PENGUJIAN BLACK BOX APLIKASI INKANTEEN BERBASIS EQUIVALENCE PARTITIONING

Trevy Jonatya Novella[1], Ahlijati Nuraminah [2]

[1,2]Program Studi Ilmu Komputer
Sekolah TinggiIlmu Manajemen dan Ilmu Komputer ESQ
Jl. TB Simatupang, RT.3/RW.3, Cilandak Timur., Jakarta 12560
Email: [1]t.jonatya.n@students.esq.ac.id, [2]ahlijati.nuraminah@esq.ac.id

(Naskah masuk: dd mmm yyyy, diterima untuk diterbitkan: dd mmm yyyy)
(1 baris kosong, 10pt)

Abstrak

Selama masa pandemi *covid-19* banyak aplikasi yang dirancang untuk meminimalisir terjadinya kerumunan salah satunya aplikasi kantin daring. Aplikasi InKanteen *mobile* dirancang untuk mempermudah proses jual beli di kantin. Sebelum aplikasi dapat digunakan oleh pengguna, harus melalui proses pengujian terlebih dahulu. Berdasarkan data pengujian aplikasi InKanteen tahap pertama sebanyak 63.2% *test case passed* dan 36.8% *test case failed*. Setelah dilakukan proses perbaikan oleh *developer*, akan dilakukan pengujian lanjutan menggunakan *black box testing* dengan metode *equivalence partitioning*. Pengujian *equivalence patitioning* adalah pengujian fungsionalitas *input* atau *output* pada perangkat lunak dan kondisi *input* menggambarkan kumpulan keadaan yang *valid* atau *invalid.* Pengujian ini bertujuan untuk mengetahui seberapa tinggi tingkat keberhasilan aplikasi InKanteen dengan metode *equivalence partitioning*. Setelah dilakukan proses pengujian hasil yang didapat dari perhitungan *software metrics testing* sebesar 94% *test case passed* dan sebesar 6% *test case failed*. Rekomendasi yang diberikan untuk kegagalan 6% tersebut yaitu menambahkan validasi pada fitur *tenant* bagian *password* agar *input*an sesuai *requirement* sistem. Berdasarkan hasil pengujian dapat dikatakan aplikasi InKanteen sudah layak digunakan oleh pengguna.

Kata kunci: *mobile apps, quality assurance, blacbox testing, equivalence partitioning testing, software testing metrics*

INKANTEEN APPLICATION BLACK BOX TESTING BASED ON EQUIVALENCE PARTITIONING

Abstract

During the COVID-19 pandemic, many applications were designed to minimize crowds, one of which was the online canteen application. The InKanteen mobile application is designed to simplify the buying and selling process in the canteen. Before the application can be used by users, it must go through a testing process first. Based on the test data of the first InKanteen application as much as 63.2% test case passed and 36.8% test case failed. After the improvement process has been carried out by the developer, further testing will be carried out using black box testing with the equivalence partitioning method. Equivalence patitioning testing is testing the input or output functionality of the software and the input conditions describe a set of valid or invalid states. This test aims to determine how high the success rate of the InKanteen application with the equivalence partitioning method is. After the testing process, the results obtained from the calculation of software metrics testing are 94% test cases passed and 6% test cases failed. The recommendation given for the 6% failure is to add validation to the password section of the tenant feature so that the input matches the system requirements. Based on the test results, it can be said that the InKanteen application is suitable for use by users

Keywords: *mobile apps, quality assurance, blacbox testing, equivalence partitioning testing, software testing metrics*

1. PENDAHULUAN

Dengan adanya kemajuan teknologi saat ini kinerja manusia dapat lebih efisien dan efektif. Dikarenakan hal tersebut bermunculan aplikasi yang dirancang untuk memudahkan kehidupan manusia. Kemajuan teknologi ini terjadi hampir di semua aspek kehidupan, salah satunya adalah kemajuan teknologi pada bidang bisnis.

Berdasarkan data (APJJI, 2022), sebanyak 89.03% penduduk Indonesia memakai telepon genggam. Maka dari itu PT Kanovasi Inmaya Nusantara melihat peluang besar untuk memanfaatkan teknologi mobile untuk membuat aplikasi kantin sekolah daring. PT Kanovasi Inmaya Nusantara melihat proses jual beli di kantin sekolah yang dilakukan secara konvensional yaitu siswa datang ke kantin membeli makanan atau minuman, kemudian melakukan pembayaran secara langsung kepada pengurus kantin dapat menimbulkan beberapa hal di zaman pandemi saat ini contohnya menyebabkan siswa berkerumun, dan media penyebaran virus melalui uang kertas.

InKanteen akan memberikan pengalaman kantin baru untuk ekosistem sekolah, mengubah kantin sekolah menjadi tujuan baru dan menarik. Sebelum aplikasi dapat digunakan oleh siswa atau tenant perlu dilakukannya sebuah pengujian aplikasi atau *testing*. Pengujian perangkat lunak merupakan elemen kritis dalam menentukan kualitas suatu perangkat lunak.

Pada pengujian aplikasi InKanteen tahap pertama oleh penjamin kualitas sistem ditemukan beberapa kegagalan fitur, dengan persentase 63.2% *passed* dan 36.8% *failed*. Setelah dilakukan perbaikan oleh *developer* perlu dilakukan pengujian ulang aplikasi InKanteen menggunakan metode yang dapat digunakan untuk mengukur penerimaan dan penggunaan teknologi yaitu metode *Black Box Testing* dengan teknik *Equivalence Partitioning*, untuk mengetahui tingkat keberhasilan pengujian aplikasi InKanteen menggunakan metode *Black box testing* dengan teknik *Equivalence Partitioning*.

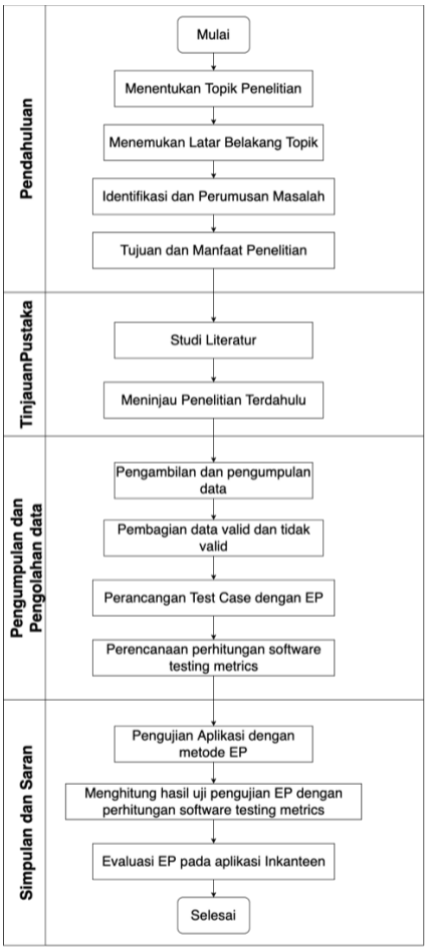
Pada penelitian ini akan menggunakan aplikasi Inkanteen *merchant* dengan fitur *Login, Menu, Reports, Order* dan *Tenant.*

Equivalence Partitioning adalah metode pengujian *black box* yang membagi domain *input* dari suatu program ke dalam kelas-kelas data dan akan diturunkan menjadi kasus uji(Pressman, 2001). Menurut (Jovanovic, 2009) Kelas ekuivalen yaitu kumpulan status *valid* atau tidak *valid* untuk kondisi *input*, dan dapat didefinisikan dengan beberapa cara berikut:

a. Kondisi *input* menentukan *range* : Satu kelas ekuivalensi *valid* dan dua tidak *valid* didefinisikan, contoh dari *input* penggunaan *range* yaitu jangkauan bulan adalah 1 sampai 12 yang mewakili Januari sampai Desember, sehingga *range input*-an yang dikatakan *valid* adalah *input*-an 1 sampai 12, dan *range input*-an yang dikatakan tidak *valid* adalah *input*-an <1, >12, atau *blank*.
b. Kondisi *input requires a specific value*: satu kelas ekivalensi yang *valid* dan dua tidak *valid* didefinisikan, contoh dari penggunaan nilai tertentu adalah ketika *password* mempunyai *requirement* sebanyak enam digit *alphanumeric string*, maka *input condition specific value* mewajibkan *password* terdiri dari enam *character string*.

2. METODOLOGI PENELITIAN

Pada Gambar 1 merupakan representasi dari tiap-tiap tahapan yang dilakukan dalam pelaksanaan penelitian ini.



Gambar 1 Alur Penelitian

Pada tahap pendahuluan dilakukan pencarian topik pembahasan dengan cara menggali permasalahan disekitar yang dapat diselesaikan. Setelah menentukan topik kemudian menemukan latar belakang, mengidentifikasi permasalahan, menjelaskan tujuan dan manfaat yang diperoleh dari hasil penelitian.

Pada tahap tinjauan pustaka berfokus kepada studi literatur ke berbagai buku, jurnal, skripsi, dan *website*. Studi yang dilakukan guna mempelajari pengetahuan lebih lanjut dari penelitian terkait dan menjadi referensi tulisan. Studi literatur yang dilakukan meliputi berbagai materi tentang *Quality Assurance, Testing, Black box Testing, Equivalence Partitioning*.

Tahap pengumpulan dan pengolahan data berasal dari dokumentasi *project* InKanteen yang didapatkan dari *project manager* aplikasi InKanteen. Dokumentasi *project* ini mencakup fitur yang akan ada di aplikasi Inkanteen, *flow* dari setiap *action* aplikasi Inkanteen, dan beberapa *requirement* terkait fitur aplikasi Inkanteen. Pada tahapan ini didapatkan alur kerja dan *behavior* yang diharapkan pada aplikasi InKanteen.

Pembagian data uji *valid* dan tidak *valid* akan dibagi berdasarkan *requirements* aplikasi pada figma, yang dimana data *valid* adalah seluruh inputan atau *behavior* perangkat lunak seperti yang diharapkan pengguna dan harus dapat diakses oleh pemangku kepentingan. Sedangkan untuk data tidak valid adalah seluruh inputan atau *behavior* perangkat lunak yang tidak diharapkan dan dapat mengeluarkan pesan *error*. Setiap kondisi untuk perhitungan hasil akhir akan menggunakan perhitungan *software testing metrics* kategori *process metrics*.

Pada tahap simpulan dan saran, simpulan menunjukan apakah data yang digunakan cocok dengan metode tersebut dan memberikan saran perbaikan untuk penelitian selanjutnya.

Analisis data yang digunakan adalah *Software metrics* yaitu memberikan kriteria dan pengukuran objektif untuk pengambilan keputusan manajemen. *Software metrics* pengukuran yang melibatkan angka agar dapat meningkatkan proses pengembangan perangkat lunak dan meningkatkan semua aspek pengelolaan proses(Mladenova, 2020).

Terdapat beberapa cara untuk mengukur kualitas hasil akhir produk menggunakan *process metrics*(Bahar et al., 2019);

Tabel 1 *Process Metrics*

Metrics	***Formula***	**Deskripsi**
Passed Test Cases Percentage	$\frac{number\ passed\ tc}{total\ number\ tc} x100$	Metrik yang menunjukkan persentase kasus uji yang lulus
Failed Test Cases Percentage	$\frac{number\ failed\ tc}{total\ number\ tc} x100$	Metrik yang menunjukkan persentase kasus uji yang tidak lulus
Test Execute Coverage	$\frac{number\ tc\ execute}{total\ tc\ written} x100$	Metrik yang menunjukkan persentase kasus uji yang dijalankan
Test not Execute Coverage	$\frac{tc\ not\ execute}{total\ tc\ written} x100$	Metrik yang menunjukkan persentase kasus uji yang tidak dijalankan

3. HASIL DAN PEMBAHASAN

3.1 Kebutuhan Fungsional

Pada aplikasi ini terdapat kebutuhan fungsional dari masing masing fitur antara lain

a. Fitur *Login*

Fungsi ini digunakan untuk membatasi pengguna yang dapat mengakses sistem sesuai dengan perannya masing-masing. Pengguna yang tidak sah akan ditolak. Jika pengguna memasukan *login* yang salah, maka pesan *error* akan dikirim sesuai dengan *response error* yang didapat. Pada halaman ini terdapat dua *input* yaitu *email* dan *password* dengan *requirement* format *e-mail* wajib *valid email address* contohnya email@mail.com email@mail.com dan format password dapat berupa angka dan huruf, minimal 6 *characters*.

b. Fitur *Order*

Fungsi ini digunakan untuk pengguna melihat dan mengatur pesanan masuk yang akan dikategorikan menjadi *order review* yaitu pesanan yang harus disiapkan pada saat itu, *order schedule* yaitu pesanan yang harus disiapkan pada waktu yang sudah ditentukan, dan *order history* yaitu seluruh *list* pesanan dengan status berhasil atau gagal.

c. Fitur *Menu*

Fungsi ini digunakan untuk pengguna melihat, menambah, merubah, menghapus menu yang ada pada *tenant* tersebut. Pada fitur ini juga terdapat fitur untuk menambah, menghapus, dan mengatur posisi dari kategori yang nantinya akan dilihat oleh pembeli. Adapun beberapa *requirement* pada fitur *menu* yaitu format nama *menu minimal* 3 *characters*, format harga *menu* hanya dapat *input* angka, format deskripsi *menu* dapat *input* angka, huruf dan *symbol*, format *estimated time* hanya dapat *input* angka

d. Fitur *Reports*

Fungsi ini digunakan untuk pengguna melakukan penarikan dana hasil penjualan. Pada fungsi ini pula pengguna dapat melihat riwayat penarikan dana yang sudah dilakukan dan pengguna dapat menambah metode penarikan dana dapat berupa Bank ataupun *e-wallet.* Adapun beberapa *requirement* pada fitur *reports* yaitu Format nominal *withdrawal* hanya dapat *input* angka, format *Account name* hanya dapat *input* huruf, format *bank number* hanya dapat input angka, format *verification code* dapat *input* angka dan huruf.

e. Fitur *Tenant*

Fungsi ini digunakan untuk pengguna merubah detail toko seperti nama toko, nomor telepon, *email*, dan *password* akun. Pengguna dapat mengatur Bahasa pada fitur *tenant*, dan dapat melihat *privacy policy, term and condition*, dan *user guide*. Adapun beberapa *requirement* fitur *tenant* yaitu format *outlet name* dapat *input* angka, huruf, dan simbol dengan *minimal* 3 *characters.*

3.2 Penyusunan *Test Case* dan Hasil Pengujian

Berikut adalah penyusunan *test case* berdasarkan *test case equivalence partition*. *Test case* dipisah sesuai dengan fitur aplikasi InKanteen yaitu Fitur *Login, Menu, Order, Reports*, dan *Tenant*.

a. *Test Case Login*

Pada pengujian *test case login* Tabel 2 terdapat sebanyak 4 *test case* pengujian yang terbagi menjadi 1 *test case valid* dan 4 *test case invalid.*

Tabel 2 *Test Case* Fitur *Login*

TC ID	*Test Case Name*	*Expected Result*	*Category*	Hasil
1	*Login*	*Success Login* dan *redirect to homepage*	*Valid*	*PASSED*
2	*Login with wrong format email*	*Popup* Informasi *email* atau *password* salah	*Invalid*	*PASSED*
3	*Login with password less than 6 char*	*Popup* Informasi *email* atau *password* salah	*Invalid*	*PASSED*
4	*Login with wrong password*	*Popup Informasi email* atau *password* salah	*Invalid*	*PASSED*

Setelah dilakukan *test* pada fitur *login* didapatkan hasil sebanyak 4 *test case PASSED* dan 0 *test case FAILED.*

b. *Test Case Menu*

Pada pengujian *test case menu* Tabel 3, terdapat sebanyak 9 *test case* pengujian yang terbagi menjadi 3 *test case valid* dan 6 *test case invalid.*

Tabel 3 *Test Case Fitur Menu*

TC ID	*Test Case Name*	*Expected Result*	*Category*	Hasil
1	Tambah *menu*	*Success* tambah *menu*	*Valid*	*PASSED*
2	Nama *menu* terdiri dari *2 characthers*	*Error validate* nama *menu min. 3 Characters*	*Invalid*	*PASSED*
3	Nama *menu* terdiri dari 40 *characthers*	*Error validate* nama *menu max. 30 Characters*	*Invalid*	*PASSED*
4	Harga *di-input alphabet*	Harga tidak dapat di-*input*	*Invalid*	*PASSED*
5	Harga *di-input* minus	Harga tidak dapat di-*input*	*Invalid*	*PASSED*
6	*Add* kategori	*Add new Categori*	*Valid*	*PASSED*
7	*Add* kategori *blank*	*Error the name field is required*	*Invalid*	*PASSED*
8	*Search menu*	*Menu* muncul sesuai *input*	*Valid*	*PASSED*
9	*Search menu random*	*Menu* tidak akan muncul	*Invalid*	*PASSED*

Setelah dilakukan *test* pada fitur *menu* didapatkan hasil sebanyak 9 *test case PASSED* dan 0 *test case FAILED.*

c. *Test Case Order*

Pada pengujian *test case order* Tabel 4, terdapat sebanyak 6 *test case* pengujian yang terbagi menjadi 4 *test case valid* dan 2 *test case invalid.*

Tabel 4 *Test Case* Fitur *Order*

TC ID	*Test Case Name*	*Expected Result*	*Category*	Hasil
1	*Order review*	*Status order* menjadi *inprogress*	*Valid*	*PASSED*
2	*Order review - cancel*	*Status order* menjadi *Failed*	*Valid*	*PASSED*
3	*Order review – cancel blank*	*Alasan pembatalan required*	*Invalid*	*PASSED*
4	*Order Schedule*	*Status order* menjadi *inprogress*	*Valid*	*PASSED*
5	*Order schedule - cancel*	*Status order* menjadi *Failed*	*Valid*	*PASSED*
6	*Order schedule – cancel blank*	Alasan *pembatalan required*	*Invalid*	*PASSED*

Setelah dilakukan *test* pada fitur *order* didapatkan hasil sebanyak 6 *test case PASSED* dan 0 *test case FAILED*.

d. *Test Case Repots*

Pada pengujian *test case repots* , terdapat sebanyak 7 *test case* pengujian yang terbagi menjadi 1 *test case valid* dan 6 *test case invalid.*

Tabel 5 *Test Case* Fitur *Reports*

TC ID	*Test Case Name*	*Expected Result*	*Category*	Hasil
1	*Account bank*	*Success add new method*	*Valid*	*PASSED*
2	*Account name < 2 char*	*Alert min 3 char*	*Invalid*	*PASSED*
3	*Account name > 30 char*	*Alert min 30 char*	*Invalid*	*PASSED*
4	*Bank number < 3 char*	*Alert min 3 char*	*Invalid*	*PASSED*
5	*Bank number > 30 char*	*Alert min 30 char*	*Invalid*	*PASSED*
6	*Verification code wrong*	*Error account Failed to add*	*Invalid*	*PASSED*
7	*Withdrawal Nominal with alphabet*	Tidak bisa *input* nominal	*Invalid*	*PASSED*

Setelah dilakukan *test* pada fitur *reports* didapatkan hasil sebanyak 7 *test case PASSED* dan 0 *test case FAILED*.

e. *Test Case Tenant*

Pada pengujian *test case tenant* Tabel 6, terdapat sebanyak 10 *test case* pengujian yang terbagi menjadi 1 *test case valid* dan 9 *test case invalid.*

Tabel 6 *Test Case* Fitur *Tenant*

TC ID	*Test Case Name*	*Expected Result*	*Category*	Hasil
1	*Edit Tenant*	*Success Edit profile*	*Valid*	*PASSED*
2	Nama Tenant 2 char	*Popup minimal 3 char*	*Invalid*	*PASSED*
3	Nama Tenant 40 char	*Popup max 35 char*	*Invalid*	*PASSED*
4	Number HP Tenant 2 char	*Popup minimal 10 number*	*Invalid*	*PASSED*
5	Number HP Tenant 25 char	*Popup max 20 char*	*Invalid*	*PASSED*
6	Number HP Tenant with symbol	*Popup The string supplied did not seem to be a phone number*	*Invalid*	*PASSED*
7	Number HP Tenant with alphabet	*Popup The string supplied did not seem to be a phone number*	*Invalid*	*PASSED*
8	*Current Password* tidak sesuai	*Popup* Informasi *current password* salah	*Invalid*	*FAILED*
9	*Password < 3 char*	*Password min 3 char*	*Invalid*	*FAILED*
10	*Password blank*	*Error alert required*	*Invalid*	*PASSED*

Setelah dilakukan *test* pada fitur *tenant* didapatkan hasil sebanyak 8 *test case PASSED* dan 2 *test case FAILED*.

Berdasarkan data hasil pengujian dari fitur *login*, *menu*, *order*, *reports* dan *tenant* diperoleh sebanyak 34 *test case PASSED* dan sebanyak 2 *test case FAILED* dengan jumlah keseluruhan *execute test case* 36 *case*.

Berdasarkan data hasil uji dari seluruh fitur dilakukan pengolahan data dengan *Software Testing Metrics* sebagai berikut

1. Percentage Test Case Execute

Percentage Test Case Execute adalah persentase banyak kasus uji yang dieksekusi dari total keseluruhan kasus uji yang telah dirancang.

Rumus untuk menghitung *Percentage Test Case Execute adalah*

$$\frac{Number\ of\ Test\ cases\ executed}{Total\ number\ of\ Test\ cases\ written}\ x\ 100 = x\ \%$$

Maka Percentage Test Case Execute =

$$\frac{36}{83}\ x\ 100 = 43{,}37\ \%$$

2. Percentage Test Case not Execute

Percentage Test Case not Execute adalah persentase banyak kasus uji yang tidak dieksekusi dari total keseluruhan kasus uji yang telah dirancang.

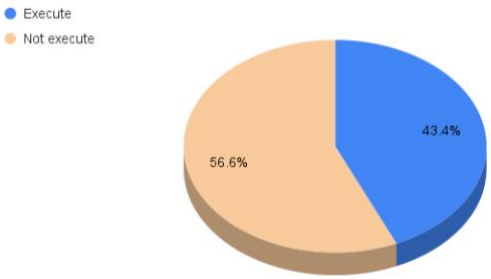
Rumus untuk menghitung *Percentage Test Case not Execute* adalah

$$\frac{Number\ of\ Test\ cases\ not\ executed}{Total\ nubmer\ of\ Test\ cases\ written}\ x\ 100 = x\ \%$$

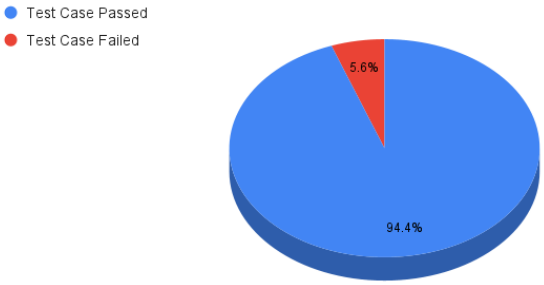
Maka *Percentage Test Case not Execute* =

$$\frac{47}{83}\ x\ 100 = 56{,}62\ \%$$

Berdasarkan hasil perhitungan dapat dilihat *Percentage Test Case Execute* dan *Percentage Test Case not Execute* pada Gambar 2



Gambar 2 *Test Execute Coverage Percentage*



Gambar 3 *Test Case Status*

Dari hasil pengujian didapatkan sebanyak 43.4% *test case execute* dan 56.6% *test case not execute* hal tersebut dikarenakan pada keseluruhan *test case* yang dirancang terdapat beberapa *test case* yang jenis datanya sudah ada setidaknya 1(satu) kasus uji untuk mewakili *equivalence partition* dan ada pula *test case* yang sifatnya memvalidasi *behavior redirect page* sehingga tidak dilakukan pengujian.

3. Percentage Test Case Passed

Percentage Test Case Passed adalah persentase banyak kasus uji yang *passed* dari total keseluruhan kasus uji yang telah dieksekusi.

Rumus untuk menghitung *Percentage Test Case Passed* adalah

$$\frac{Number\ of\ Test\ cases\ Passed}{Total\ nubmer\ of\ Test\ execute}\ x\ 100 = x\ \%$$

Maka *Percentage Test Case Passed* =

$$\frac{34}{36}\ x\ 100 = 94{,}4\ \%$$

4. Percentage Test Case Failed

Percentage Test Case Failed adalah persentase banyak kasus uji yang *failed* dari total keseluruhan kasus uji yang telah dieksekusi.

Rumus untuk menghitung *Percentage Test Case Failed* adalah

$$\frac{Number\ of\ Test\ cases\ Failed}{Total\ nubmer\ of\ Test\ execute}\ x\ 100 = x\ \%$$

Maka *Percentage Test Case Failed* =

$$\frac{2}{36}\ x\ 100 = 5{,}55\ \%$$

Berdasarkan hasil perhitungan dapat dilihat *Percentage Test Case Passed* dan *Percentage Test Case Failed* pada Gambar 3.

3.3 Rekomendasi Perbaikan

Dari keseluruhan hasil pengujian *test case* terdapat 2 *test case failed* berada pada fitur tenant yaitu *test case password*. Ketika *input Current Password* yang tidak sesuai kemudian meng-*input*-kan *new password* hasil yang didapat adalah *success alert* berhasil merubah *password*, yang dimana seharusnya tidak bisa merubah *password* dikarenakan *current password* tidak sesuai, hal yang menyebabkan kegagalan pada fitur ini adalah tidak adanya validasi pada *current password* sehingga walaupun *user* meng-*input*-kan *current password* secara acak atau tidak sesuai sistem akan memberikan *response* berhasil merubah *password*.

Kegagalan lainnya pada *test case Password* < 3 *characters* yaitu ketika *user* meng-*input*-kan *new password* < 3 characters maka akan berhasil *update password*, yang dimana sesuai dengan *requirement system password* diberikan batas minimal yaitu harus lebih dari sama dengan 3 *character*. Hal ini terjadi juga karena tidak adanya validasi pada *new password* sehingga *user* masih bisa meng-*input*-kan *new password* kurang dari 3 *character*.

Dengan adanya 2 kegagalan pada fitur *tenant* maka hal yang dapat diperbaiki adalah memberikan validasi untuk fitur *update password* yang dimana sistem harus dapat memvalidasi apakah *current password* yang di-*input* sudah sesuai dengan *account merchant* yang sedang *login* dan memvalidasi jumlah minimum dan maksimum untuk *new password*.

Secara keseluruhan berdasarkan referensi data dari 3 perusahaan dan menurut(Afzal, 2007) batas minimal hasil pengujian yang dapat diterima atau dikatakan layak sebesar 90%. Dikarenakan hasil pengujian aplikasi InKanteen menggunakan *software metrics testing* mendapat sebesar 94% *test case passed* maka dapat dikatakan aplikasi InKanteen merchant untuk fitur *login*, *menu*, *order*, *reports*, dan *tenant* sudah dapat diterima atau layak digunakan.

4. KESIMPULAN

Berdasarkan hasil dari analisis dan eksperimen yang telah dilakukan, dapat diperoleh simpulan yaitu pengujian aplikasi InKanteen menggunakan metode *Black box testing* teknik *Equivalence Partition* dengan perhitungan data menggunakan *software*

testing metrics mendapatkan hasil pengujian tingkat keberhasilan sebesar 94% dengan sebanyak 34 *test case passed* dan sebanyak 2 *test case failed*.

Suatu aplikasi atau sistem dapat dikatakan layak ketika tingkat keberhasilan *test case passed* minimal sebesar 90% sehingga dengan hasil perhitungan yang didapat bisa dikatakan aplikasi InKanteen dapat dikatakan layak pakai. Untuk langkah selanjutnya perlu dilakukan perbaikan pada 2 *test case failed* yaitu menambahkan validasi pada fitur *password* sehingga seluruh *test case* dapat berjalan sesuai harapan.

Kemudian untuk *indicator process metrics* yang diuji dapat semakin beragam sehingga hasil pengujian *metrics case* lainnya guna melengkapi dan memperkuat hasil akhir pengujian.

DAFTAR PUSTAKA

AFZAL, W., 2007. Metrics in Software Test Planning and Test Design Processes. (January), p.105.

APJJI, 2022. Profil Internet Indonesia 2022. *Asosiasi Penyelenggara Jasa Internet Indonesia (APJJI)*. [online] Available at: <http://apjii.or.id/v2/upload/Laporan/Profil Internet Indonesia 2012 %28INDONESIA%29.pdf>.

BAHAR, E., Nasution, S., Dewi, A.R. and Wahyuni, R.S., 2019. Manual Test Metrics in Genetic Algorithm. *Journal of Physics: Conference Series*, 1235(1), pp.2–9. https://doi.org/10.1088/1742-6596/1235/1/012070.

JOVANOVIC, I., 2009. Software Testing Methods and Techniques. *The IPSI BgD Transactions on Internet Research*, [online] 5(1), pp.30–41. Available at: <http://www.internetjournals.net/journals/tir/2009/January/Full Journal.pdf#page=31>.

MLADENOVA, T., 2020. Software Quality Metrics - Research, Analysis and Recommendation. *2020 International Conference Automatics and Informatics, ICAI 2020 - Proceedings*. https://doi.org/10.1109/ICAI50593.2020.9311361.

PRESSMAN, R.S., 2001. *Software Engineering: a Practitioner's Approach*. FIFTH EDIT ed. *Software Engineering Journal*, https://doi.org/10.1049/sej.1995.0031.