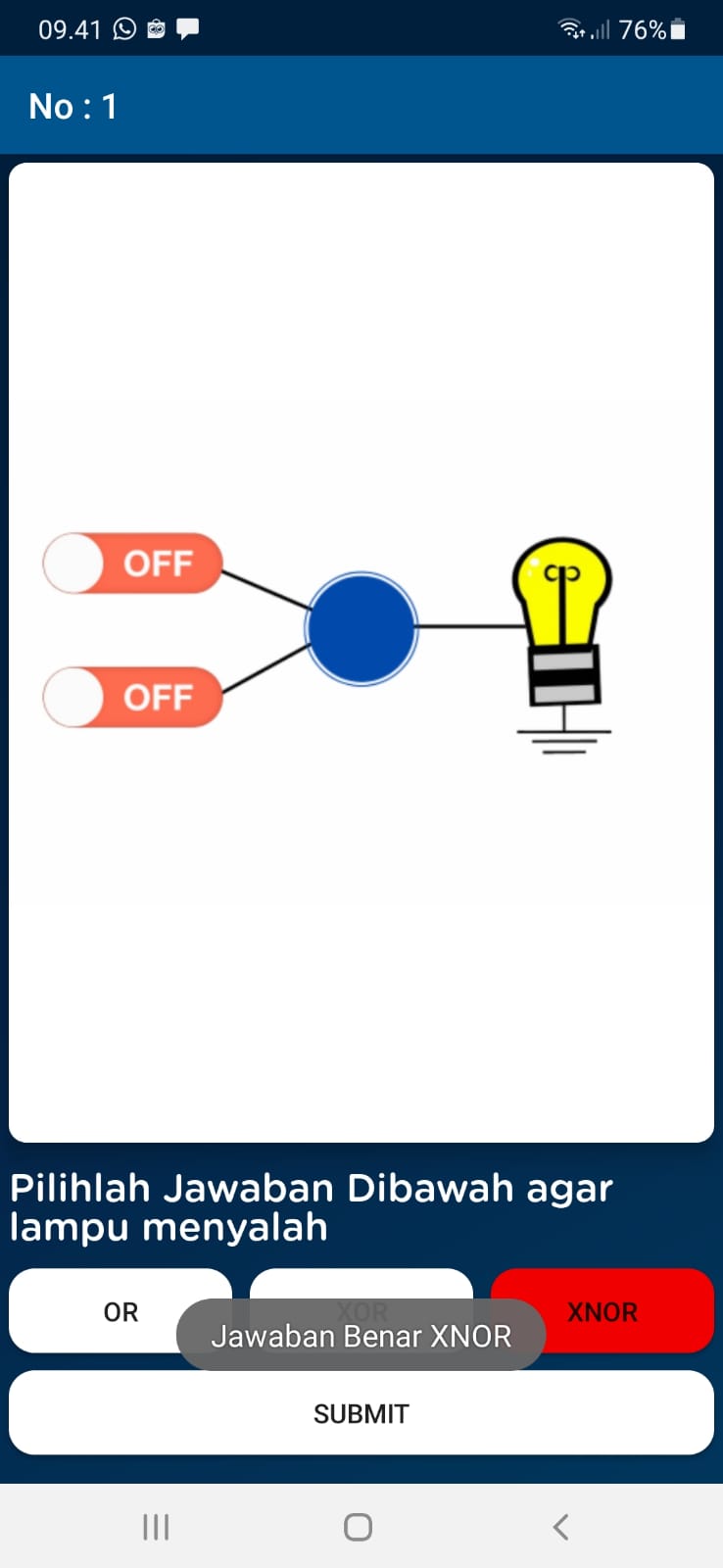
Jika *user* belum memilih jenis kategori maka saat *user* menekan Mulai maka system akan menampilkan sebuah *pop-up* Harap Pilih Kategori.

### 4.2.5 Pengujian Tombol Submit

Tombol *submit* terapat pada halaman soal. Tombol ini berfungsi sebagai klarifikasi terhadap jawaban yang dipilih. Jika jawaban yang dipilih adalah benar maka setelah disubmit lampu akan hidup dan terdapat *pop-up* bahwa jawaban benar. Jika jawaban yang dipilih adalah salah maka setelah disubmit lampu akan tetap dalam keadaan mati dan terdapat *pop-up* bahwa jawaban yang dipilih adalah salah dan kemudian *pop-up* juga memberitahu pilihan jawaban yang benar. Berikut adalah table hasil uji.

**Tabel 4. 11 Pengujian Tombol Submit (Benar)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pengujian dan Hasil Uji (data benar)** | | | | |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button Submit* | Jawaban Benar | Lampu hidup dan muncul *pop-up* bahwa jawaban yang diinput adalah benar | Sesuai yang diharapkan | Berhasil |

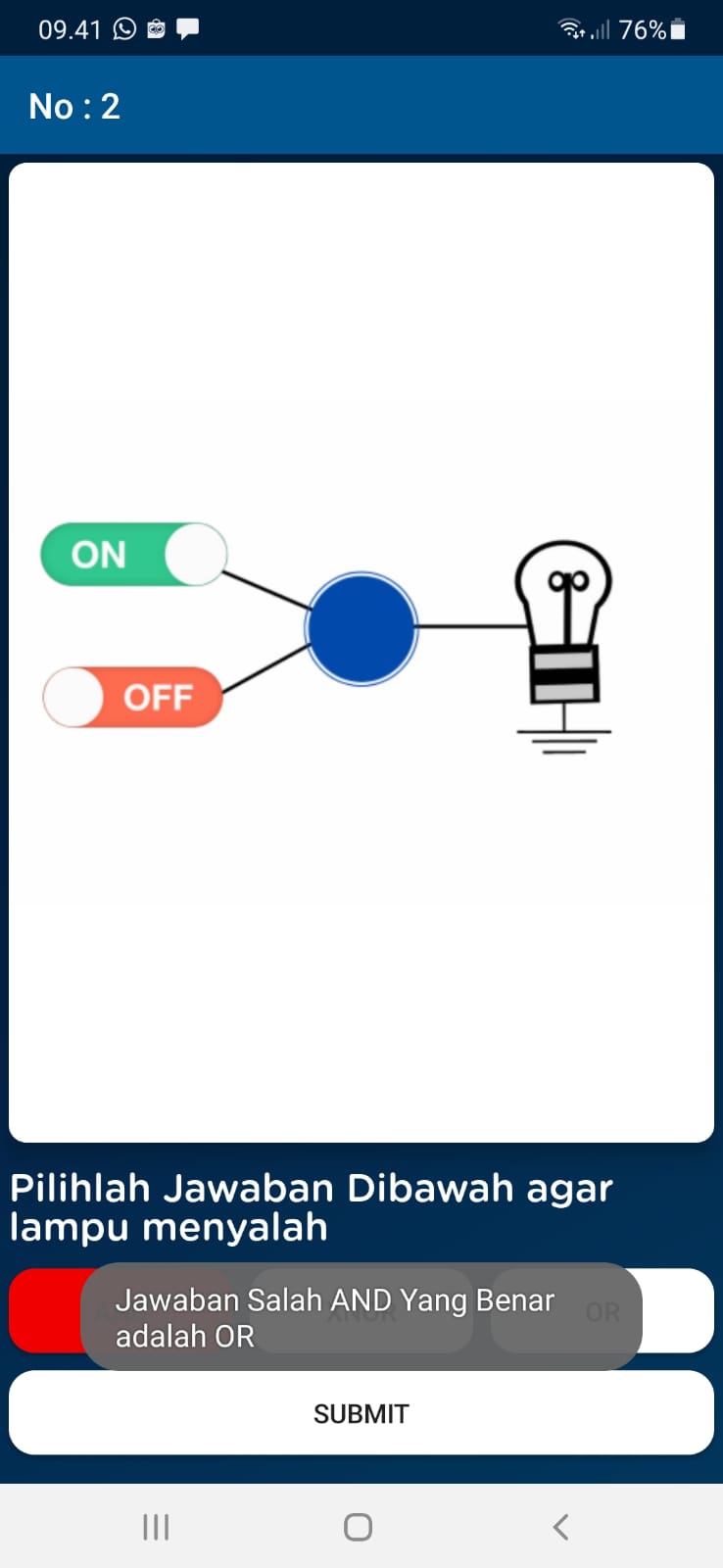


**Gambar 4. 12 Pengujian Tombol Submit (Benar)**

Jika data yang dimasukkan bernilai Benar maka Lampu akan hidup dan terdapat *pop­-up* ulasan bahwa jawaban yang kita pilih adalah benar.

**Tabel 4. 12 Pengujian Tombol Submit (Salah)**

| **Pengujian dan Hasil Uji (data salah)** | | | | |
| --- | --- | --- | --- | --- |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button Submit* | Jawaban Salah | Lampu tetap dalam keadaan mati dan muncul ­*pop-up* bahwa jawaban yang di input adalah salah kemudian memberitahu jawaban yang benar. | Sesuai pengamatan | Berhasil |



**Gambar 4. 13 Pengujian Tombol Submit (Salah)**

Jika data yang dimasukkan bernilai Salah maka Lampu tetap mati.

### 4.2.6 Pengujian Tombol *Continue*

Tombol *continue* terletak pada halaman *score*. Tombol ini berfungsi untuk melanjutkan permainan lagi dengan menampilkan kategori soal lagi. Berikut adalah table hasil ujinya.

**Tabel 4. 13 Pengujian Tombol Continue**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pengujian dan Hasil Uji (data benar)** | | | | |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button Continue* | *OnClick* | Menuju ke halaman kategori. | Sesuai | Berhasil |

### 4.2.7 Pengujian Tombol Exit

Tombol *exit* pada halaman *score*. Berfungsi menyudahi permainan dan menuju ke halaman utama. Berikut adalah table hasil ujinya.

**Tabel 4. 14 Pengujian Tombol Exit**

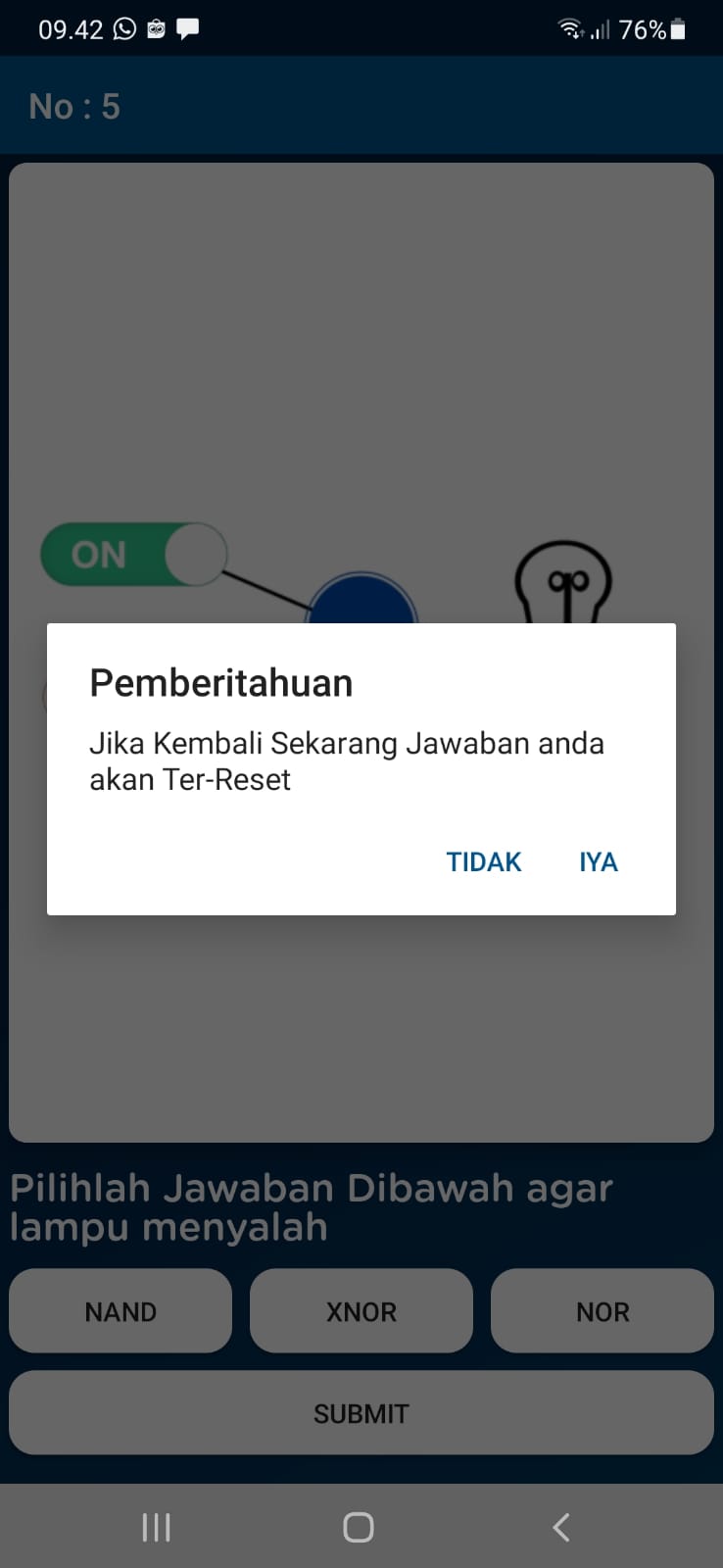
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pengujian dan Hasil Uji (data benar)** | | | | |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button Exit* | *OnClick* | Menuju halaman Utama | Sesuai yang diharapkan | Berhasil |

### 4.2.8 Pengujian Tombol *Back* pada Soal

Tombol *back* pada soal berfungsi jika *user* ingin mengakhiri permainan langsung tanpa harus menyelesaikannya dan jawabanya akan ter-*reset*.

**Tabel 4. 15 Pengujian Tombol Back pada Soal**

| **Pengujian dan Hasil Uji (data benar)** | | | | |
| --- | --- | --- | --- | --- |
| **Yang diuji** | **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Status** |
| *Button back* | *OnClick* | Permainan berakhir dan jawaban di *reset*. | Sesuai yang diharapkan | Berhasil |



**Gambar 4. 14 Pengujian Tombol Back pada Soal**

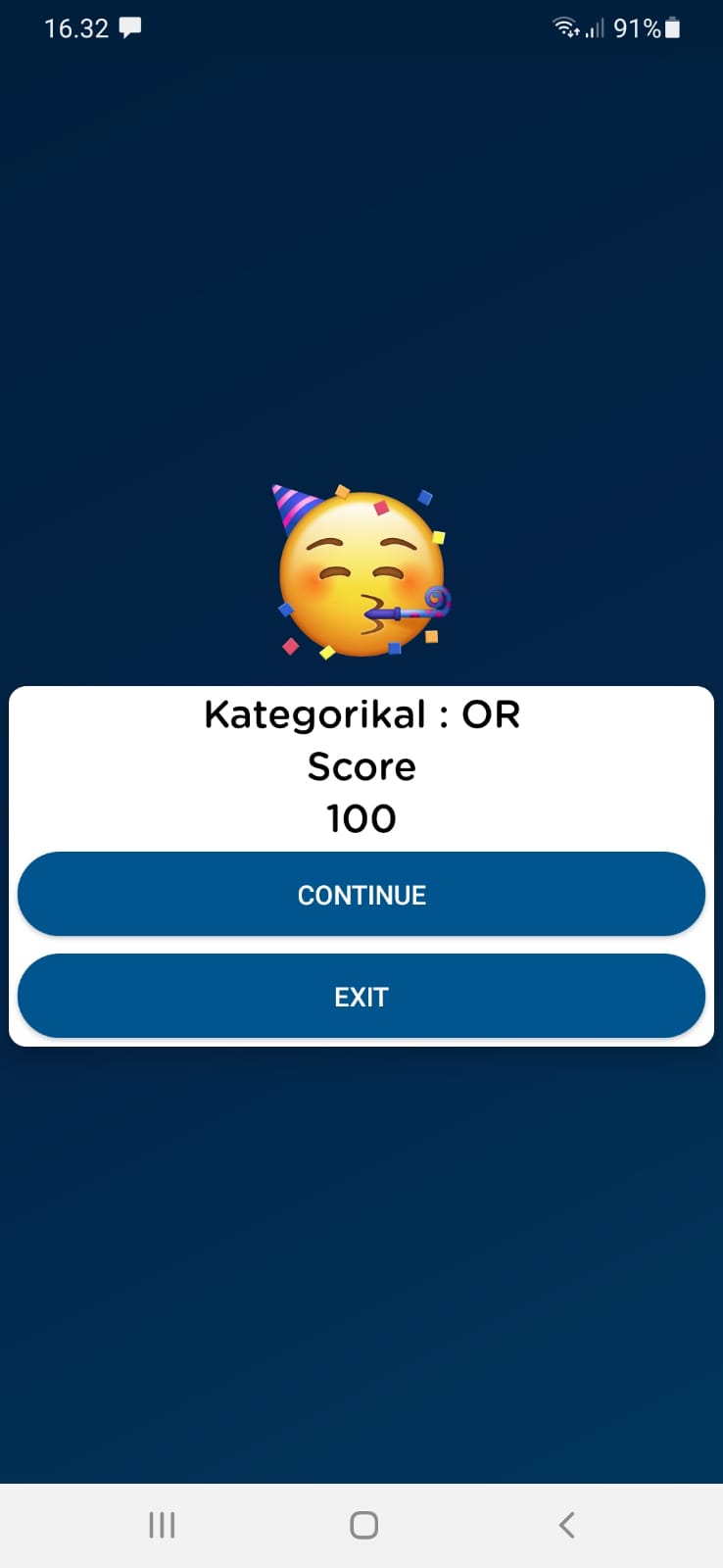
Jika pilih Ya maka permainan akan berakhir dan jawaban di *reset*. Jika Tidak maka permainan akan tetap berlanjut.

### 4.2.9 Pengujian Akurasi Jawaban

Pengujian akurasi jawaban dilakukan dengan cara mencocokan kunci jawaban dengan pilihan jawaban yang benar. Pengujian dilakukan dua kali pada kategori OR dan NOR. Berikut adalah tabel hasil pengujiannya:

**Tabel 4. 16 Pengujian Soal Kategorikal OR**

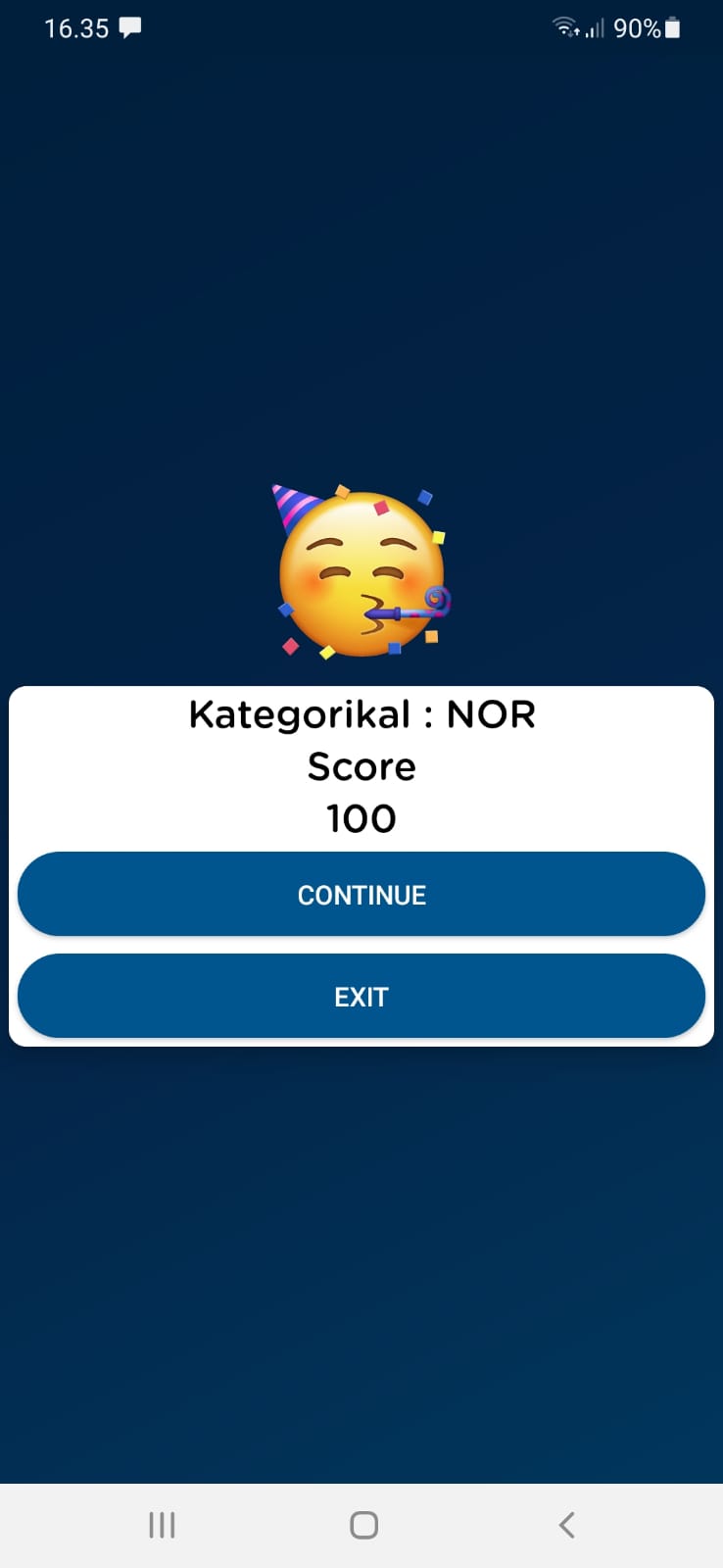
|  |  |  |  |
| --- | --- | --- | --- |
| Soal No | Pilihan Jawaban (Benar) | Kunci Jawaban | Hasil |
| 1. | ON | ON | Sesuai |
| 2. | ON | ON | Sesuai |
| 3. | AND | AND | Sesuai |
| 4. | NAND | NAND | Sesuai |
| 5. | OR | OR | Sesuai |
| 6. | XNOR | XNOR | Sesuai |
| 7. | NAND | NAND | Sesuai |
| 8. | XOR | XOR | Sesuai |
| 9. | NAND | NAND | Sesuai |
| 10. | NOR | NOR | Sesuai |



**Gambar 4. 15 Hasil Pengujian Soal Kategorikal OR**

**Tabel 4. 17 Pengujian Soal Kategorikal NOR**

|  |  |  |  |
| --- | --- | --- | --- |
| Soal No | Pilihan Jawaban (Benar) | Kunci Jawaban | Hasil |
| 1. | OFF | OFF | Sesuai |
| 2. | OFF | OFF | Sesuai |
| 3. | NOR | NOR | Sesuai |
| 4. | NAND | NAND | Sesuai |
| 5. | OR | OR | Sesuai |
| 6. | OR | OR | Sesuai |
| 7. | NOR | NOR | Sesuai |
| 8. | XOR | XOR | Sesuai |
| 9. | AND | AND | Sesuai |
| 10. | XNOR | XNOR | Sesuai |



**Gambar 4. 16 Hasil Pengujian Soal Kategorikal NOR**

Dari hasil pengujian akurasi jawaban yang dilakukan sebanyak dua kali, diperoleh hasil yaitu:

**Tabel 4. 18** **Akurasi Jawaban**

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Soal (JS) | Jumlah Soal Yang Sesuai (JSYS) | Jumlah Soal Yang Tidak Sesuai (JSYTS) | Tingkat Akurasi  (JSYS/JS)x100% |
| 20 | 20 | 0 | 100% |

## 4.3 Pembahasan Hasil

Berdasarkan hasil pengujian aplikasi *game* Gerbang Logika yang telah dilakukan, diketahui bahwa aplikasi ini berjalan dengan baik. Adapun pembahasan hasil dari aplikasi *game* Gerbang Logika sebagai berikut:

1. Aplikasi *game* Gerbang Logika dibuat menggunakan Android Studio.
2. *Tools* yang digunakan untuk merancang *scenario* pada aplikasi menggunakan *HIPO diagram, UseCase diagram, Activity diagram*, dan *Flowchart diagram.*
3. Pada aplikasi terdapat fitur Latihan, pemilihan kategori soal, dan *score.*
4. Fitur Latihan dapat digunakan agar *user* dapat memahami penggunaan *operator* logika yang digunakan di dalam aplikasi.
5. Fitur Pemilihan Kategori Soal adalah fitur yang dapat menyajikan soal berdasarkan kategori yang dipilih.
6. Kategori soal digolongkan menjadi dua yaitu *Difficulty* (mudah*,* sedang*,* dan sulit) dan Kategorikal (And, Or, Nand, Nor, Xor, dan Xnor).
7. Pada *bank* soal terdapat total **sembilan puluh soal** yang terdiri dari delapan belas soal mudah (And, Or, Nand, Nor, Xor, dan Xnor masing-masing tiga soal), tiga puluh enam soal sedang(And, Or, Nand, Nor, Xor, dan Xnor masing-masing enam soal), dan tiga puluh enam soal sulit(And, Or, Nand, Nor, Xor, dan Xnor masing-masing enam soal).
8. Operator yang digunakan sebanyak enam yaitu And, Or, Nand, Nor, Xor, dan Xnor.
9. Input yang digunakan yaitu dari dua input sampai empat input.
10. Setelah jawaban di input maka soal akan menampilkan *true table* dari rangkaian.
11. Setiap kategori soal terdiri dari lima *modul* yang masing-masing modul terdiri dari sepuluh soal.
12. Pemanggilan acak terletak pada pemanggilan *modul*.
13. Fitur *Score* adalah fitur yang akan menampilkan perolehan nilai berdasarkan jumlah jawaban yang benar.
14. Dalam permainan, soal yang disajikan dijawab hanya dengan menentukan satu jawaban.
15. Aplikasi akan mengkoreksi jawaban secara otomatis dengan mencocokan jawaban dengan kunci jawaban dengan algoritma IF ELSE.
16. Hasil uji coba akurasi jawaban diperoleh seratus persen.
17. Perubahan lampu hidup dan mati memanfaatkan *paging* atau pergantian halaman. Untuk satu desain soal yang utuh diperlukan dua gambar rangkaian yang sama tetapi diberi perbedaan untuk desain lampu. Gambar satu lampu diberi warna kuning sebagai presentasi bahwa lampu hidup dan gambar dua tidak diberi warna yang menandakan bahwa lampu mati.

# BAB V

**KESIMPULAN DAN SARAN**

## 5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan dapat kita tarik kesimpulan bahwa (tentang apa yang sudah dikerjakan):

1. Aplikasi mudah digunakan karena berbasis Android.
2. Waktu pengerjaan soal lebih singkat karena tidak ada proses pembuatan table kebenaran.
3. Operator yang digunakan sebanyak enam yaitu And, Or, Nand, Nor, Xor, dan Xnor.
4. Input yang digunakan yaitu dari dua input sampai empat input.
5. Kategori soal digolongkan menjadi dua yaitu *Difficulty* (mudah*,* sedang*,* dan sulit) dan Kategorikal (And, Or, Nand, Nor, Xor, dan Xnor).
6. Kategori soal digolongkan menjadi dua yaitu *Difficulty* berdasarkan jumlah input (mudah*,* sedang*,* dan sulit) dan Kategorikal berdasarkan operator (And, Or, Nand, Nor, Xor, dan Xnor).
7. Pada *bank* soal terdapat total sembilan puluh soal yang terdiri dari delapan belas soal mudah (And, Or, Nand, Nor, Xor, dan Xnor masing-masing tiga soal), tiga puluh enam soal sedang(And, Or, Nand, Nor, Xor, dan Xnor masing-masing enam soal), dan tiga puluh enam soal sulit(And, Or, Nand, Nor, Xor, dan Xnor masing-masing enam soal).
8. True Table akan muncul setelah jawaban di *submit,* sehingga proses pembelajaran akan tersampaikan.
9. Setiap kategori soal terdiri dari lima *modul* yang masing-masing modul terdiri dari sepuluh soal.
10. Pemanggilan acak terletak pada pemanggilan *modul.*
11. Proses koreksi jawaban dilakukan secara otomatis oleh system dengan mencocokan jawaban yang dipilih dengan kunci jawaban.
12. Penerapan Aljabar Boolean terletak pada lampu yang hanya menghasilkan dua output yaitu lampu hidup dan lampu mati.
13. Jika jawaban benar maka lampu pada rangkaian akan hidup dan jika jawaban salah maka lampu akan tetap mati.

## 5.2 Saran

Berikut adalah saran yang penting untuk diperhatikan dalam pengembangan penelitian ini agar dapat lebih baik lagi yaitu:

1. Aplikasi ini belum *support* untuk tampilan Tab dan perangkat dengan *Operating System* selain Android.
2. Kontrol aplikasi untuk melakukan tambah, kurang, edit, dan hapus soal masih dilakukan *manual* yaitu dengan merubah kode program dan *build* ulang menggunakan kode *editor* (Android Studio).

# DAFTAR PUSTAKA

Arifin, R. (2017). Media Pembelajaran Berbasis Video Animasi Pada Mata. *Bina Insani ICT Journal. 4 (1)*, 83-04.

Hanifah, S. M. (2018). *Sistem Pendukung Keputusan Menentukan Mahasiswa Berprestasi Menggunakan Metode Simple Additive Weighting.* Jakarta.

Herliandy, L. D., Nurhasanah, Suban, M. E., & Kuswanto, H. (2020). Pembelajaran Pada Masa Pandemi Covid-19. *Jurnal Teknologi Pendidikan*, 65-70.

Jakarta, P. D. (2021, Februari 21). *Data Pemantauan*. Retrieved from Corona Jakarta: corona.jakarta.go.id/id/data-pemantauan

Kuswanto, J., & Radiansah, F. (2018). Media Pembelajaran Berbasis Android Pada Mata Pelajaran Sistem Operasi Jaringan Kelas XI. *Jurnal Media Infotama vol. 14*, 15-20.

Ladjamudin. (2013). *Analisis dan Desain Sistem Informasi.* Yogyakarta: Graha.

Pressman. (2015). *Rekayasa Perangkat Lunak: Pendekatan Praktisi Buku I.* Yogyakarta: Andi.

Rezaldy, A., Hayadi, B. H., & Ropianto, M. (2021, Maret 21). *SEJARAH SINGKAT DAN MACAM-MACAM LOGIKA INFORMATIKA*. Retrieved from Academia: academia.edu

Saputra, A., & Agustin, F. (2012). *Membangun Sistem Aplikasi E-Commerce dan SMS.* Jakarta: PT Elex Media Komputindo.

Sari, E. L., HD, N. S., & Arnandi, W. (2018). RANGKAIAN SISTEM KELISTRIKAN LAMPU PENERANGAN, LAMPU REM, DAN KLAKSON PADA KENDARAAN MITSUBISHI L300. *RIDTEM (Riset Diploma Teknik Mesin). 1 (1)*, 1-5.

Sugiartowo, & Ambo, S. N. (2018). SIMULASI RANGKAIAN KOMBINASIONAL SEBAGAI MEDIA PEMBELAJARAN SISTEM DIGITAL PADA FAKULTAS TEKNIK UNIVERSITAS MUHAMMADIYAH JAKARTA.

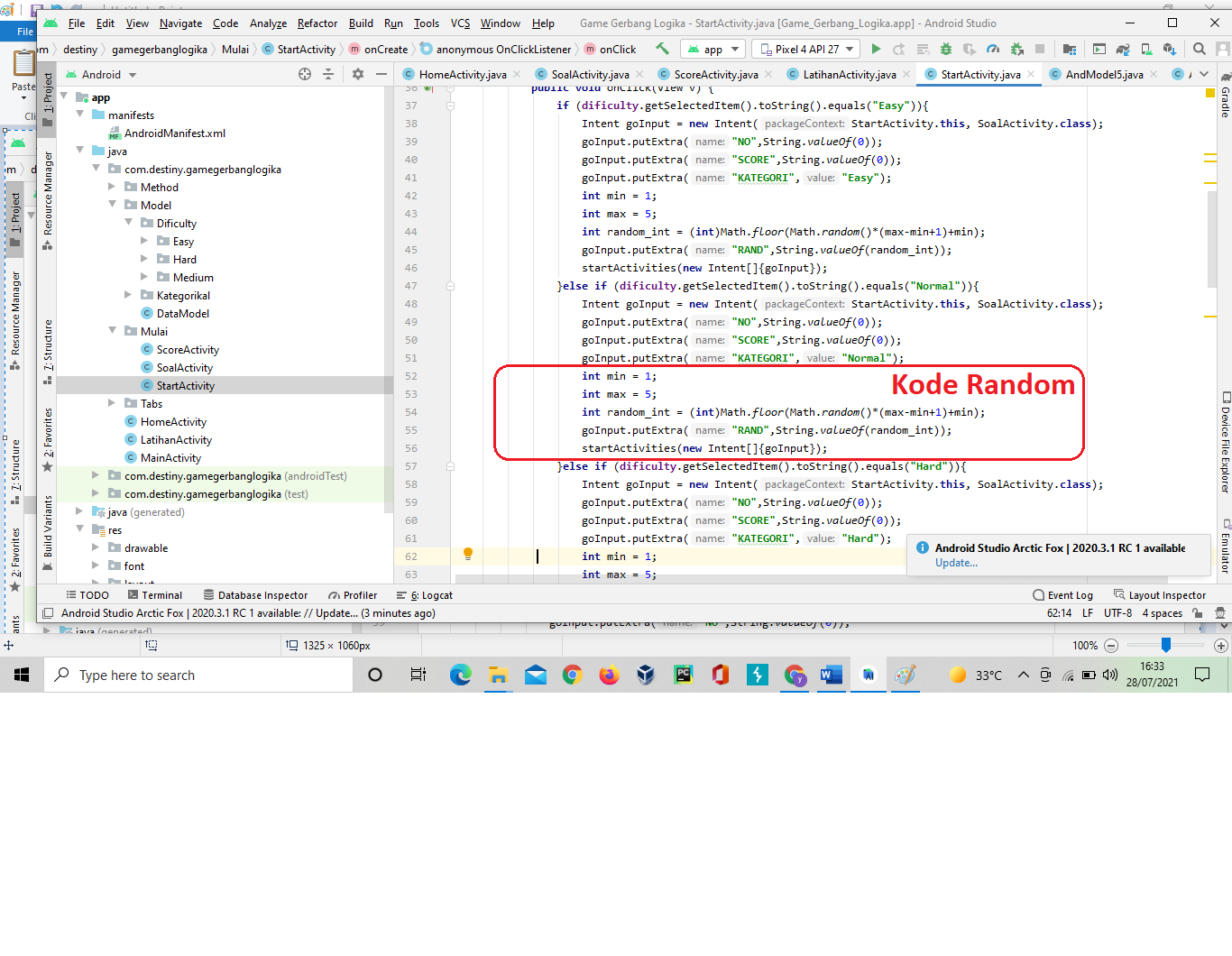
Sukamto, R. A., & Shalahudin, M. (2015). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek.* Bandung: Informatika.

Tedjo, D. (2017). *Hirearcy Plus Input Output Process.*

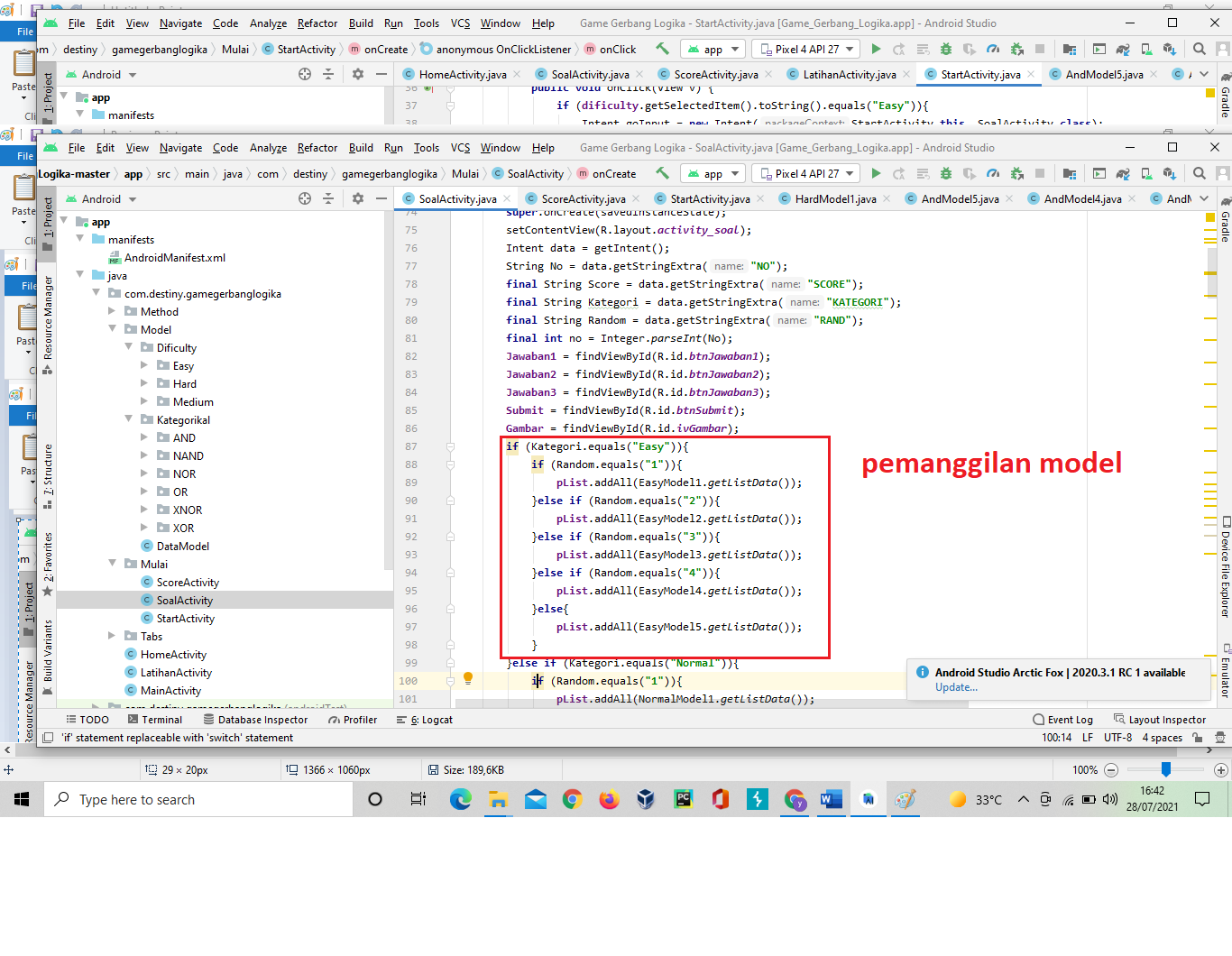
Universitas Muhammadiyah Jakarta. (2021, Maret 21). *About: Fakultas Teknik UMJ*. Retrieved from Fakultas Teknik Universitas Muhammadiyah Jakarta: https://ft.umj.ac.id

Yasin. (2012). *Rekayasa Perangkat Lunak Berorientasi Objek.* Jakarta: Mitra Wacana Media.

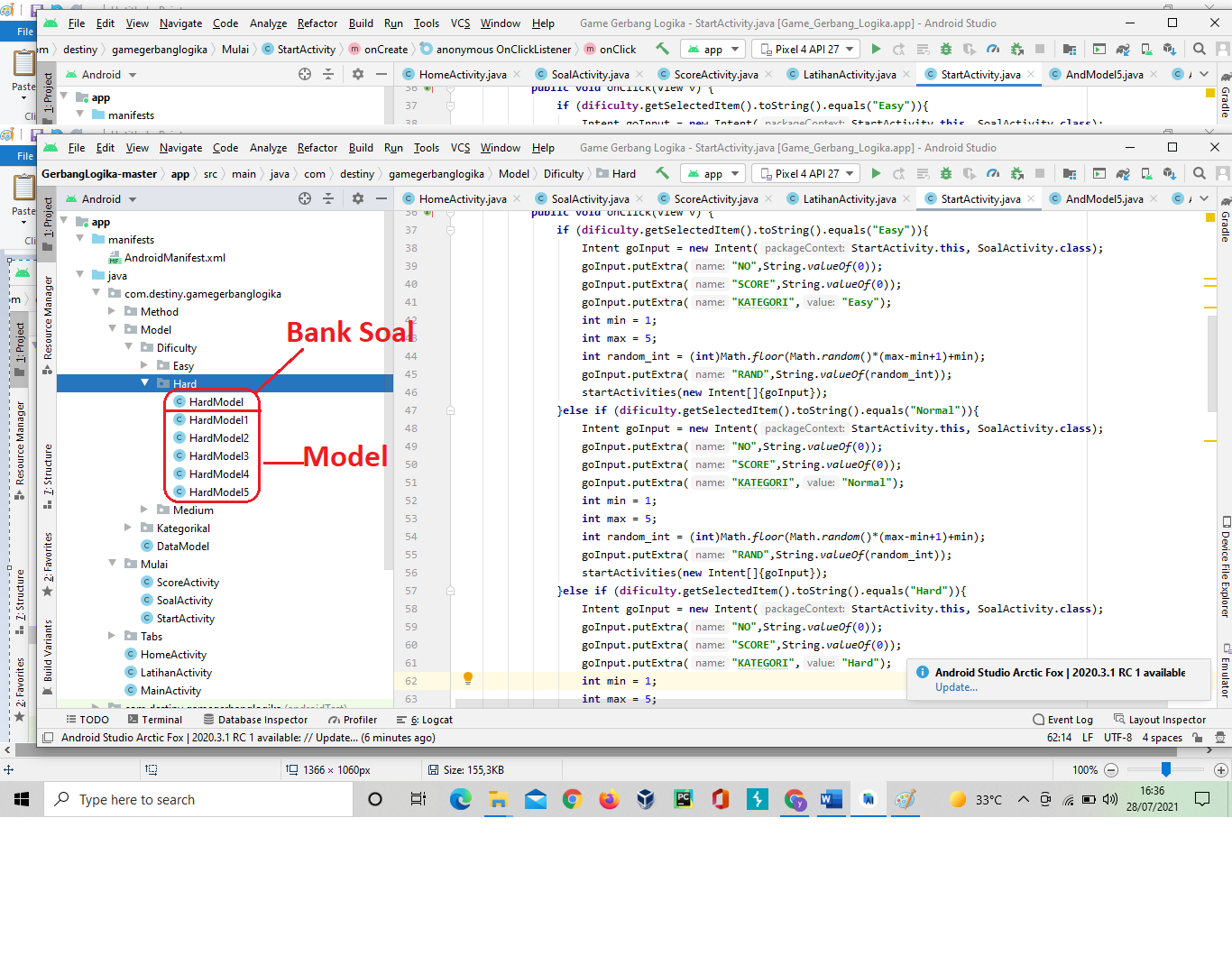
**Lampiran 1 Kode Random**



**Lampiran 2 Pemanggilan Model**



**Lampiran 3 Bank soal dan Model**



**Lampiran 4 Paging Lampu Hidup dan Mati**

