**JTRISTE**, Vol.13, No.1, Maret 2026, pp. 11~22
ISSN: **2355-3677 | e-ISSN : 2460-8548**

# Pengujian White Box Basis Path Testing Fitur Menghitung Aplikasi ComfyLearn

**Oleh:**

**Michelle Valene Tanod[1], Husni Angriani [2*], Afifah[3]**
[1,2]Sistem Informasi, STMIK KHARISMA Makassar
[3]Bisnis Digital, STMIK KHARISMA Makassar
e-mail: [1]michellevalene_22@kharisma.ac.id, [2] husniangriani@kharisma.ac.id,
[3]afifah@kharisma.ac.id

***Abstrak:*** Penelitian ini bertujuan untuk menguji kode program fitur pembelajaran pada aplikasi ComfyLearn menggunakan metode *white box testing* dengan teknik *basis path*. ComfyLearn merupakan aplikasi edukatif berbasis Android yang ditujukan bagi anak usia dini dalam kegiatan belajar sambil bermain. Untuk memastikan kualitas dan keandalan aplikasi, diperlukan pengujian perangkat lunak yang sistematis. Penelitian ini menerapkan metode *white box testing* dengan teknik *basis path testing* untuk menguji kode program pada fitur pembelajaran menghitung, karena metode ini mampu mengevaluasi logika program secara menyeluruh. Teknik ini mencakup pembuatan *flowchart*, *flowgraph*, *graph matrix,* perhitungan *cyclomatic complexity*, penentuan jalur independen, serta penyusunan *test case*. Berdasarkan hasil perhitungan menunjukkan bahwa fitur menghitung memiliki nilai *cyclomatic complexity* sebesar 5, termasuk kategori rendah. Hal ini berarti kompleksitas logika program tidak terlalu tinggi sehingga potensi *error* yang ditemukan relatif sedikit. Terdapat 5 jalur independen yang perlu diuji. Pengujian dilakukan melalui *unit testing* dan *manual testing* untuk memverifikasi apakah proses pembelajaran telah berjalan sesuai dengan yang diharapkan. Hasil pengujian menunjukkan bahwa kode program pada fitur pembelajaran menghitung berfungsi dengan baik dan sesuai ekspektasi setelah dilakukan sebuah perbaikan. Dengan demikian dapat disimpulkan bahwa struktur kode pada bagian fitur tersebut cukup andal dan dapat dijadikan dasar untuk pengembangan lebih lanjut.

***Kata kunci:*** *Pembelajaran, white box testing, basis path testing, cyclomatic complexity, test case*

***Abstract:*** This research aims to test the learning features in the ComfyLearn application using the *white box testing* method with the *basis path* technique. ComfyLearn is an Android-based educational application aimed at young children in learning while playing. To ensure the quality and reliability of applications, systematic software testing is required. This research applies the *white box testing* method with the *basis path testing* technique to test the program code for the learning counting feature, because this method is able to evaluate the program logic as a whole. This technique includes making *flowcharts*, *flowgraphs*, *graph matrices*, calculating *cyclomatic complexity*, determining independent paths, and preparing *test cases*. Based on the calculation results, it shows that the counting feature has a *cyclomatic complexity* value of 5, including the low category. This means that the complexity of the program logic is not too high so there is relatively little potential for *errors* to be found. There are 5 independent paths that need to be tested. Testing is carried out through *unit testing* and *manual testing* to verify whether the learning process has run as expected. The test results show that the program code in the learning counting feature functions well and meets expectations after making improvements. Thus, it can be concluded that the code structure in this feature section is quite reliable and can be used as a basis for further development.

***Keywords:*** *Learning, white box testing, basis path testing, cyclomatic complexity, test case*

[*] Corresponding author : Husni Angriani (husniangriani@kharisma.ac.id)

***Diterima :*** *Agustus, 2025* ***Disetujui :*** *Oktober, 2025* ***Dipublikasikan :*** *Maret, 2026*

## 1. Pendahuluan

Di era perkembangan teknologi saat ini telah membawa perubahan besar dalam kehidupan manusia, contohnya dalam dunia pendidikan. Di dunia pendidikan metode pembelajaran konvensional terkadang kurang menarik dan membuat anak cepat merasa bosan, maka dari itu dikembangkan berbagai macam aplikasi untuk membantu aktivitas sehari-hari dalam belajar, dan hiburan anak-anak [1]. Penggunaan aplikasi *mobile* berbasis edukasi menjadi salah satu solusi yang efektif, karena dapat diakses kapan saja dan di mana saja, serta memungkinkan anak memiliki rasa ingin belajar dengan pendekatan yang menyenangkan [2]. Aplikasi ComfyLearn merupakan platform pembelajaran interaktif yang dirancang khusus untuk anak usia 3 sampai 6 tahun dalam belajar membaca, menghitung, dan bermain secara bersamaan. Aplikasi ini mengusung konsep belajar sambil bermain, dimana anak-anak diminta agar aktif berinteraksi dengan aplikasi untuk meningkatkan minat dan keterlibatan anak-anak dalam proses pembelajaran [3]. Aplikasi ComfyLearn memiliki fitur pembelajaran membaca yang menampilkan urutan huruf lengkap dengan suara pelafalan serta pengenalan angka. Terdapat juga materi pengenalan suku kata sederhana seperti ba, bi, bu, be, bo. Sementara itu pada fitur belajar menghitung anak-anak akan diberikan beberapa contoh soal penjumlahan dan pengurangan sederhana yang dirancang untuk mengasah tingkat pemahaman anak.

Seiring meningkatkannya kebutuhan akan perangkat lunak yang mendukung kegiatan belajar, penting untuk memastikan bahwa aplikasi yang dikembangkan memiliki kualitas yang baik. Meskipun perangkat lunak dapat memberikan berbagai jenis layanan tidak menjamin kualitas yang optimal. Tanpa adanya proses pengujian yang terstruktur dapat menyebabkan perangkat lunak memiliki banyak bug atau *error*, yang dapat mengganggu kenyamanan pengguna [4]. Oleh karena itu, pengujian merupakan salah satu tahap penting dalam pengembangan perangkat lunak. Pengujian perangkat lunak bertujuan untuk mengevaluasi dan memastikan bahwa aplikasi berjalan sesuai dengan kebutuhan pengguna tanpa kendala teknis dengan menggunakan teknik pengujian yang sistematis [5].

Selama proses pengembangan aplikasi ComfyLearn untuk memastikan keberhasilan struktur kode program, diperlukan sebuah pengujian. Pengujian tidak hanya untuk menemukan kesalahan, tetapi juga memvalidasi bahwa seluruh alur logika program telah diimplementasikan dengan benar. Pendekatan pengujian berbasis struktur internal program memungkinkan pengembang untuk menelusuri setiap jalur logika, memastikan bahwa alur eksekusi berjalan sebagaimana mestinya dan tidak menyimpan potensi kegagalan yang tersembunyi [6]. Oleh karena itu, pemilihan metode pengujian yang mampu mengevaluasi alur kendali secara menyeluruh menjadi penting untuk menjamin kualitas perangkat lunak [7]. Secara umum, terdapat dua pendekatan pengujian perangkat lunak, yaitu *white box* dan *black box. White box* pengujian yang berfokus pada logika dan struktur kode suatu program untuk memastikan bahwa setiap pernyataan yang disajikan dalam kode program berjalan sesuai sehingga meningkatkan kualitas dan keandalan dari perangkat lunak [8] [9], sedangkan *black box* berfokus pada penilaian masukan dan keluaran sistem tanpa mempertimbangkan bagaimana kode program diimplementasikan.[10] [11]. Salah satu metode yang sesuai dengan tujuan dari penelitian ini adalah *white box*, karena memungkinkan untuk menganalisis langsung logika program dan alur kode secara menyeluruh [12]. Dalam metode *white box* terdapat beberapa teknik, salah satunya adalah *basis path testing* yang digunakan dalam penelitian ini. *Basis path testing* adalah teknik yang menguji setiap alur program untuk menilai kompleksitasnya, dan setiap lintasan yang diuji satu kali tanpa pengulangan, sehingga seluruh jalur independen dapat terverifikasi [13]. Metode *basis path testing* mencakup pembuatan *flowchart* untuk menggambarkan jalur-jalur dari suatu program yang akan diuji, *flowgraph* dibuat untuk visualisasi struktur kontrol program berdasarkan *flowchart* yang telah dibuat, menghitung *cyclomatic complexity (CC)* untuk mendapatkan nilai tingkat kompleksitas suatu program**,** serta penentuan jalur independen yang diperoleh dari hasil perhitungan *cyclomatic complexity (CC)* dan membuat *test case* [14] [15]. Pada penelitian ini, fokus pengujian dilakukan pada kode program fitur menghitung karena selain menyajikan soal penjumlahan dan pengurangan sederhana, fitur ini juga memiliki tambahan logika seperti *reset progress*, navigasi ke halaman utama, serta tampilan animasi *confetti* ketika pengguna menyelesaikan level. Kompleksitas logika tersebut menjadikan fitur menghitung lebih rawan terhadap kesalahan implementasi, sehingga dipilih sebagai bahan uji dengan metode *white box testing* menggunakan teknik *basis* path testing.

Dengan menerapkan *basis path testing* pada aplikasi ComfyLearn dapat memastikan bahwa kualitas struktur kode program yang telah diterapkan pada aplikasi tersebut memiliki

keandalan dan tingkat keamanan yang sesuai prosedur [16]. Selain itu hasil pengujian yang telah dilakukan dapat menjadi referensi bagi pengembang dalam meningkatkan kualitas aplikasi ComfyLearn untuk pengembangan selanjutnya.

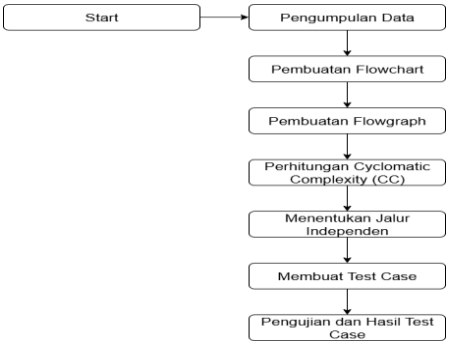
## 2. Metode Penelitian

### 2.1. Jenis Data dan Sumber Data

Dalam penelitian ini jenis data yang digunakan adalah data eksperimental, karena data dikumpulkan melalui eksperimen struktur logika program, *flowchart*, *flowgraph*, serta jalur eksekusi kode program aplikasi ComfyLearn. Analisis dilakukan secara langsung dengan menggunakan metode *white box*, khususnya teknik *basis path testing*

Sumber data dalam penelitian ini termasuk data primer, karena peneliti sendiri yang secara langsung mengumpulkan dan menganalisis kode program aplikasi ComfyLearn tanpa melalui perantara atau pihak manapun.

### 2.2. Tahapan Penelitian




Gambar 1 Tahapan Penelitian

Alur tahapan pengujian pada aplikasi ComfyLearn dapat dilihat pada Gambar 1 yang dimulai dengan pengumpulan data, yaitu pemilihan potongan kode program yang akan diuji, khususnya pada fitur pembelajaran menghitung. Setelah data diperoleh, dilakukan pembuatan *flowchart* yang menggambarkan alur logika program dari potongan kode fitur yang akan diuji. Tahapan selanjutnya adalah pembuatan *flowgraph* berdasarkan *flowchart* yang telah dibuat sebelumnya. *Flowgraph* digunakan untuk mengidentifikasi node (urutan eksekusi program) dan edge (arah alur logika). Selanjutnya dilakukan pembuatan *graph matrix*, yaitu sebuah matriks dua dimensi yang menggambarkan hubungan antar node dalam *flowgraph*. Setiap elemen matriks diisi dengan angka 1 apabila terdapat hubungan antar node, dan 0 jika tidak terdapat hubungan. Matriks ini kemudian digunakan dalam perhitungan *cyclomatic complexity* (*CC*) guna menentukan tingkat kompleksitas kode program serta jumlah jalur independen yang harus diuji. Berdasarkan nilai *CC* yang diperoleh, dilakukan proses penentuan jalur independen dari masing-masing *flowgraph*, yang kemudian dijadikan dasar dalam penyusunan *test case*. Setiap *test case* disusun dengan memuat tiga elemen utama, yaitu kegiatan pengujian, kondisi, dan hasil yang diharapkan. Terakhir, dilakukan pengujian dan analisis hasil *test case* menggunakan dua metode, yaitu *unit testing* dan *manual testing*, untuk memastikan bahwa logika program telah berjalan sesuai dengan yang diharapkan.

## 3. Hasil Dan Pembahasan

### 3.1. Deskripsi Data

Penelitian ini menggunakan sumber data berupa kode program aplikasi ComfyLearn yang dikembangkan menggunakan platform Android. Aplikasi ini dirancang sebagai media pembelajaran interaktif bagi anak usia dini dengan menyediakan berbagai fitur edukatif, termasuk fitur pembelajaran membaca huruf dan menghitung angka. Fokus utama dari pengujian ini adalah kode program pada fitur menghitung, karena fitur tersebut akan sering digunakan dalam mendukung proses belajar anak secara mandiri. Data yang dianalisis berupa struktur logika dari file kode sumber yang menampilkan beberapa contoh soal hitungan hingga akhir level.

Setiap bagian kode program diidentifikasi dan dipetakan berdasarkan fungsi utamanya, kemudian dianalisis secara menyeluruh menggunakan metode *white box testing* dengan

pendekatan *basis path testing*. Pendekatan ini dilakukan untuk mengevaluasi logika internal dari program, memastikan bahwa semua jalur logika program dalam fitur tersebut dapat diuji secara sistematis dan menyeluruh. Dengan demikian, proses yang dilakukan tidak hanya mencakup struktur program secara umum, namun juga detail pengendalian alur, percabangan logika, dan respon program terhadap input pengguna. Berdasarkan data kode yang dianalisis, dilakukan pemetaan file dan aktivitas utama yang diuji dalam penelitian ini.

Untuk memudahkan pemahaman, dibuatkan Tabel Jalur Pengujian Fitur yang dapat dilihat pada Tabel 1

Tabel 1 Jalur Pengujian Fitur

| No | Nama File/Fitur | Nama Activity/Kelas | Fungsi Utama Fitur |
|---|---|---|---|
| 1 | ArithmeticActivity.kt | ArithmeticActivity | Menampilkan soal berhitung berdasarkan level yang sudah disediakan |

Berdasarkan Tabel 1, file atau fitur yang akan menjadi dasar dalam proses pengujian, di mana akan dibuatkan skenario alur programnya. Hasil pengujian dari setiap *test case* akan disertai dengan keterangan berupa status 'berhasil' atau 'gagal'. Berikut merupakan potongan kode program dari fitur menghitung yang dijadikan objek pengujian *white box* sekaligus memperlihatkan asal dari *flowchart* yang disusun dalam penelitian ini.

| No | Deskripsi Node | Potongan Kode Program |
|---|---|---|
| 1 | OnCreate() | `override fun onCreate(savedInstanceState: Bundle?) {`<br>`super.onCreate(savedInstanceState)`<br>`binding = ActivityArithmeticBinding.inflate(layoutInflater)`<br>`setContentView(binding.root)`<br>`loadProgress()`<br>`setupNewQuestion()`<br><br>`binding.nextProblemButton.setOnClickListener {`<br>`if (isAllLevelsCompleted()) {`<br>`Toast.makeText(this, getString(R.string.all_levels_completed), Toast.LENGTH_SHORT).show()`<br>`} else {`<br>`handleNextProblem()`<br>`}`<br>`}`<br>`}` |
| 2 | Inflate layout dengan View Binding | `binding = ActivityArithmeticBinding.inflate(layoutInflater)`<br>`setContentView(binding.root)` |
| 3 | LoadProgress() | `private fun loadProgress() {`<br>`val prefs = getSharedPreferences("ArithmeticProgress", MODE_PRIVATE)`<br>`currentLevel = prefs.getInt("currentLevel", 1)`<br>`problemsCompletedInLevel = prefs.getInt("problemsCompletedInLevel", 0)`<br>`}` |
| 4 | updateProgress UI() | `private fun updateProgressUI() {`<br>`val level = levels[currentLevel - 1]`<br>`val levelName = getString(level.nameResId)` |

| No | Deskripsi Node | Potongan Kode Program |
|---|---|---|
| | | `binding.levelNameText.text = getString(R.string.level_name_format, currentLevel, levelName)`<br>`binding.levelProgressBar.max = problemsPerLevel`<br>`binding.levelProgressBar.progress = problemsCompletedInLevel`<br>`}` |
| 5 | setupNewQuestion() | `private fun setupNewQuestion() {`<br>`updateProgressUI()`<br>`if (isAllLevelsCompleted()) {`<br>`binding.nextProblemButton.isEnabled = false`<br>`Toast.makeText(this, getString(R.string.all_levels_completed), Toast.LENGTH_LONG).show()`<br>`// Jangan generate soal baru`<br>`generateLastQuestionIfCompleted()`<br>`return`<br>`}`<br>`binding.nextProblemButton.isEnabled = true`<br>`generateQuestion()`<br>`}` |
| 6 | Apakah currentlevel > levels.size? | `private fun isAllLevelsCompleted(): Boolean {`<br>`return currentLevel > levels.size ||`<br>`(currentLevel == levels.size && problemsCompletedInLevel >= problemsPerLevel)`<br>`}` |
| .. | .. | ... |

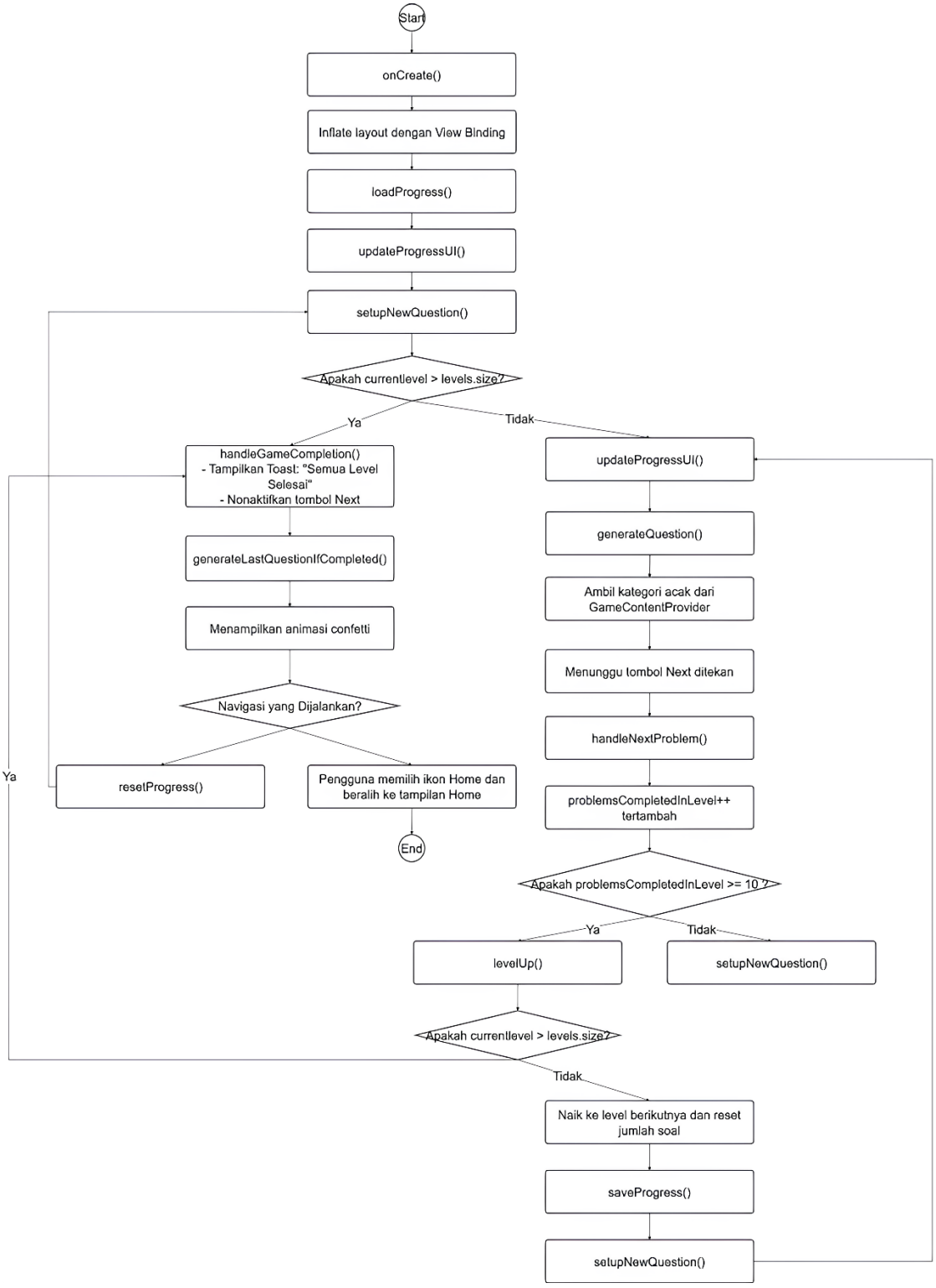
### 3.2. Pembahasan

Bagian ini akan membahas hasil pengujian *white box* terhadap kode program pada fitur menghitung dalam aplikasi ComfyLearn menggunakan *teknik basis path testing*, dan langkah-langkah pembuatan penelitian ini sesuai dengan tahapan penelitian pada Gambar 1.

Untuk mendukung proses analisis jalur logika program, ditampilkan *flowchart* kode program fitur menghitung dapat dilihat pada Gambar 2, yang menjadi dasar dalam pembuatan *flowgraph*, perhitungan kompleksitas siklomatik *cyclomatic complexity (CC)*, identifikasi jalur independen, dan penyusunan *test case*. Untuk menghitung *cyclomatic complexity (CC)* menggunakan Persamaan 1:

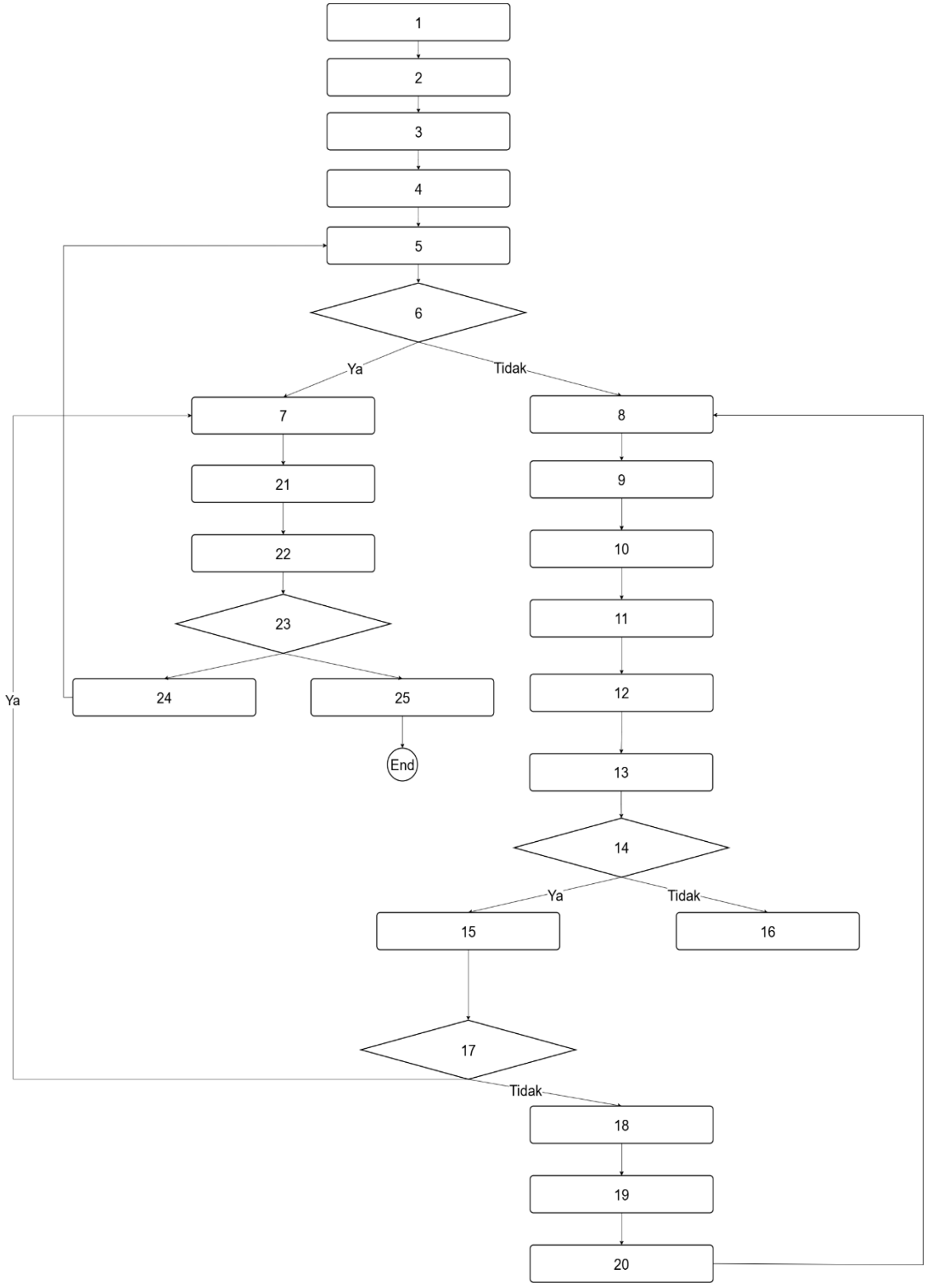
$$V(G) = E - N + 2P$$

Keterangan:
V(G) = *Cyclomatic Complexity*
E = jumlah *edge*
N = jumlah *node*
P = jumlah komponen terhubung (umumnya 1)




Gambar 2 *Flowchart* Kode Program Fitur Menghitung

Setelah pembuatan *flowchart*, dibuatkan *flowgraph* untuk mengetahui tiap *node* dan *edge*. Untuk *flowgraph* kode program fitur menghitung dapat dilihat pada Gambar 3.




Gambar 3 *Flowgraph* Kode Program Fitur Menghitung

Setelah pembuatan *flowgraph*, dibuatkan *graph matrix* yang merupakan matriks dua dimensi yang jumlah baris dan kolomnya disesuaikan dengan jumlah node pada *flowgraph*. Setiap elemen dalam matriks diisi dengan angka 1 jika terdapat hubungan antar node (edge), dan 0 jika tidak terdapat hubungan. *Graph matrix* ini dibuat untuk mendukung perhitungan *cyclomatic complexity (CC)*. *Graph matrix* fitur menghitung dapat dilihat pada Gambar 4.

| Dari/Ke | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | end |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | 1 | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | 1 | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | 1 | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | 1 | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | 1 | 1 | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| 8 | | | | | | | | | 1 | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | 1 | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | 1 | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | 1 | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | 1 | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | 1 | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | 1 | 1 | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | 1 | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | 1 | | | | | | | | | | | 1 | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | | 1 | | | | | | | |
| 19 | | | | | | | | | | | | | | | | | | | | 1 | | | | | | |
| 20 | | | | | | | | 1 | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | | | | | | | | 1 | | | | |
| 22 | | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| 23 | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | |
| 24 | | | | | 1 | | | | | | | | | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | | | | | | | | | | | | 1 |

Gambar 4 *Graph Matrix* Fitur Menghitung

Setelah pembuatan *graph matrix*, dicarilah nilai *cyclomatic complexity (CC)* untuk fitur menghitung dengan menggunakan Persamaan 1. Berikut ini hasil perhitungan *cyclomatic complexity (CC)* dari fitur menghitung:

1. Fitur menghitung, berdasarkan *flowgraph* kode program fitur menghitung, diperoleh:
   Jumlah *node* (N) = 25
   Jumlah *edge* (E) = 28
   Komponen terhubung (P) = 1
   Maka nilai *cyclomatic complexity* dengan menggunakan Persamaan 1 yaitu:

$$V(G) = 28 – 25 + 2(1) = 5$$

Nilai ini menunjukkan bahwa terdapat 5 jalur independen yang harus diuji dari kode program fitur menghitung.

Setelah menghitung *cyclomatic complexity (CC)*, didapatkan nilai *CC* pada fitur menghitung sebesar 5, maka terdapat 5 jalur independen atau jalur unik yang akan dibuat menjadi *test case* sebagai bahan pengujian. Berikut ini jalur independen fitur menghitung.

1. Jalur Independen Fitur Menghitung
   Jalur independen 1
   Pengguna memulai pembelajaran menghitung, dimulai dari level pertama, lalu lanjut ke soal berikutnya.
   1 → 2 → 3 → 4 → 5 → 6 (Tidak) → 8 → 9 → 10 → 11 → 12 → 13 → 14 (Tidak) →16.
   Jalur independen 2
   Pengguna mempelajari 10 soal dalam 1 level, lalu meningkat ke level berikutnya dan pembelajaran dilanjutkan.
   1 → 2 → 3 → 4 → 5 → 6 (Tidak) → 8 → 9 → 10 → 11 → 12 → 13 → 14 (Ya) → 15 → 17 (Tidak) → 18 → 19 → 20 → 8.
   Jalur independen 3
   Pengguna mempelajari beberapa contoh soal, sampai berada di level terakhir, sehingga ditampilkan confetti dan muncul pesan “Semua Level Selesai”.
   1 → 2 → 3 → 4 → 5 → 6 (Tidak) → 8 → 9 → 10 → 11 → 12 → 13 → 14 (Ya) → 15 → 17 (Ya) → 7 → 21→ 22.
   Jalur independen 4
   Pengguna memulai pembelajaran, tetapi seluruh level telah selesai, sehingga langsung ditampilkan pesan “Semua Level Selesai” dan pengguna memilih ikon Home untuk kembali ke halaman utama.
   1 → 2 → 3 → 4 → 5 → 6 (Ya) → 7 → 21 → 22 → 23 → 25 → end.
   Jalur independen 5
   Setelah game selesai (handleGameCompletion() dipanggil), pengguna memilih untuk resetProgress() agar mulai dari awal lagi.
   1 → 2 → 3 → 4 → 5 → 6 (Ya) → 7 → 21 → 22 → 23 → 24 → 5.

Setelah membuat jalur independennya, dibuatkan sebuah *test case* untuk mengetahui apakah *test case* dari kode program fitur menghitung dapat berjalan sesuai yang diharapkan. Hasil pengujian *test case* kode program fitur menghitung dapat dilihat pada Tabel 2 dan Tabel 3.

Tabel 2 Hasil Pengujian *Test Case* Kode Program Fitur Menghitung

| No | *Test Case* | Kegiatan Pengujian | Hasil yang diharapkan | Keterangan |
|---|---|---|---|---|
| TC1 | Menampilkan dan menyelesaikan soal di Level 1 | *ArithmeticActivity* dijalankan, menampilkan soal level 1, klik tombol *next* untuk lanjut ke contoh soal berikutnya sehingga *progress bar* berjalan | Memulai pembelajaran, kemudian contoh soal level 1 di tampilkan. Selanjutnya menekan tombol *Next* sehingga *progress bar* berjalan sesuai jumlah soal yang diselesaikan. | Berhasil |
| TC2 | Menyelesaikan contoh soal dan naik ke level berikutnya | Menyelesaikan beberapa contoh soal pembelajaran di tiap level yang disediakan untuk naik ke level berikutnya | Berhasil menyelesaikan beberapa contoh soal pada satu level, sehingga sistem secara otomatis menaikkan pengguna ke level berikutnya dan menampilkan contoh soal dari level baru. | Gagal |
| TC3 | Menyelesaikan seluruh level hingga selesai | Menyelesaikan semua soal dari level 1 sampai level 4 | Sistem menampilkan animasi confetti dan pesan “Semua Level Selesai”. Tombol Next dinonaktifkan sehingga tidak bisa lanjut ke soal lagi.pesan “Semua Level Selesai” | Gagal |
| TC4 | Membuka kembali activity saat semua level telah selesai lalu kembali ke Home | Membuka ArithmeticActivity dengan kondisi seluruh level sudah selesai, kemudian menekan tombol Home | Sistem langsung menampilkan animasi confetti dan pesan “Semua Level Selesai” dan ketika tombol Home dipilih, pengguna dialihkan ke halaman utama. | Gagal |

| | | | | |
|---|---|---|---|---|
| TC5 | Reset progress setelah semua level selesai | Setelah handleGameCompletion() dipanggil, pengguna menekan tombol reset | Sistem mereset progres belajar, mengatur ulang ke level awal dan menyimpan ulang data progress baru. | Gagal |

Tabel 3 Hasil Perbaikan Pengujian *Test Case* Kode Program Fitur Menghitung

| No | *Test Case* | Kegiatan Pengujian | Hasil yang diharapkan | Keterangan |
|---|---|---|---|---|
| TC1 | Menampilkan dan menyelesaikan soal di Level 1 | *ArithmeticActivity* dijalankan, menampilkan soal level 1, klik tombol *next* untuk lanjut ke contoh soal berikutnya sehingga *progress bar* berjalan | Memulai pembelajaran, kemudian contoh soal level 1 di tampilkan. Selanjutnya menekan tombol *Next* sehingga *progress bar* berjalan sesuai jumlah soal yang diselesaikan. | Berhasil |
| TC2 | Menyelesaikan contoh soal dan naik ke level berikutnya | Menyelesaikan beberapa contoh soal pembelajaran di tiap level yang disediakan untuk naik ke level berikutnya | Berhasil menyelesaikan beberapa contoh soal pada satu level, sehingga sistem secara otomatis menaikkan pengguna ke level berikutnya dan menampilkan contoh soal dari level baru. | Berhasil |
| TC3 | Menyelesaikan seluruh level hingga selesai | Menyelesaikan semua soal dari level 1 sampai level 4 | Sistem menampilkan animasi confetti dan pesan “Semua Level Selesai”. Tombol Next dinonaktifkan sehingga tidak bisa lanjut ke soal lagi.pesan “Semua Level Selesai” | Berhasil |
| TC4 | Membuka kembali activity saat semua level telah selesai lalu kembali ke Home | Membuka ArithmeticActivity dengan kondisi seluruh level sudah selesai, kemudian menekan tombol Home | Sistem langsung menampilkan animasi confetti dan pesan “Semua Level Selesai” dan ketika tombol Home dipilih, pengguna dialihkan ke halaman utama. | Berhasil |

| | | | | |
|---|---|---|---|---|
| TC5 | Reset progress setelah semua level selesai | Setelah handleGameCompletion() dipanggil, pengguna menekan tombol reset | Sistem mereset progres belajar, mengatur ulang ke level awal dan menyimpan ulang data progress baru. | Berhasil |

Berdasarkan hasil pengujian yang dapat dilihat pada Tabel 2 dan Tabel 3 *test case* kode program fitur menghitung, seluruh skenario yang diuji hanya 1 alur yang berhasil dan bagian *test case* 2, *test case* 3, *test case* 4, dan *test case* 5 terjadi kegagalan atau *failed* yang artinya adanya kesalahan dalam logika internal kode program fitur menghitung. Maka dilakukan sebuah perbaikan untuk *test case* yang bermasalah agar hasil semua *test case* dapat berjalan optimal. *Test case* yang dirancang mencakup menampilkan contoh soal hitungan, animasi confetti, dll. Setelah dilakukan sebuah perbaikan dapat disimpulkan bahwa logika program pada fitur menghitung telah berjalan dengan baik.

## 4. Kesimpulan

Berdasarkan hasil pengujian aplikasi ComfyLearn dengan metode *white box* menggunakan teknik *basis path* menunjukkan bahwa selama proses uji ditemukan beberapa kendala pada kode program fitur menghitung. Oleh karena itu, diperlukan perbaikan kode program agar fungsi pembelajaran menghitung dapat berjalan sesuai kebutuhan pengguna. Pengujian dilakukan dengan menyusun *flowchart*, *flowgraph* dan *graph matrix* dari kode program fitur menghitung, dilanjutkan dengan perhitungan *cyclomatic complexity (CC)* untuk menentukan jumlah jalur independen yang perlu diuji. Hasil nilai *cyclomatic complexity* fitur menghitung sebesar 5 menunjukkan bahwa terdapat 5 jalur independen, yang seluruhnya diuji melalui *test case* yang akan dirancang. Dalam standar pengujian perangkat lunak, nilai *CC* fitur menghitung termasuk dalam kategori rendah yang berarti jumlah jalur logika yang perlu diuji tidak terlalu banyak, serta struktur program cukup sederhana dan mudah dipahami. Hal ini juga menunjukkan bahwa kemungkinan terjadinya error atau kesalahan logika dalam program relatif kecil. Meskipun demikian selama proses pengujian *unit testing dan manual testing* masih ditemukan beberapa kendala seperti fungsi penyajian soal hitungan secara acak dalam satu level, serta menampilkan pesan “Semua Level Selesai” yang belum berjalan optimal, sehingga diperlukan perbaikan kode program pada fitur pembelajaran menghitung agar fungsionalitas aplikasi dapat berjalan sepenuhnya sesuai kebutuhan. Secara keseluruhan, hasil pengujian pada kode program fitur menghitung telah berjalan sesuai yang diharapkan setelah mengalami perbaikan, sehingga logika perhitungan dapat berfungsi dengan benar dan mendukung anak dalam memahami konsep penjumlahan dan pengurangan secara interaktif.

## Daftar Pustaka

[1] I. Bahroni and R. Purwanto, “Aplikasi Pembelajaran (E-learning) Mengenal Huruf Hijaiyah bagi Anak-anak Berbasis Mobile untuk Mendukung Pembelajaran Secara Mandiri,” *J. Edukasi dan Penelit. Inform.*, vol. 4, no. 2, p. 163, 2018, doi: 10.26418/jp.v4i2.25566.

[2] Y. J. Solissa, F. Putra, A. N. Putri, and S. R. C. Nursari, “Pengujian White Box Berbasis Path pada Form Daftar Jobstreet.co.id,” *KONSTELASI Konvergensi Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 353–362, 2023, doi: 10.24002/konstelasi.v3i2.8287.

[3] W. Mandiri, I. A. Sobari, and F. Akbar, “493992-None-7a907Fc3,” vol. IV, no. 2, pp. 159–164, 2018.

[4] Y. Pratiwi and L. W. Widianti, “Implementasi Whitebox Testing Dengan Teknik Basis Path Pada Pengujian Halaman Pencarian Program Promo,” *J. Kecerdasan Buatan dan Teknol. Inf.*, vol. 4, no. 2, pp. 173–180, 2025, doi: 10.69916/jkbti.v4i2.280.

[5] M. I. Shiddiq, “Implementasi White Box Testing Berbasis Path Pada Form Login Aplikasi Berbasis Web,” *J. Siliwangi*, vol. 8, no. 1, pp. 1–6, 2022.

[6] C. T. Pratala, E. M. Asyer, I. Prayudi, and A. Saifudin, “Pengujian White Box pada

Aplikasi Cash Flow Berbasis Android Menggunakan Teknik Basis Path," *J. Inform. Univ. Pamulang*, vol. 5, no. 2, p. 111, 2020, doi: 10.32493/informatika.v5i2.4713.

[7] S. Sacra, J. Sains, S. Ningrat, and S. Noris, "Pegujian Aplikasi Sistem Informasi Web Dengan Whitebox Testing Teknik Basis Path," vol. 2, no. 3, pp. 115–121, 2022.

[8] D. Wintana, D. Pribadi, and M. Y. Nurhadi, "Analisis Perbandingan Efektifitas White-Box Testing dan Black-Box Testing," *J. Larik Ldng. Artik. Ilmu Komput.*, vol. 2, no. 1, pp. 8–16, 2022, doi: 10.31294/larik.v2i1.1382.

[9] R. I. Ndaumanu, "Pengujian Sistem Informasi Perpustakaan Berbasis Website dengan Basis Path Testing," *Justek J. Sains dan Teknol.*, vol. 6, no. 1, p. 123, 2023, doi: 10.31764/justek.v6i1.13808.

[10] C. P. C. Munaiseche and G. C. Rorimpandey, "Penerapan Metode Basis Path Analysis dalam Pengujian White Box Sistem Pakar," *Pros. Semin. Nas. Sist. Inf. dan Teknol.*, vol. 5, no. 1, pp. 124–128, 2021.

[11] Alfian Permana Putra and Ilyas Nuryasin, "Pengujian sistem informasi monitoring dava kebab menggunakan white box testing dengan teknik basis path," *INFOTECH J. Inform. Teknol.*, vol. 5, no. 1, pp. 63–75, 2024, doi: 10.37373/infotech.v5i1.1147.

[12] M. Septian Jaelani, "Pengujian White Box Terhadap Sistem Login Berbasis Web Menggunakan Teknik Basis Path," *J. Siliwangi*, vol. 9, no. 2, pp. 32–37, 2023, [Online]. Available: https://loginer.netlify.app/

[13] M. F. Londjo, "Implementasi White Box Testing Dengan Teknik Basis Path Pada Pengujian Form Login," *J. Siliwaangi*, vol. 7, no. 2, pp. 35–40, 2021.

[14] W. A. Nugraha, "Pengujian White Box Berbasis Path Pada Form Autentikasi Berbasis Mobile," *J. Siliwangi Seri Sains dan Teknol.*, vol. 8, no. 2, pp. 42–47, 2022, doi: 10.37058/jssainstek.v8i2.4098.

[15] H. Rafli *et al.*, "JOISIE licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0) PENERAPAN WHITEBOX TESTING PADA PENGUJIAN SISTEM MENGGUNAKAN TEKNIK BASIS PATH," *J. Inf. Syst. Informatics Eng.*, vol. 8, no. 1, pp. 101–111, 2024, [Online]. Available: https://doi.org/10.35145/joisie.v8i1.4229

[16] B. Prasetya and I. Nuryasin, "Pengujian Kualitas Website Pt Media Citra Digitalindo Blitar Menggunakan White Box Testing Dengan Teknik Basis Path," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 10, no. 2, pp. 1595–1601, 2025, doi: 10.29100/jipi.v10i2.7693.