JUSTIN
Jurnal Sistem dan Teknologi Informasi
p-ISSN : 2460-3562 / e-ISSN : 2620-8989
DOI: 10.26418/justin.v9i1.42416
Vol. 09, No. 1, Januari 2020

Aplikasi Smart Entrance Exam Berbasis Android dengan Algoritma Fisher-Yates

Dimas Dandy Aryarajendra Suprapto[a1], Fauziah[a2], Nur Hayati[a3]
[a]*Program Studi Informatika Universitas Nasional*
Jl. Sawo Manila, RT.6/RW.7, Jati Padang, Jakarta Selatan, DKI Jakarta 12520
[1]dimazdandy@gmail.com
[2]fauziah@civitas.unas.ac.id
[3]nurhayati@civitas.unas.ac.id

Abstrak

Ujian masuk penerimaan mahasiswa baru adalah kegiatan rutin yang dilakukan oleh Universitas Nasional tiap ajaran baru. Ujian masuk memanfaatkan teknologi untuk mengurangi penggunaan kertas sebagai media ujian. Perkembangan aplikasi Android diterapkan pada pelaksanaan ujian masuk penerimaan mahasiswa baru yang sebelumnya berbasis komputer (*Computer Base Test*). Tetapi kekurangan dalam penggunaan CBT, camaba harus datang ke kampus dan bergantian antar peserta untuk melakukan ujian. Penelitian ini bertujuan untuk mengembangkan sistem ujian masuk mahasiswa baru secara *online* menjadi aplikasi berbasis android, sehingga memudahkan bagi camaba dalam melakukan ujian serta lebih efisien baik dari segi waktu maupun tempat karena dapat dilakukan dimana saja asal terhubung internet serta tidak perlu bergantian dengan peserta lain untuk melakukan ujian. Peneliain ini menggunakan metode *waterfall* serta algoritma *fisher-yates* yang berguna dalam proses pengacakan soal secara efektif, dan mencegah kecurangan dalam ujian serta terintegrasi firebase *realtime* database dan SQLite untuk mengelola data. Hasil pengujian algoritma *fisher-yates* dengan lima ratus data uji pengacakan soal didapatkan hasil berbeda antar pengujian tanpa duplikasi soal. Kemudian pada pengujian *whitebox* sesuai standar ANSI dengan menghitung parameter *cyclomatic complexity, region*, dan *independent path* dari 4 pengujian diperoleh hasil sama berjumlah 12 artinya logika dan alur program yang dibuat sudah sesuai. Pengujian metode *blackbox* didapatkan hasil pengujian sesuai dengan yang diharapkan. Jadi dapat disimpulkan aplikasi *smart entrace exam* berbasis Android berjalan dengan semestinya sesuai dengan pengujian yang telah dilakukan.

Kata kunci: Ujian, Android, Fisher-Yates, Waterfall, Firebase.

Smart Entrance Exam Application Based on Android with Fisher-Yates Algorithm

Abstract

Entrance examinations for new college students admissions are routine activity carried out by the National University for each new teaching. Entrance exams make use of technology to reduce the use of paper as a test medium. The development of the Android application is applied to the implementation of the entrance examination for new admissions which was previously computer-based (Computer Base Test). But the lack of use of CBT, camaba has to come to campus and take turns between participants to take the exam. This study aims to develop an online new student entrance examination system into an Android-based application, making it easier for camaba to conduct exams and more efficient both in terms of time and place because it can be done anywhere as long as it is connected to the internet and there is no need to take turns with other participants to do it. exam. This study uses the waterfall method and Fisher-yates algorithm which is useful in the process of scrambling questions effectively, and prevents cheating in exams and integrates firebase realtime database and SQLite to manage data. The results of testing the Fisher-Yates algorithm with five hundred randomization tests showed different results between tests without duplication of questions. Then in the whitebox test according to ANSI standards by calculating the parameters of cyclomatic complexity, region, and independent path from 4 tests, the same results were obtained totaling 12, meaning that the logic and program flow made were appropriate. Testing the blackbox method obtained the test results as expected. So it can be concluded that the Android-based smart entrace exam application runs properly according to the tests that have been carried out.

Keywords: Exam, Android, Fisher-Yates, Waterfall, Firebase

I. PENDAHULUAN

Ujian dalam dunia pendidikan adalah untuk mengukur taraf pencapain tujuan dalam proses pembelajaran sehingga peserta didik dan instansi pendidikan dapat mengetahui tingkat kemampuan dalam bidang studi yang di tempuh [1]. Ujian masuk penerimaan mahasiswa baru adalah hal yang rutin dilakukan setiap instansi perguruan tinggi tiap memasuki tahun ajaran baru. Ujian masuk berguna untuk menyeleksi calon mahasiswa baru yang terpilih sesuai dengan kriteria yang ditentukan [2]. Penerapan teknologi informasi sudah banyak digunakan dalam berbagai bidang, salah satunya adalah pendidikan [3]. Teknologi yang telah banyak digunakan untuk melakukan kegiatan ujian masuk adalah dengan menggunakan sistem *Computer Based Test* (CBT) [4].

Pada ujian masuk Universitas Nasional menggunakan *Computer Based Test* (CBT). Namun penggunaan CBT memiliki kekurangan yaitu camaba harus datang ke kampus dan bergantian antar peserta untuk melakukan ujian karena memiliki keterbatasan tempat dan waktu. Tujuan penelitian ini adalah untuk mengembangkan sistem ujian masuk mahasiswa baru secara *online* menjadi aplikasi berbasis android, sehingga memudahkan bagi camaba dalam melakukan ujian serta lebih efisien baik dari segi waktu maupun tempat karena dapat dilakukan dimana saja asal terhubung internet serta tidak perlu bergantian dengan peserta lain untuk melakukan ujian..

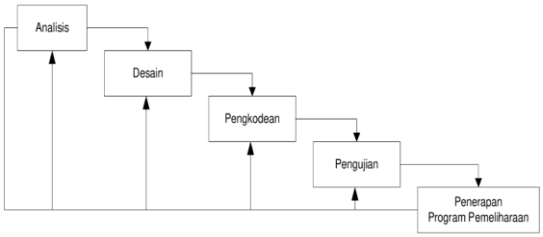
Android adalah sistem operasi yang dibuat oleh Google bersifat *open source* dan dirancang untuk perangkat mobile dan tablet [5]. Tidak hanya berbasis android, pada aplikasi ini menerapkan algoritma *Fisher-Yates* yang berfungsi mengacak soal dari database secara otomatis. Metode pengacakan ini bertujuan menghasilkan variasi soal dengan kriteria yang sama serta menghindari tindak kecurangan antar peserta ujian [6]. Algoritma *Fisher-Yates* memiliki tingkat efektivitas dalam pengacakan soal yang optimal, sehingga tidak terjadi pengulangan atau duplikasi pada soal yang sama.

Pada penelitian sebelumnya yang dilakukan oleh Arief Hasan, Supriadi, dan Zamzami pada tahun 2017 telah membuat aplikasi soal ujian online penerimaan mahasiswa baru dengan CBT serta menggunakan dbms mysql [1]. Kemudian penelitian yang dilakukan oleh Victor Asih, Andi Saputra, Ridho Taufiq Subagio membuat aplikasi ujian berbasi android menggunakan algoritma fisher-yates namun belum terdapat fitur daftar hanya terdapat login [2]. Penelitian berikutnya dilakukan oleh Laurentinus dan Riska Diana tahun 2018 membuat aplikasi penerimaan mahasiswa baru berbasis android dimana dalam aplikasi tersebut terdapat fitur daftar, login, dan mengerjakan soal ujian [3]. Selanjutnya penelitian yang dilakukan oleh Wawan Gunawan, dan Herry Derajad Wijaya tahun 2019 membuat aplikasi pembelajaran berbasis android dengan menggunakan firebase *realtime* database [5]. Penelitian yang dilakukan oleh Saiful Bukhori, dkk tahun 2016 dengan judul "*randomization of institutional testing program test of english as a foreign language using fisheryates algorithm*" membuat ujian *online* dengan membuat kategori soal [6].

Berdasarkan dari penelitian sebelumnya, aplikasi *Smart Entrance Exam* menggunakan sistem yang sudah berbasis android, dapat melakukan pendaftaran akun secara *online* via aplikasi, menggunakan metode *waterfall* dalam perancangan aplikasi, menggunakan algortima *Fisher-Yates* untuk mengacak soal ujian agar tiap user mendapat urutan soal yang berbeda, membuat fitur kategori soal, terintegrasi firebase *realtime* database untuk menampung data camaba secara *realtime*, integrasi database SQLite untuk menampung soal ujian masuk, serta hasil ujian yang langsung muncul ketika selesai ujian.

II. METODE PENELITIAN

Penelitian ini menggunakan metode pengembangan *waterfall* dengan pendekatan sistematis dan berurutan dalam pengembangan aplikasi. Metode *waterfall* terdiri dari beberapa tahapan pengembangan yaitu analisa, desain, pengkodean, pengujian dan pemeliharaan aplikasi [7].



Gambar 1. Metode Waterfall

A. Analisa

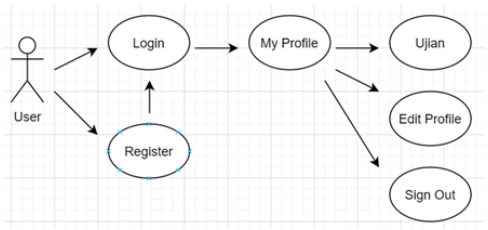
Tahap awal dalam metode *waterfall* yaitu analisa kebutuhan sistem dengan Analisa konsep dasar aplikasi kemudian pengumpulan data melalui penelitian, observasi, dan studi literatur yang terkait dengan penelitian. Adapun kebutuhan sistem yang diperlukan dalam membuat aplikasi ini adalah sebagai berikut.

1) Perangkat Keras: Perangkat keras pada penelitian kali ini terdiri dari Laptop (HP Pavilion) dengan spesikasi sebagai berikut CPU AMD Ryzen 5 3550H, RAM 8GB DDR4, HDD 1TB, NVDIA Geforce GTX 1050.

2) *Perangkat Lunak:* Perangkat lunak pada penelitian kali ini terdiri dari Windows 10 64bit (Sistem Operasi), dan Android Studio 3.6

B. Desain

Tahap kedua yaitu mercancang desain tampilan aplikasi yang akan dibuat mengenai tampilan *user interface*, dan kebutuhan material untuk aplikasi [8]. Dalam perancangan desain aplikasi menggunakan *use case* diagram untuk menggambarkan alur pada desain sistem aplikasi.



Gambar 2. Use case diagram rancangan aplikasi.

Pada gambar 2 adalah *use case* diagram penjelasan interaksi antara user dengan aplikasi yang dipresentasikan dengan langkah sederhana dimulai dengan proses *register* terlebih dahulu.

C. Pengkodean

Tahap ketiga adalah proses *coding* dengan menuliskan dalam bahasa pemrograman [9]. Kemudian mengimplementasikan desain yang sudah dibuat kedalam IDE Android studio, bahasa Java, serta firebase *realtime* database dan SQLite.

D. Pengujian

Tahap keempat melakukan pengujian yang telah dibangun apakah aplikasi sudah memenuhi kebutuhan yang diharapkan [10]. Pada pengujian aplikasi ini menggunakan metode *whitebox* dan *blackbox*.

E. Pemeliharaan

Tahap terakhir yaitu melakukan pemeliharaan dan perbaikan pada aplikasi berdasarkan *feedback* dari pengguna dan memperbaharuinya.

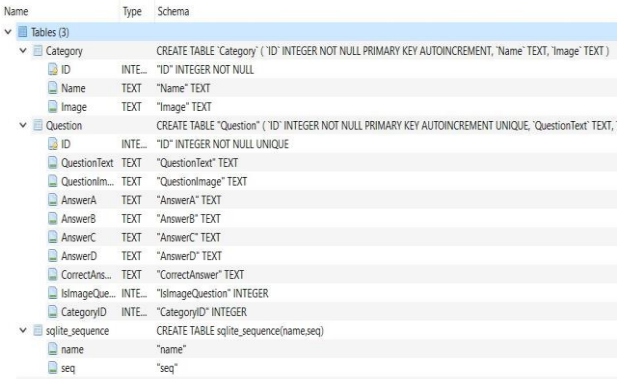
III. HASIL DAN PEMBAHASAN

Penelitian ini mengembangkan aplikasi *smart entrance exam* atau ujian masuk berbasis Android menggunakan algoritma *fisher-yates*. Pengembangan aplikasi menggunakan metode *waterfall* dengan langkah-langkah pelaksanaan penelitian 5 tahapan, yaitu Analisa, desain system, penulisan kode program, pengujian aplikasi, dan pemeliharaan. Aplikasi ini menggunakan firebase *realtime* database dan sqlite sebagai tempat penyimpanan data.

A. Struktur Database

Gambar 3. Struktur database firebase

Pada gambar 3 adalah struktur database firebase yang digunakan untuk minyimpan data user secara realtime. Terdiri dari beberapa data yaitu usename, password, email, jalur masuk, jenis kelami, nama lengkap, nomor handphone, password, program kuliah, program studi, status ujian, dan url_photo_profile yang digunakan untuk menyimpan foto profil yang diupload.

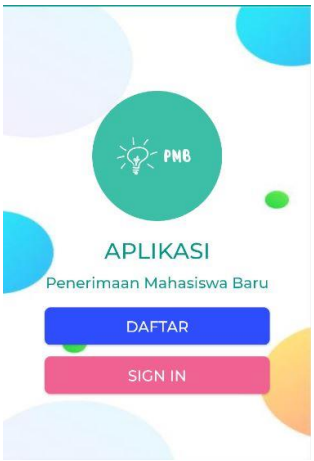


Gambar 4. Struktur database SQLite

Pada gambar 4 adalah struktur database SQLite yang berguna untuk menampung data soal dan jawaban. Terdapat 3 tabel yaitu c*ategory*, *question*, dan *sqlite_sequence*. Tabel *category* untuk menyimpan data kategori soal, kemudian table *question* menyimpan data soal, dan *sqlite_sequence* untuk menyimpan data jumlah kategori dan soal.

B. Desain Aplikasi

Pada desain aplikasi berisi tampilan program yang berkaitan dengan proses rancangan sistem yang telah dibuat [11].



Gambar 5. Menu Utama

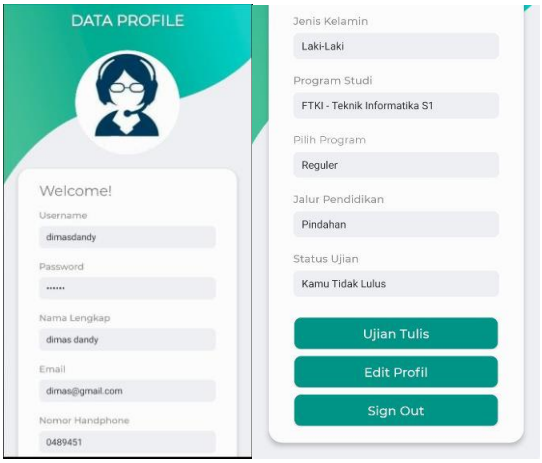
Gambar 5 merupakan tampilan menu utama. Pada menu utama terdapat dua item menu diantaranya adalah daftar dan *sign in.* Bila belum mempunyai akun pilih *menu* daftar, sedangkan bila sudah memiliki akun pilih *sign in* [12].

Gambar 6. Menu Daftar

Kemudian pada gambar 6 adalah menu daftar. Menampilkan *form* registrasi bagi calon mahasiswa baru yang ingin melakukan pendaftaran [13].

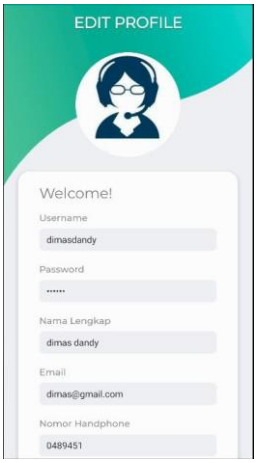
Gambar 7. Menu Sign in

Pada gambar 7 adalah menu *sign in*. Bagi user/camaba yang sudah memiliki akun dapat melakukan login.



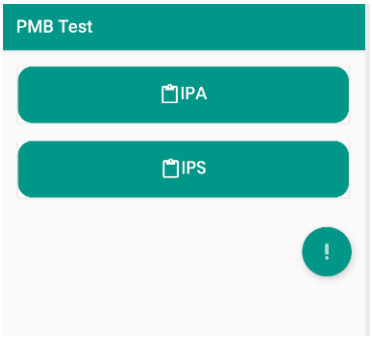
Gambar 8. Halaman Data Profile

Gambar 8 adalah halaman data profile yang berisikan data diri user. Kemudian terdapat 3 tombol yaitu ujian tulis untuk melakukan ujian, edit profil, dan *sign out.*



Gambar 9. Halaman Edit Profile

Gambar 9 adalah halaman edit profile untuk mengedit data diri user bila terjadi kesalahan penulisan ketika melakukan pendaftaran.



Gambar 10. Halaman Kategori

Kemudian pada gambar 10 adalah halaman kategori soal. Terdapat 2 kategori soal yaitu IPA dan IPS.

Gambar 11. Halaman Soal Ujian

Kemudian pada gambar 11 adalah halaman soal ujian. Pertanyaan yang muncul pada soal menggunakan

algoritma *fisher-yates* sehingga pertanyaan muncul secara acak [14].

Gambar 12. Halaman Hasil Ujian

Pada gambar 12 adalah halaman hasil ujian. Halaman ini menunjukkan evaluasi penilaian dari soal yang dikerjakan pada kategori yang dipilih [15]. Terdapat *result* berupa lulus/tidak ujian masuk, waktu yang ditempuh saat ujian, total soal, jawaban benar salah, dan soal yang tidak terjawab

C. Implementasi Algoritma Fisher-Yates

Algoritma *Fisher-Yates* adalah sebuah algoritma yang ditemukan oleh Ronald Fisher dan Frank Yates untuk mengubah urutan masukan yang dihasilkan secara acak [16]. Algoritma ini menghasilkan persentase keacakan yang tinggi dengan proses yang lebih komleks [17]. Kelebihan dari algoritma *fisher-yates* adalah kemudahan implementasi dan keluaran atau output yang benar [18]. Pemilihan algoritma *fisher*-yates bertujuan agar soal yang disajikan kepada calon mahasiswa baru dapat teracak dan tidak memiliki urutan soal yang sama dengan peserta lain [19]. Proses pengujian algoritma *fisher-yates* pada aplikasi dengan pengacakan 20 soal dari 40 soal yang tersedia digambarkan pada table sebagai berikut:

TABEL I
PENGUJIAN ALGORITMA FISHER-YATES

Range	Roll	Scratch	Result
		1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 16,17,18,19,20,21,22,23,24,25,26 , ,27,28,29,30,31,32,33,34,35,36,37, 38,39,40	
1-20	15	1,2,3,4,5,6,7,8,9,10,11,12,13,14, 16,17,18,19,20,21,22,23,24,25,26 , ,27,28,29,30,31,32,33,34,35,36,37, 38,39,40	15
1-19	19	1,2,3,4,5,6,7,8,9,10,11,12,13,14, 16,17,18,20,21,22,23,24,25,26, ,27,28,29,30,31,32,33,34,35,36,37, 38,39,40	19,15
1-18	37	1,2,3,4,5,6,7,8,9,10,11,12,13,14, 16,17,18,20,21,22,23,24,25,26, ,27,28,29,30,31,32,33,34,35,36, 38,39,40	37,19,15
1-17	25	1,2,3,4,5,6,7,8,9,10,11,12,13,14, 16,17,18,20,21,22,23,24,26, ,27,28,29,30,31,32,33,34,35,36, 38,39,40	25,37,19, 15
1-16	31	1,2,3,4,5,6,7,8,9,10,11,12,13,14, 16,17,18,20,21,22,23,24,26, ,27,28,29,30,32,33,34,35,36, 38,39,40	31,25,37, 19,15
1-15	3	1,2,4,5,6,7,8,9,10,11,12,13,14, 16,17,18,20,21,22,23,24,26, ,27,28,29,30,32,33,34,35,36, 38,39,40	3,31,25,3 7,19,15
1-14	36	1,2,4,5,6,7,8,9,10,11,12,13,14, 16,17,18,20,21,22,23,24,26, ,27,28,29,30,32,33,34,35,38,39,40	36,3,31,2 5,37,19,1 5
1-13	18	1,2,4,5,6,7,8,9,10,11,12,13,14, 16,17,20,21,22,23,24,26, ,27,28,29,30,32,33,34,35,38,39,40	18,36 3,31,25,3 7,19,15
1-12	11	1,2,4,5,6,7,8,9,10,12,13,14, 16,17,20,21,22,23,24,26, ,27,28,29,30,32,33,34,35,38,39,40	11,18,36 3,31,25,3 7,19,15
1-11	33	1,2,4,5,6,7,8,9,10,12,13,14, 16,17,20,21,22,23,24,26, ,27,28,29,30,32,34,35,38,39,40	33,11,18, 36 3,31,25,3 7,19,15
1-10	28	1,2,4,5,6,7,8,9,10,12,13,14, 16,17,20,21,22,23,24,26, ,27,29,30,32,34,35,38,39,40	28,33,11, 18,36 3,31,25,3 7,19,15
1-9	16	1,2,4,5,6,7,8,9,10,12,13,14, 17,20,21,22,23,24,26, ,27,29,30,32,34,35,38,39,40	16,28,33, 11,18,36 3,31,25,3 7,19,15
1-8	20	1,2,4,5,6,7,8,9,10,12,13,14, 17,21,22,23,24,26,27, 29,30,32,34,35,38,39,40	20,16,28, 33,11,18, 36 3,31,25,3 7,19,15
1-7	26	1,2,4,5,6,7,8,9,10,12,13,14, 17,21,22,23,24, ,27,29,30,32,34,35,38,39,40	26,20,16, 28,33,11, 18,36 3,31,25,3 7,19,15
1-6	2	1,4,5,6,7,8,9,10,12,13,14, 17,21,22,23,24, ,27,29,30,32,34,35,38,39,40	2,26,20,1 6,28,33,1 1,18,36 3,31,25,3 7,19,15
1-5	27	1,4,5,6,7,8,9,10,12,13,14, 17,21,22,23,24, 29,30,32,34,35,38,39,40	27,2,26,2 0,16,28,3 3,11,18,3 6 3,31,25,3 7,19,15
1-4	10	1,4,5,6,7,8,9,12,13,14, 17,21,22,23,24, 29,30,32,34,35,38,39,40	10,27,2,2 6,20,16,2 8,33,11,1 8,36 3,31,25,3 7,19,15

1-3	4	1,5,6,7,8,9,12,13,14, 17,21,22,23,24, 29,30,32,34,35,38,39,40	4,10,27,2 ,26,20,16 ,28,33,11 ,18,36 3,31,25,3 7,19,15
1-2	9	1,5,6,7,8,12,13,14, 17,21,22,23,24, 29,30,32,34,35,38,39,40	9,4,10,27 ,2,26,20, 16,28,33, 11,18,36 3,31,25,3 7,19,15
	24	1,5,6,7,8,12,13,14, 17,21,22,23,29,30,32,34,35,38,39 ,40	24,9,4,10 ,27,2,26, 20,16,28, 33,11,18, 36 3,31,25,3 7,19,15

Berdasarkan tabel I hasil pengacakan soal oleh algoritma *fisher-yates* dimana didapatkan soal terpilih sebanyak 20 buah dari 40 buah soal diantaranya adalah soal nomor 24,9,4,10,27,2,26,20,16,28,33,11,18,36 3,31,25,37,19,15.

TABEL III
PENGUJIAN ALGORITMA FISHER-YATES

Peng-ujian	Hasil Pengacakan Soal
2	17,12,35,40,34,25,27,10,29,16,1,7,8,24,20,5,31,28,13,6
3	32,3,37,23,8,38,4,17,15,35,14,31,19,26,5,9,6,33,20,7
4	4,40,10,34,27,2,9,36,32,12,33,1,35,13,38,17,28,20,19,37
5	20,10,23,13,7,38,37,4,19,11,6,36,40,28,14,26,30,31,34,35

Pada table II dilakukan kembali pengujian pengacakan soal, didapati urutan soal yang berbeda-beda dengan persentase 100% tanpa terjadi pengulangan atau duplikasi pada soal.

Pengujian Algoritma Fisher Yates

Pengujian Ke-	Hasil Random Nomor Soal	Nomor Yang Mengalami Duplikasi	Persentase
496	23 11 2 31 39 22 26 14 20 30 16 37 33 13 21 7 24 12 4 3	NULL	100
497	29 16 14 24 5 3 40 12 6 7 35 18 13 28 8 22 20 9 30 38	NULL	100
498	37 11 17 8 12 21 22 15 16 23 3 26 13 36 18 1 19 34 7 32	NULL	100
499	31 1 3 27 24 13 34 29 26 6 25 18 30 33 5 8 20 21 7 40	NULL	100
500	30 2 12 26 22 27 32 31 39 20 4 11 35 19 33 38 18 34 13 7	NULL	100

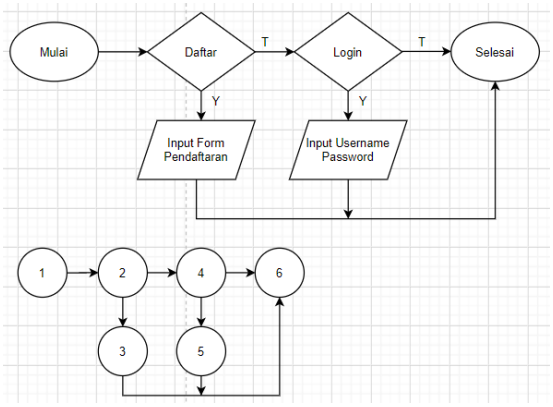
Gambar 13. Data Hasil Uji Pengacakan Soal

Pada gambar 13 adalah data hasil pengujian *random* soal menggunakan algoritma *fisher-yates* tanpa duplikasi dengan persentase 100%.

D. Pengujian Metode Whitebox

Metode whitebox adalah pengujian software yang memberikan hasil struktur data dari kontrol program melalui *flowgraph* dan *flowchart* [20]. *Flowgraph* digunakan pada tahapan pengujian yang berfokus pada penggambaran aliran kontrol dari sebuah program sedangkan *flowchart* digunakan untuk menggambarkan logika dari program sedangkan. Perhitungan whitebox menguji dengan menghitung 3 parameter yaitu *cyclomatic complexity*, *region*, dan *independent path*. *Cyclomatic complexity* menghitung jumlah path dari *edge* dan *node* pada flowgraph. Region berdasarkan *predicate node* yaitu jumlah *if else* pada *flowgraph*. Langkah terakhir adalah menyusun path pada *flowgraph* berdasarkan jumlah path yang dihitung pada *cyclomatic complexity*.

1) Flowchart dan Flowgraph Menu Utama:

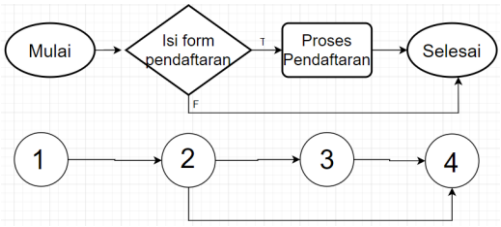


Gambar 14. Flowchart dan flowgraph Menu Utama

TABEL IIIII
PENGUJAIN WHITEBOX PADA MENU UTAMA

Menghitung *Cyclomatic Complexity* dari *Edge* dan *Node*	Menghitung berdasarkan *Predicate Node* (P), dimana P = 2.	Path pada *flowgraph* di atas
E = 7 **N = 6** **V(G) = E – N + 2** **= 7 – 6 + 2** **= 3** **Jadi, jumlah Path = 3 Path**	V(G) = P + 1 = 2 + 1 = 3 Jadi *Region (R)* pada *flowgraph* = 3	Path 1 = 1-2-3-6 Path 2 = 1-2-4-5-6 Path 3 = 1-2-4-6

2) Flowchart dan Flowgraph Menu Daftar:

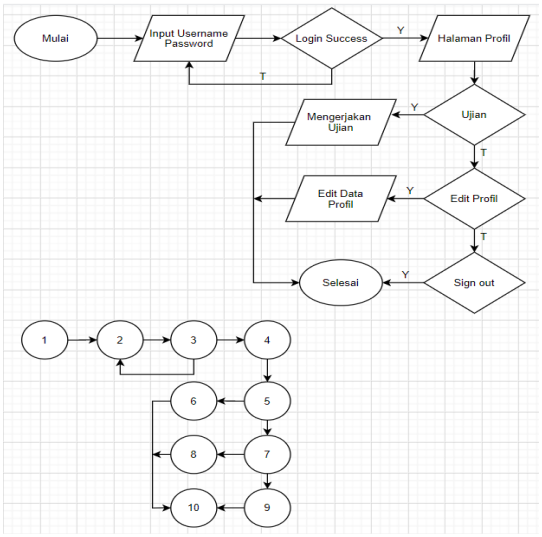


Gambar 15. Flowchart dan flowgraph Menu Daftar

TABEL IVV
PENGUJAIN WHITEBOX PADA MENU DAFTAR

Menghitung *Cyclomatic Complexity* dari *Edge* dan *Node*	Menghitung berdasarkan *Predicate Node* (P), dimana P = 1.	Path pada *flowgraph* di atas
E = 4 **N = 4** **V(G) = E – N + 2** **= 4 – 4 + 2** **= 2** **Jadi, jumlah Path = 2 Path**	V(G) = P + 1 = 1 + 1 = 2 Jadi *Region (R)* pada *flowgraph* = 2	Path 1 = 1-2-3-4 Path 2 = 1-2-4

3) Flowchart dan Flowgraph Menu Login:

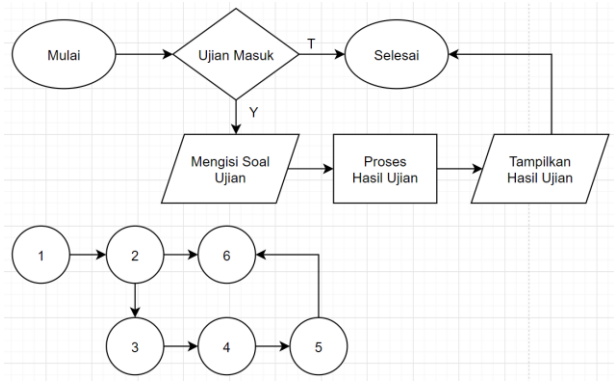


Gambar 16. *Flowchart* dan *flowgraph* Menu Login

TABEL V
PENGUJAIN WHITEBOX PADA MENU LOGIN

Menghitung *Cyclomatic Complexity* dari *Edge* dan *Node*	**Menghitung berdasarkan *Predicate Node* (P), dimana P = 5.**	**Path pada *flowgraph* di atas**
E = 13 **N = 10** **V(G) = E – N + 2** **= 13 – 10 + 2** **= 5** **Jadi, jumlah Path = 5 Path**	V(G) = P + 1 = 4 + 1 = 5 Jadi Region (R) pada flowgraph = 5	Path 1 = 1-2-3-2-3-4-5-7-9-10 Path 2 = 1-2-3-4-5-6-10 Path 3 = 1-2-3-4-5-7-8-10 Path 4 = 1-2-3-4-5-7-9-10 Path 5 = 1-2-3-2-3-4-5-6-10

4) Flowchart dan Flowgraph Menu Ujian:



Gambar 17. *Flowchart* dan *flowgraph* Menu Ujian

TABEL VI
PENGUJAIN WHITEBOX PADA MENU UJIAN

Menghitung *Cyclomatic Complexity* dari *Edge* dan *Node*	**Menghitung berdasarkan *Predicate Node* (P), dimana P = 1.**	**Path pada *flowgraph* di atas**
E = 6 **N = 6** **V(G) = E –** **N + 2** **= 6 – 6 + 2** **= 2** **Jadi, jumlah Path = 2 Path**	V(G) = P + 1 = 1 + 1 = 2 Jadi *Region (R)* pada *flowgraph* = 2	Path 1 = 1-2-3-2-3-4-5-6 Path 2 = 1-2-6

5) Hasil Pengujian Whitebox:

TABEL VII
HASIL PENGUJIAN WHITEBOX

Flowgraph	**Cyclomatic Complex city (cc)**	**Region (R)**	**Independen Path**
Menu *Home*	3	3	3
Menu Daftar	2	2	2
Menu Login	5	5	5
Menu Ujian	2	2	2
Jumlah	12	12	12

Pada tabel VII pengujian metode *whitebox* yang telah dilakukan, mendapatkan jumlah *cyclomatic complexity*, *region*, dan *independent path* yang sama yaitu 12, berarti logika dan alur program yang dibuat sudah sesuai yang diharapkan dan tidak perlu melakukan *compile* ulang.

E. Pengujian Metode Blackbox

Metode *blackbox* adalah pengujian terhadap fungsi operasi dan kegunaan aplikasi berdasarkan hasil yang didapat sudah sesuai yang diharapkan atau belum. Pada tabel VII adalah hasil pengujian aplikasi dengan metode blackbox.

TABEL VII
PENGUJIAN BLACKBOX

Pengujian	**Bentuk Pengujian**	**Hasil yang diharapkan**	**Hasil pengujian**
Menu Utama	Masuk ke halaman utama	Menampilkan menu daftar dan login	Berhasil
Menu Daftar	Menekan tombol daftar dan menginput data form	Menampilkan formulir pendaftaran dan data berhasil diinput serta tersimpan dalam database	Berhasil
Menu Login	Login dengan username password yang telah didaftarkan	Menampilkan halaman *sign in* dan berhasil masuk ke menu data profil	Berhasil
Menu Ujian	Melakukan uji coba ujian masuk	Menampilkan soal ujian secara acak tanpa	Berhasil

		duplikasi soal, dan menampilkan hasil akhir	

IV. KESIMPULAN

Setelah melakukan pengujian dan analisa Aplikasi *Smart Entrance Exam* Berbasis Android dengan Algoritma *Fisher-Yates* dapat digunakan sebagai media untuk ujian masuk pada proses penerimaan mahasiswa baru sesuai dengan pengujian yang telah dilakukan dengan standar ANSI yaitu *whitebox* dan *blackbox*.

Pada proses pengujian algoritma *fisher-yates* sebanyak lima ratus data pengujian pengacakan soal, didapati urutan soal yang berbeda-beda dengan persentase 100% tanpa terjadi pengulangan atau duplikasi pada soal. Kemudian pengujian dari sisi sistem menggunakan metode *whitebox* mendapatkan jumlah *cyclomatic complexity*, *region*, dan *independent path* yang sama yaitu 12, berarti logika dan alur program yang dibuat sudah sesuai yang diharapkan dan tidak perlu melakukan *compile* ulang.

Pada pengujian dari sisi user menggunakan metode *blackbox* didapati hasil sesuai yang diharapkan dimana pengujian pada menu yang dijalankan dapat berhasil. Dapat disimpulkan aplikasi ini telah berjalan sesuai yang diharapkan.

DAFTAR PUSTAKA

[1] M. A. Hasan, S. Supriadi, and Z. Zamzami, "Implementasi Algoritma Fisher-Yates Untuk Mengacak Soal Ujian Online Penerimaan Mahasiswa Baru (Studi Kasus : Universitas Lancang Kuning Riau)," *J. Nas. Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 291–298, 2017, doi: 10.25077/teknosi.v3i2.2017.291-298.

[2] V. Asih, A. Saputra, R. T. Subagio, U. Catur, I. Cendekia, and K. Cirebon, "Penerapan Algoritma Fisher YatesShuffle Untuk Aplikasi Ujian Berbasis Android," *J. Digit*, vol. 10, no. 1, pp. 59–70, 2020.

[3] Laurentinus and R. Diana, "Implementasi Algoritma Fisher-Yattes Pada Aplikasi Penerimaan Mahasiswa Baru Berbasis Android," *J. SISFOKOM*, vol. 07, no. September, pp. 163–173, 2018.

[4] F. P. Juniawan *et al.*, "Pengacakan Soal Ujian Penerimaan Polri," *J. Telemat.*, vol. 1, no. 1, pp. 1–13, 2019.

[5] W. Gunawan and H. D. Wijaya, "An Application of Multimedia for Basic Arabic Learning Using FisherYates Shuffle Algorithm on Android Based," *Sch. Bull.*, vol. 9771, pp. 347–355, 2019, doi: 10.21276/sb.2019.5.7.6.

[6] S. Bukhori, N. Dania Putri, and A. Andrianto, "Randomization of Institutional Testing Program Test of English as a Foreign Language Using Fisher Yates Algorithm," *Res. J. Appl. Sci. Eng. Technol.*, vol. 13, no. 1, pp. 57–63, 2016, doi: 10.19026/rjaset.13.2890.

[7] R. R. C. Putra and T. Sugihartono, "Penerapan Algoritma Fisher-Yates Shuffle pada Computer Based Test Ujian Sekolah di SMKN 1 Payung," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 18, no. 2, pp. 276–283, 2019, doi: 10.30812/matrik.v18i2.399.

[8] B. Subaeki and D. Ardiansyah, "Implementasi Algoritma Fisher - Yates Shuffle Pada Aplikasi Multimedia Interaktif Untuk Pembelajaran Tenses Bahasa Inggris," J. Infotronik, vol. 2, no. 1, pp. 67–74, 2017.

[9] A. Riyadi and E. Kartikadarma, "Penerapan Algoritma Fisher-Yates Shuffle Pada Sistem Kuis," *J. Tek. Inform. Kaputama*, vol. 4, no. 207, pp. 1–8, 2020.

[10] Ekojono, D. A. Irawati, L. Affandi, and A. N. Rahmanto, "Penerapan Algoritma Fisher-Yates Pada Pengacakan Soal Game Aritmatika," *Pros. SENTIA 2017 – Politek. Negeri Malang*, vol. 9, pp. 101–106, 2017.

[11] J. M. Hudin and E. Wati, "Penerapan Metode Fisher Yates Shuffle Untuk Sistem Informasi Ujian Online Pada Smkn P 1 Sukaraja," *Sniptek*, pp. 161–164, 2016.

[12] S.D. Siregar, D. Permatasari, A. Hutapea, D. Kinarwan," Implementation of Fisher Yates Shuffle Algorithm In Making Learning Animations" *J. Mantik*, vol. 4, no.2 August, pp. 1291–1298, 2020.

[13] A. Mulyadi and T. D. Purwanto, "Aplikasi Simulasi Toefl (Test of English As a Foreign Language) Berbasis Android Memanfaatkan Algoritma Fisher Yates Shuffle (Studi Kasus : Lc Universitas Bina Darma)," pp. 58–77, 2019.

[14] W. A. Rohmah, A. Asriyanik, and W. Apriyandari, "Implementation of the Algorithm Fisher Yates Shuffle on Game Quiz Environment," *J. Informatics Telecommun. Eng.*, vol. 4, no. 1, pp. 161–172, 2020, doi: 10.31289/jite.v4i1.3863.

[15] T. F. Revano, M. B. Garcia, B. G. M. Habal, J. O. Contreras, and J. B. R. Enriquez, "Logical guessing riddle mobile gaming application utilizing fisher yates algorithm," *2018 IEEE 10th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manag. HNICEM 2018*, no. August 2019, 2019, doi: 10.1109/HNICEM.2018.8666302.

[16] I. Maryono, W. B. Zulfikar, and R. Kariadinata, "The implementation of fisher yates shuffle on aljabar learning media based on hybrid application," *MATEC Web Conf.*, vol. 197, pp. 1–5, 2018, doi: 10.1051/matecconf/201819701006.

[17] C. Aishwarya and J. R. Beny, "Novel Architecture for Data – Shuffling Using Enhanced Fisher Yates Shuffle Algorithm," *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 1, no. 6, pp. 387–390, 2015, doi: 10.4010/2016.1121.

[18] A. Olu, "A Simulated Enhancement of Fisher-Yates Algorithm for Shuffling in Virtual Card Games using Domain-Specific Data Structures," *Int. J. Comput. Appl.*, vol. 54, no. 11, pp. 975–8887, 2012, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.114&rep=rep1&type=pdf.

[19] F. Ahmad, "Penerapan Algoritma Fisher Yates Shuffle dan Linear Congruent Method Pada Simulasi Ujian Toefl Berbasis Android," *J. Ris. Komput.*, vol. 5, no. 1, pp. 653–660, 2018.

[20] R. Priantama and Y. Priandani, "Implementasi Algoritma Fisher Yates Untuk Pengacakan Soal Pada Aplikasi Mobile Learning Kuis Fiqih Berbasis Android," *Nuansa Inform.*, vol. 13, no. 2, p. 40, 2019, doi: 10.25134/nuansa.v13i2.1951.