Pengimplementasian *Unit Testing, Integration Testing,* dan *Usability Testing* pada Aplikasi Cafeasy Berbasis Website (Studi Kasus: Kafe di Daerah Bandung)

1st Angelia Brigitta Maharani Mutiara Prabowo
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
angiestrange@student.telkomuniversity.ac.id

2nd Dana Sulistyo Kusumo
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
danakusumo@telkomuniversity.ac.id

3rd Nungki Selviandro
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
nselviandro@telkomuniversity.ac.id

***Abstrak* — Cafeasy merupakan aplikasi yang dibangun untuk membantu berlangsungnya proses bisnis untuk café di daerah kota Bandung baik dari sisi *customer* mau pun dari sisi admin. Dengan menggunakan Cafeasy, pelanggan dapat dengan mudah melakukan pemesanan dan melakukan pembayaran langsung pada aplikasi. Bagi admin yang adalah karyawan atau *owner* café, Cafeasy memberikan fitur pembukuan yang berintegrasi dengan data inventori. Dalam proses pengembangannya, dilakukan beberapa pengujian untuk memastikan bahwa aplikasi Cafeasy telah berjalan dengan sebagaimana mestinya. Pengujian yang dilakukan pada penelitian ini adalah *unit testing* untuk sisi *customer* dan admin, *integration testing* untuk sisi admin, dan *usability testing* untuk sisi *customer.* Dari hasil *unit testing,* didapat sebesar 100% fungsi dari unit yang dideskripsikan baik dari sisi *customer* maupun sisi admin sudah berjalan dengan sebagai mana mestinya. Kemudian, modul-modul yang berhubungan sudah berintegrasi sesuai fungsinya, dan secara usabilitas, aplikasi Cafeasy sudah memiliki tingkat usabilitas yang tinggi serta sistem sudah efektif memenuhi kebutuhan dan harapan dari pengguna.**

Kata kunci*— *white-box testing, unit testing, integration testing, usability testing, cafeasy

I. PENDAHULUAN

Seiring dengan perkembangan teknologi dan penggunaan internet yang semakin meningkat, banyak kesempatan baru untuk memanfaatkan kemampuan teknologi ini untuk meningkatkan kualitas pelayanan, terutama di tempat-tempat bisnis seperti restoran dan kafe. Pemesanan otomatis sudah marak digunakan untuk memudahkan proses bisnis di restoran dan kafe. Namun sayangnya, 6 dari 9 kafe di daerah Kota Bandung yang masuk ke dalam kategori medium masih melakukan kegiatan pemesanan secara manual, yang kemudian aktivitas pembukuannya pun diterapkan secara manual, dimana sangat rentan hilangnya data seperti riwayat transaksi, yang nantinya dapat menyebabkan kerugian terhadap kafe tersebut.

Berdasarkan permasalahan tersebut, dikembangkanlah perangkat lunak untuk melakukan pemesanan yang menyediakan fitur pembukuan dan pencatatan inventoris otomatis yang bernama 'Cafeasy'. Cafeasy akan dapat dijalankan dari dua sisi, yakni sisi customer untuk melakukan pemesanan dan sisi admin yang mendapatkan fitur pembukuan yang terintegrasi dengan data inventori.

Pengujian perangkat lunak sendiri adalah suatu kegiatan dengan tujuan untuk memastikan sebuah perangkat lunak yang sedang dikembangkan sudah sesuai dengan requirement yang sudah terdefinisikan sebelumnya. Fungsi dari kegiatan ini adalah untuk menemukan ketidaksesuaian antara hasil sebenarnya dengan hasil yang diharapkan [1][2]. Pengujian perangkat lunak harus dilakukan untuk melakukan pengujian perangkat lunak agar mengurangi kemungkinan terjadinya kesalahan yang dibuat oleh manusia dan mengkompensasi keterbatasan manusia dalam berkomunikasi dengan sempurna antar satu sama lain, yang dapat menyebabkan pengembangan perangkat lunak menjadi terhambat dan menjadi tergantung pada jaminan kualitas produk [3].

Dalam sistem informasi, pengujian perangkat lunak memainkan peran yang sangar krusial. Dalam masa pengujian, kemungkinan kerusakan atau error dari perangkat lunak dapat ditemukan. Dengan adanya pengujian perangkat lunak ini, diharapkan dapat mengurangi kesalahan dan error dari perangkat lunak dan berfungsi juga sebagai tolak ukur atas kualitas produk perangkat lunak tersebut [4]. Maka dari itu, sangat perlu dilakukannya aktivitas pengujian agar dapat meminimalisir terjadinya kesalahan yang menyebabkan kerugian tersebut [5].

Dikarenakan Cafeasy adalah aplikasi berbasis website yang dibuat selama penelitian ini berlangsung, maka perlu dilakukannya pengujian untuk memastikan aplikasi dapat berjalan dengan sebagaimana mestinya. Pengujian yang akan dilakukan pada penelitian ini adalah unit testing, integration testing, dan usability testing. Pengujian dalam penelitian ini akan menggunakan metode white-box testing, dimana akan dilakukan pengujian unit dan pengujian integrasi agar memvalidasi bahwa sebuah unit dari sebuah kode akan berjalan sebagaimana mestinya dan kode yang diuji juga akan menghasilkan hasil yang sama setiap kali kode tersebut

dieksekusi. Kemudian dua unit atau lebih yang sudah selesai tahap pengujian unit akan masuk ke pengujian integrasi agar interaksi antar unit tersebut tidak menemukan kendala [6][7][8]. Sedangkan pengujian usabilitas akan dilakukan agar mengevaluasi pengalaman pengguna ketika berinteraksi dengan sebuah website atau aplikasi, sehingga dapat membantu team proyek menilai kemudahan perangkat lunak tersebut dan mencapai produk akhir sesuai apa yang diharapkan oleh end-user.

Berdasarkan permasalahan di atas, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Untuk memastikan apakah fungsi dari setiap unit yang dibuat sudah berjalan sesuai dengan *product backlog* yang terdefinisikan.
2. Untuk memastikan bahwa modul-modul yang ada sudah saling berinteraksi sesuai dengan alur bisnis aplikasi.
3. Untuk mengevaluasi pengalaman pengguna ketika menggunakan dan berinteraksi dengan Cafeasy.

II. KAJIAN TEORI

A. Pengujian Perangkat Lunak

Dalam proses pengembangan perangkat lunak, terdapat kegiatan yang disebut dengan pengujian perangkat lunak, yang di dalamnya terdapat proses pengeksekusian sebuah sistem perangkat lunak untuk menentukan apakah sudah sesuai dengan spesifikasinya dan dijalankan pada *environment* yang dimaksud [9]. Pengujian perangkat lunak tidak terbatas dalam mendeteksi *error* saja, melainkan menjalankan perangkat lunak tersebut di dalam kondisi spesifik yang terkontrol, dimana kondisi tersebut adalah sebagai berikut [10]:

1. *Verification*, dimana dilakukan pengecekan atau pengujian perangkat lunak agar konsisten dan sesuai dengan mengevaluasi hasil akhirnya terhadap requirement yang sebelumnya sudah ditentukan.
2. *Error finding*, dimana dalam pengujian seharusnya disengajakan untuk membuat suatu kesalahan untuk mengetahui apakah terjadi hal yang seharusnya tidak terjadi atau pun sebaliknya.
3. *Validation*, dimana akan proses pengecekan untuk menunjukkan kesesuaian dari sistem tersebut, yakni apa yang sudah disepsifikasikan sesuai dengan apa yang pengguna inginkan.

Spesifikasi perangkat lunak merupakan artefak yang sangat penting untuk menjalankan pengujian. Karena dalam sebuah spesifikasi, terdefinisikan hal-hal yang seharusnya berjalan dengan benar, sehingga jika ada sesuatu yang salah dapat langsung teridentifikasi. Hal yang salah tersebut, dalam sebuah perangkat lunak disebut sebagai *failure*, dimana disebabkan oleh kesalahan pada *source code*, yang biasanya disebut sebagai *bug*. Pada umumnya, *developer code* akan mendiagnosis penyebab kesalahan tersebut.

Pengujian perangkat lunak diklasifikasikan berdasarkan aturan, dimana tester akan menjalankan dua tahap pertama dari proses pengujian. Scope dari tahap pertama, memodelkan environment perangkat lunak, akan menentukan apakah penguji sedang melakukan unit, integration, atau *system testing* [9].

B. *White-Box Testing*

Pada pengujian *white-box,* diperlukan pemahaman mengenai bagaimana sistem yang diuji diimplementasikan [11]. Pengujian ini sangat efisien dalam mendeteksi dan mengatasi permasalahan, dikarenakan *bugs* biasanya lebih cepat ditemukan sebelum menimbulkan masalah. Dalam pengujian ini, *test case* dikalkulasikan berdasarkan analisis struktur internal dari sistem berdasarkan *code coverage, branch coverage, path coverage, condition coverage* dan lainnya [12]. Terdapat beberapa teknik pengujian untuk pengujian *white-box,* diantaranya adalah sebagai berikut [6]:

1. *Statement coverage,* yang merupakan pengukuran dari persentasi *statement* yang telah dieksekusi oleh *test case.* Jika *statement coverage* kurang dari 100%, maka tidak semua baris dari kode telah tereksekusi.
2. *Branch coverage,* atau biasa disebut sebagai *decision coverage*, merupakan metode pengujian yang bertujuan untuk memastikan setiap *branch* dari setiap poin *decision* tereksekusi setidaknya satu kali dan memastikan setiap kode yang bisa dicapai tereksekusi. *Branch coverage* biasanya ditunjukkan untuk memenuhi *statement coverage,* jika persentasenya 100%, maka setiap control flow graph sudah dilalui.
3. *Path coverage,* pada teknik ini *test case* akan dieksekusi sedemikian rupa hingga path akan tereksekusi setidaknya satu kali. *Path testing* sendiri merupakan metode pengujian struktural yang melibatkan *source code* dari sebuah program untuk menemukan setiap path yang memungkinkan untuk dieksekusi.

C. *Unit Testing*

Unit testing merupakan sebuah praktik dalam sebuah pengembangan perangkat lunak, dimana pengujian ini bertujuan untuk memvalidasi bahwa sebuah unit dari sebuah kode akan berjalan sebagaimana mestinya dan kode yang diuji juga akan menghasilkan hasil yang sama setiap kali kode tersebut dieksekusi. Sebuah unit adalah bagian terkecil dari sebuah software yang dapat diuji, biasanya hanya memiliki satu atau beberapa input dan menghasilkan satu output. Unit testing memberikan banyak kelebihan, termasuk di dalamnya adalah menemukan bug lebih awal, mempercepat proses pengembangan perangkat lunak, mencegah regresi bug, serta memberikan pemahaman dari kode program [6][7].

D. *Integration Testing*

Integration testing merupakan tahap selanjutnya dari proses pengujian setelah *unit testing* selesai dilakukan. Setelah unit yang diuji perindividu berjalan dengan baik, ketika diuji dengan menggabungkan dua unit atau lebih yang telah diintegrasi kebanyakan tidak berjalan dikarenakan perbedaan tipe error yang ditemukan pada perbedaan level pengujian. Tujuan dari *integration testing* adalah untuk meletakkan unit-unit pada *environment* semestinya, dan mengatur interaksi antar unit tersebut secara menyeluruh [8].

E. *Usability Testing*

Usability testing merupakan sebuah metode untuk mengukur seberapa mudah sebuah perangkat lunak aplikasi [13]. *Usability testing* mengevaluasi pengalaman pengguna ketika berinteraksi dengan sebuah website atau aplikasi,

sehingga dapat membantu team proyek menilai kemudahan perangkat lunak tersebut. Prosesnya sendiri dilakukan dengan menanyakan pengguna asli, diluar *developer* untuk menyelesaikan sebuah task terkait dengan perangkat lunak. Hasil dari proses tersebut akan dianalisa untuk melihat kemungkinan permasalahan dan membuat ruang untuk melakukan improvisasi produk [14].

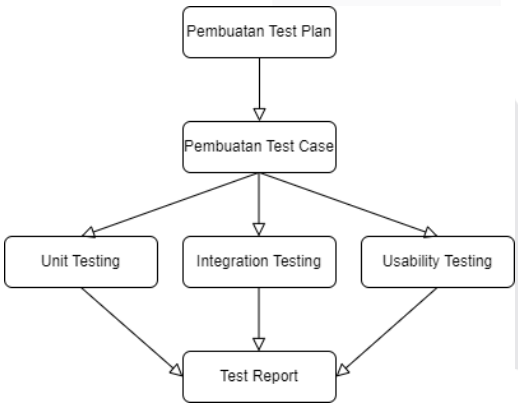
F. *System Usability Scale*

Skala usabilitas dari sebuah sistem, atau yang biasa disebut dengan SUS (*System Usability Scale*) pertama kali dikembangkan oleh John Brooke pada tahun 1996 dimana nilai dari SUS adalah untuk memberikan satu skor referensi untuk pandangan partisipan terkait usabilitas suatu produk. SUS terdiri dari 10 item, dimana item dengan nomor ganjil adalah untuk item yang positif dan nomor genap adalah untuk item yang negatif. Responden dari SUS diminta untuk memberikan poin untuk setiap item dari skala 1 (sangat tidak setuju) hingga 5 (sangat setuju). Dengan perhitungan sebagai berikut [15]:

1. Setiap item bernomor ganjil, skor dari setiap item yang didapat dari skor pengguna akan dikurangi 1.
2. Setiap item bernomor genap, skor akhir yang didapat dari nilai 5 dikurangi skor item yang didapat dari pengguna.
3. Skor SUS didapat dari hasil penjumlahan skor setiap item yang kemudian dikali 2,5.

III. METODE

Merujuk pada Gambar 1, alur pelaksanaan pengujian dari aplikasi *website* Cafeasy disajikan dalam bentuk bagan yang terdiri dari pembuatan *test plan* dan *test case*, pelaksanaan pengujian, yang dalam penelitian ini adalah *unit testing, integration testing*, dan *usability testing*, dan diakhiri dengan penulisan *report* untuk setiap pengujian yang dilakukan.



GAMBAR 1
ALUR PENGUJIAN

A. Pembuatan *Test Plan*

Sebelum dilakukannya sebuah pengujian, perlu dibuat terlebih dahulu sebuah *plan* atau rencana untuk menjadikannya sebagai referensi utama. Dalam penelitian ini, referensi utama diambil dari *product backlog* yang tertulis dalam bentuk *user story,* sebagai *requirement* yang perlu ada di dalam aplikasi untuk setiap *sprint* yang berlangsung. Tabel 1 merupakan *product backlog* yang dihasilkan selama proses *scrum* yang dilakukan untuk fokus pada penelitian ini.

TABEL 1
PRODUCT BACKLOG CAFEASY YANG DIGUNAKAN

Backlog ID	Deskripsi
PM-02	Sebagai pelanggan cafe, saya ingin melakukan sign in dengan menuliskan nama saja, sehingga saya dapat melakukan pemesanan.
PM-05	Sebagai pelanggan cafe, saya ingin melihat deskripsi dari setiap makanan dan minuman, sehingga saya dapat mengetaui detail dari makanan atau minuman yang dipesan
PM-06	Sebagai pelanggan cafe, saya ingin melihat daftar makanan dan minuman berdasarkan kategorinya, sehingga saya dapat menemukan makanan atau minuman yang saya inginkan dengan mudah.
PM-07	Sebagai pelanggan cafe, saya ingin melihat harga, dan status / ketersediaan dari setiap makanan dan minuman, sehingga saya dapat meyimpulkan makanan atau minuman apa yang enak atau sering dipesan
PM-08	Sebagai pelanggan cafe, saya ingin menentukan jumlah dari setiap makanan atau minuman yang dipesan, sehingga sesuai dengan kebutuhan saya.
PM-09	Sebagai pelanggan cafe, saya ingin menambahkan catatan untuk makanan atau minuman yang saya pesan, sehingga makanan dan minuman yang dipesan sesuai dengan keinginan saya.
PM-10	Sebagai pelanggan cafe, saya ingin mengganti pesanan apabila pembayaran belum dilakukan, sehingga saya dapat menyesuaikan pesanan saya.
PM-13	Sebagai pelanggan cafe, saya ingin melihat daftar pesanan yang saya pesan sebelum melakukan pembayaran, sehingga tidak ada kekeliruan dalam pesanan saya
PM-14	Sebagai pelanggan cafe, saya ingin melihat status pembayaran menggunakan Bank Transfer / E-Wallet dari pesanan yang saya lakukan, sehingga saya dapat memastikan bahwa pesanan yang saya lakukan sudah terbayar ataupun belum.
PM-15	Sebagai pelanggan cafe, saya ingin melihat status pembayaran menggunakan cash dari pesanan yang saya lakukan, sehingga saya dapat memastikan bahwa pesanan yang saya lakukan sudah terbayar ataupun belum.
PM-21	Sebagai pelanggan cafe, saya ingin melihat status pada setiap menu, sehingga saya dapat melakukan pemesanan pada menu yang tersedia.
PM-22	Sebagai pelanggan cafe, saya ingin melihat detail pesanan saya sebelum melakukan pembayaran, sehingga saya dapat memastikan Kembali pesanan yang saya pilih
KF-01	Sebagai pemilik dan karyawan cafe, saya ingin mendaftarkan akun, sehingga café saya terdaftar pada aplikasi.
KF-02	Sebagai pemilik dan karyawan cafe, saya ingin melakukan sign in, sehingga saya dapat masuk ke halaman utama dan menggunakan fitur yang tersedia.
KF-03	Sebagai pemilik dan karyawan cafe, saya ingin melihat pesanan masuk yang dilakukan oleh pelanggan.
KF-04	Sebagai pemilik dan karyawan cafe, saya ingin melihat status pembayaran yang dilakukan oleh pelanggan, sehingga saya dapat mengkonfirmasi pesanannya
KF-07	Sebagai pemilik dan karyawan cafe, saya ingin menambahkan makanan atau minuman ke dalam menu, sehingga jika ada makanan atau minuman baru dapat ditambahkan ke dalam menu.
KF-08	Sebagai pemilik dan karyawan cafe, saya ingin mengedit nama, harga, deskripsi, dan foto makanan dan minuman dalam menu, sehingga jika ada perubahan harga dapat di edit.

KF-09	Sebagai pemilik dan karyawan cafe, saya ingin mengedit data inventori stok barang, sehingga data yang ada dapat disesuaikan dengan kebutuhan.
KF-10	Sebagai pemilik dan karyawan cafe, saya ingin melakukan filterisasi pada menu pembukuan, sehingga saya dapat melihat transaksi masuk per-hari, per-minggu ,dan per-bulan
KF-11	Sebagai pemilik dan karyawan cafe, saya ingin melihat menu makanan dan minuman dan melakukan filterisasi pada menu inventori data, sehingga saya dapat melihat data stok barang per-hari, per-minggu, dan per-bulan.
KF-12	Sebagai pemilik dan karyawan cafe, saya ingin mencetak pembukuan dari transaksi per-hari ke dalam google sheets, sehingga saya dapat menyimpannya.
KF-13	Sebagai pemilik dan karyawan cafe, saya ingin melihat data pesanan yang dilakukan pelanggan, sehingga saya dapat melihat status pembayaran yang dilakukan pelanggan
KF-14	Sebagai pemilik dan karyawan cafe, saya ingin melihat halaman kategori menu, sehingga saya dapat menambah, edit, dan menghapus kategori sesuai dengan menu yang tersedia pada café.
KF-16	Sebagai pemilik dan karyawan café, saya ingin melihat halaman profil admin, sehingga saya dapat melihat data saya serta melakukan perbaruan data.
KF-17	Sebagai pemilik dan karyawan café, saya ingin melihat data pelanggan yang sudah melakukan pendaftaran pada aplikasi, sehingga dapat membantu saya untuk tracking pengguna.
KF-18	Sebagai pemilik dan karyawan café, saya ingin melakukan logout dari aplikasi, sehingga ketika pergantian shif karyawan lain dapat melakukan sign in dengan menggunakan akunnya.

B. Pembuatan *Test Case*

Setelah *test plan* sudah dibuat, tahap selanjutnya adalah penulisan *test case* dan *scenario* dimana dokumen ini berisikan langkah-langkah yang akan diikuti oleh penguji untuk menguji suatu fitur tertentu. Tujuan dibuatnya dokumen ini adalah untuk memvalidasi apakah aplikasi sudah berjalan sesuai dengan apa yang diharapkan.

C. *Unit Testing*

Pelaksanaan pengujian *unit* dilakukan terhadap sisi *customer* dan sisi admin dari aplikasi, dengan menggunakan Jest sebagai framework pengujian. Tahap pengujian unit dilakukan dengan menggunakan metode *white-box testing*, dimana pengujian dilakukan dengan akses langsung ke struktur internal, *logic*¸ dan implementasi dari sumber kode perangkat lunak yang diuji. Pengujian dilakukan dengan pendekatan *statemen*t atau *line coverage*, dimana fokus utamanya adalah mengukur sejauh mana kode sumber dari unit terkecil (fungsi atau metode) telah dieksekusi selama rangkaian pengujian. Semakin tinggi *statement coverage* yang diraih berkorelasi dengan probabilitas untuk mendeteksi lebih banyak cacat dan meningkatkan reliabilitas perangkat lunak, serta mendapatkan kepercayaan pada kualitas perangkat lunak yang berfungsi sebagai indikator kelengkapan dan efektivitas pengujian [16]

D. *Integration Testing*

Pelaksanaan pengujian integrasi dilakukan untuk aplikasi dari sisi admin dengan Cypress sebagai *tool* pengujiannya. Pengujian integrasi bertujuan untuk menguji apakah modul-modul pada aplikasi sudah saling berinteraksi sesuai *product backlog* yang sebelumnya telah didefinisikan. Alur dari pengujian integrasi disesuaikan dengan *use case diagram* untuk sisi admin. Metode yang digunakan adalah dengan *black-box testing* dimana pengujian berfokus kepada fungsionalitas dari aplikasi dilihat dari sudut pandang pengguna. Pada penelitian ini, terdapat 6 modul yang diuji, yakni modul *sign up, log in*, data menu, data kategori, data pelanggan dan data transaksi. Pada Tabel 4 terlampir detail alur dari modul yang diuji pada pengujian. Pengujian ini dilakukan dengan cara menguji lebih dari satu modul bersamaan, sebagai contoh, untuk Test Case ID IT-TC-KF01 yang diintegrasikan adalah modul *login*, data kategori, dan data menu (lihat kolom modul pada Tabel 4).

E. *Usability Testing*

Pelaksanaan *usability testing* dilakukan untuk mengevaluasi usabilitas aplikasi dari sudut pandang pengguna. Pengujian dilakukan dengan menggunakan pendekatan *in person moderated*, agar proses pengujian tidak melibatkan distraksi yang berlebih karena *environment* lebih terkontrol, dan mencegah terjadinya gangguan teknis atau jaringan karena dilaksanakan secara langsung. Pada pengujian ini, partisipan akan diberikan *task scenario*, untuk mengukur aspek efektivitas dan efisiensi. Setelah partisipan sudah menyelesaikan *scenario* yang diberikan, partisipan diminta untuk mengisi kuesioner terkait pengalaman yang dirasakan saat menyelesaikan *scenario*. Pertanyaan yang diajukan terdiri dari 10 pertanyaan, dengan rentang nilai 1 untuk pernyataan sangat tidak setuju, hingga 5 untuk pernyataan sangat setuju. Poin yang didapat nantinya akan dihitung untuk mendapatkan nilai SUS atau *System Usability Scale* yang dapat dihitung dengan (1).

$$SUS = \Big(\big((Q1-1)+(Q3-1)+(Q5-1)+(Q7-1)+(Q9-1)\big)+\big((5-Q2)+(5-Q4)+(5-Q6)+(5-Q8)+(5-Q10)\big)\Big) \times 2{,}5 \quad (1)$$

F. Pembuatan *Test Report*

Merupakan langkah penting dalam proses pengujian perangkat lunak. Laporan ini berisi hasil dan temuan dari pengujian yang telah dilakukan. Sebuah test report yang baik harus jelas, ringkas, dan informatif, sehingga dapat memberikan informasi yang bermanfaat bagi semua pemangku kepentingan yang terlibat dalam proyek.

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian

1. *Unit Testing*

Pada *unit testing* untuk sisi dari *customer*, terdapat 12 *product backlog* untuk *backend*, dan menghasilkan *test case* sebanyak 30 buah seperti yang terlihat pada Tabel 2, terdapat 12 *product backlog* untuk *backend* dari sisi admin, dan menghasilkan *test case* sebanyak 44 buah seperti yang terlampir pada Tabel 3. Terdapat product backlog backend pada customer tidak diambil dikarenakan memerlukan integrasi dengan Google dan Midtrans, sedangkan pada admin terdapat *product backlog* yang tidak diambil karena memerlukan integrasi dengan *Google Spreadsheet*.

TABEL 2
REKAP *UNIT TESTING* CUSTOMER

Backlog ID	Deskripsi Singkat	Jumlah *Test case*	Stmts (%)	Status	
				Passed	*Failed*
PM-07	Customer dapat melihat menu yang ada	8	94,73	8	-
PM-05	Customer dapat melihat detail menu yang dipilih				
PM-06	Customer dapat memilih menu dengan tambahan kategori				
PM-09	*Customer* dapat menyantumkan catatan dari setiap item yang dipilih				
PM-21	Customer dapat melihat status menu				
PM-02	Customer dapat menuliskan nama saja untuk login	2	96,77	2	-
PM-08	Customer dapat menentukan jumlah pesanan dari setiap item yang dipilih	12	82	12	-
PM-10	Customer dapat mengganti pesanan jika belum dibayar atau tidak sesuai pesanan				
PM-13	Customer dapat melihat daftar pesanan yang dipilih				
PM-22	Customer dapat melihat detail pesanan yang dipilih				
PM-14	Customer dapat melihat status pembayaran dengan Bank Transfer / e-wallet	8	81,53	8	-
PM-15	Customer dapat melihat status pembayaran dengan cash				
All files Line Coverage (%)			**86,2**		

TABEL 3
REKAP *UNIT TESTING* ADMIN

Backlog ID	Deskripsi Singkat	Jumlah *Test case*	Stmts (%)	Status	
				Passed	*Failed*
KF-01, KF-02	Admin berhasil melakukan sign in dan sign up	13	88,17	13	-
KF-16	Admin dapat melihat profil admin dan mengedit profil				
KF-18	Admin dapat melakukan log out				
KF-10	Admin dapat melakukan filterisasi menu pembukuan	7	85,18	7	-
KF-03	Admin dapat melihat pesanan yang masuk setelah pembayaran berhasil dilakukan oleh pelanggan				
KF-04	Admin dapat melihat status pembayaran pelanggan				
KF-13	Admin dapat melihat data pesanan pelanggan				
KF-07	Admin dapat menambahkan makanan atau minuman baru ke dalam menu	24	93,22	24	-
KF-08	Admin dapat melakukan edit menu (nama menu, harga menu, stok menu, deskripsi menu, dan kategori menu)				
KF-11	Admin dapat menggunakan filterisasi menu inventori				
KF-09	Admin dapat mengedit data inventori stok barang				
All files Line Coverage (%)			**90,21**		

2. *Integration Testing*

Pada *integration testing*, dari 15 *product backlog* untuk sisi admin yang terdefinisikan, terbentuk 6 buah *test case* untuk melihat apakah modul-modul sudah saling berintegrasi dengan baik. Tabel 4 merupakan hasil yang didapat dari proses pengujian integrasi.

TABEL 4
REKAP *INTEGRATION TESTING* ADMIN

TC ID	Backlog ID	Modul	Deskripsi	Expected Result	Status
IT-TC-	KF-02, KF-07, KF-12,	Login	Mengunjungi "localhost:3000"	Site terbuka	Pass

KF01	KF-14		Memasukkan email	Email berhasil diinput	Pass
			Memasukkan nama pengguna	Nama pengguna berhasil diinput	Pass
			Click tombol submit	Pengguna berhasil login	Pass
			Terbuka halaman yang memiliki "/ProfileAdmin"	Halaman berhasil terbuka	Pass
		Data Kategori	Navigasi ke halaman Data Kategori	Halaman berhasil terbuka	Pass
			Click button tambah kategori	Form tambah kategori akan muncul	Pass
			Masukkan nama kategori yang akan ditambahkan	Nama kategori berhasil diinput	Pass
			Click button simpan	Nama kategori berhasil	Pass
			Muncul pop up "Data Berhasil Disimpan"	Pop up muncul di layar pengguna	Pass
		Data Menu	Navigasi ke halaman Data Menu	Halaman berhasil terbuka	Pass
			Click tombol tambah menu	Form tambah menu terbuka	Pass
			Memasukkan nama menu	Nama menu berhasil diinput	Pass
			Memasukkan harga	Harga berhasil diinput	Pass
			Memasukkan stok	Stok berhasil diinput	Pass
			Memasukkan deskripsi	Deskripsi berhasil diinput	Pass
			Pilih kategori menu	Kategori berhasil dipilih	Pass
			Upload gambar menu	Gambar berhasil diupload	Pass
			Click tombol simpan	Menu baru berhasil ditambah	Pass
			Click tombol "Ekspor ke Spreadsheet"	Halaman spreadsheet terbuka dan berhasil mengekspor data menu	Pass
IT-TC-KF02	KF-01, KF-02	Sign Up	Mengunjungi "localhost:3000"	Site terbuka	Pass
			Buka halaman Sign Up	Halaman berhasil terbuka	Pass
			Memasukkan email	Email berhasil diinput	Pass
			Memasukkan nama pengguna	Nama pengguna berhasil diinput	Pass
			Memasukkan nama café	Nama café berhasil diinput	Pass
			Memasukkan alamat café	Alamat café berhasil diinput	Pass
			Memasukkan deskripsi café	Deskripsi café berhasil diinput	Pass
			Memasukkan nama pemilik café	Nama pemilik café berhasil diinput	Pass
			Memasukkan nomor telepon	Nomor telepon berhasil diinput	Pass
			Menguplod foto café	Foto café berhasil diupload	Pass
			Click tombol submit	Pengguna berhasil sign up	Pass
			Muncul pop up "Daftar Berhasil!"	Pop up muncul di layar pengguna	Pass
			Redirect ke halaman Login Admin	Halaman login terbuka	Pass
		Login	Memasukkan nama pengguna	Nama pengguna dapat diinput	Pass
			Memasukkan sandi	Sandi dapat diinput	Pass
			Click tombol submit	Pengguna berhasil login	Pass
			Terbuka halaman yang memiliki "/ProfileAdmin"	Halaman berhasil terbuka	Pass
IT-TC-KF03	KF-08, KF-09, KF-11	Data Menu (Edit)	Memasukkan nama menu pada search bar	Tabel menampilkan nama menu yang dicari	Pass
			Click tombol edit	Muncul form detail menu	Pass
			Masukkan nama menu baru	Nama menu berhasil diinput	Pass
			Masukkan stok menu baru	Stok baru berhasil diinput	Pass
			Click tombol simpan	Nama dan stok berhasil terupdate	Pass
IT-TC-KF04	KF-03, KF-04, KF-10, KF-12, KF-13	Data Transaksi	Memasukkan nama pelanggan pada search bar	Semua data transaksi pelanggan akan	Pass

				ditampilkan pada tabel	
			Click tombol bayar cash	Muncul pop up konfirmasi ubah status bayar	Pass
			Click tombol iya	Status bayar pelanggan berhasil diubah	Pass
			Click tombol "Ekspor ke Spreadsheet"	Membuka halaman spreadsheet dan data transaksi berhasil diekspor	Pass
IT-TC-KF05	KF-16	Profile Admin	Click tombol perbaharui	Terbuka halaman update	Pass
			Memasukkan email	Email pengguna dapat diinput	Pass
			Memasukkan nama pengguna	Nama pengguna dapat diinput	Pass
			Memasukkan sandi	Sandi dapat diinput	Pass
			Memasukkan nama café	Nama café berhasil diinput	Pass
			Memasukkan alamat café	Alamat café berhasil diinput	Pass
			Memasukkan deskripsi café	Deskripsi café berhasil diinput	Pass
			Memasukkan nama pemilik café	Nama pemilik café berhasil diinput	Pass
			Memasukkan nomor telepon	Nomor telepon berhasil diinput	Pass
			Mengupload foto café	Foto café berhasil diupload	Pass
			Click tombol simpan	Pengguna berhasil mengupdate	Pass
			Muncul pop up "Update Berhasil!"	Pop up muncul di layar pengguna	Pass
IT-TC-KF-06	KF-17	Data Pelanggan	Memasukkan nama pelanggan pada search bar	Semua data pelanggan akan ditampilkan pada tabel	Pass

3. *Usability Testing*

Pelaksanaan *usability testing* dimulai dengan membuat task-task dan *scenario* yang mencakup fungsionalitas website yang nantinya akan dilakukan oleh pengguna. Terdapat tujuh orang partisipan yang merupakan pelanggan dari café yang diminta untuk mengerjakan *scenario.* Efektivitas website akan diukur dengan menghitung rata-rata persentase keberhasilan partisipan dalam mengerjakan task yang diberikan, dengan menggunakan rumus *completion rate.* Hasil pengolahan data untuk tingkat efektivitas dari website Cafeasy sisi *customer* dapat dilihat pada Tabel 5.

TABEL 5
HASIL *COMPLETION RATE*

	Task yang berhasil	**Task keseluruhan**	**Persentase Keberhasilan**
P1	9	9	100%
P2	7	9	78%
P3	7	9	78%
P4	9	9	100%
P5	9	9	100%
P6	9	9	100%
P7	9	9	100%
Rata-rata			**94%**

Kemudian, diukur efisiensi dari website berdasarkan rata-rata waktu yang dibutuhkan oleh partisipan, yang didapat dengan menggunakan rumus *time based efficiency.* Hasil pengolahan data untuk tingkat efisiensi dari website Cafeasy sisi *customer* dapat dilihat pada Tabel 6.

TABEL 6
HASIL *TIMED BASED EFFICIENCY*

	Kode Task									
Formula	***TG1***	***TG2***	***TG3***	***TG4***	***TG5***	***TG6***	***TG7***	***TG8***	***TG9***	***Total***
$\sum_{i=i}^{N} \frac{n}{t}$	0,187	0,081	0,131	0,160	0,061	0,061	0,108	0,140	0,126	1,056
Rata-rata (task/detik)										0,117

Setelah itu, didapatkan pula beberapa kendala yang dialami oleh partisipan saat mengerjakan *task scenario* yang diberikan. Kendala tersebut disajikan pada Tabel 7.

TABEL 7
DAFTAR KENDALA YANG DIALAMI

Kode Partisipan	**Kode Kendala**	**Kendala yang dialami**	**Jenis**
P1, P7	M1	Untuk melakukan edit pesanan diberikan tulisan yang lebih jelas, seperti "Edit" dibandingkan catatan.	Khusus
P2	M2	Untuk mengubah catatan kalau bisa lanjut dari catatan sebelumnya tidak perlu tulis dari awal	Khusus
P4, P5	M3	Saat mengurangi jumlah item dalam keranjang tidak perlu ada pop up konfirmasi setiap mengurangi 1 item	Khusus
P7	M4	Kursor saat akan mengklik catatan di bagian edit pemesanan ada baiknya diubah agar lebih mudah dipahami	Khusus

Setelah partisipan selesai mengerjakan *scenario* yang diberikan, partisipan diminta untuk mengisi kuesioner SUS. Skor rata-rata yang didapat adalah sebesar 81,4 dengan satu

partisipan mengisi semua pertanyaan dengan skor 3 atau netral.

B. Analisis Hasil Pengujian

Pada pengujian unit di bagian customer seperti yang terlampir pada Tabel 2 , dari total 30 *test case* yang diuji, semua *test case* menghasilkan status passed dengan *line coverage* untuk seluruh file berada pada angka 86,02% yang berarti dari setiap baris yang dapat dijalankan pada aplikasi sisi customer, sebanyak 86,02% diantaranya telah dieksekusi. Sama halnya untuk bagian admin yang terlampir pada Tabel 3, dari total 44 *test case* yang dibuat juga sudah menghasilkan status passed untuk semua *test case* yang terdefinisi, dengan *line coverage* untuk keseluruhan file berada pada angka 90,21%. Hal ini berarti kode yang diuji sudah berperilaku sesuai yang diharapkan dan sesuai dengan spesifikasi atau *product backlog* yang ditetapkan, serta sebagian besar baris kode sudah tereksekusi baik dari segi customer maupun admin.

Pada kegiatan *integration testing* yang dilakukan terhadap aplikasi dari bagian admin, dengan 11 *product backlog* yang terdefinisikan menghasilkan 6 buah *test case* dengan modul *sign up, log in,* data menu, data kategori, data transaksi dan google spreadsheet yang dapat dilihat pada Tabel 4. Merujuk pada tabel, dengan deskripsi yang merupakan langkah pada test case menghasilkan status *pass* sesuai dengan apa yang diharapkan. Yang berarti, setiap modul yang berkaitan sudah berintegrasi satu sama lain sesuai dengan kegiatan pada *use case diagram* admin, serta berdasarkan pada *product backlog* yang ditetapkan.

Pelaksanaan *usability testing* untuk sisi customer dilakukan dengan melibatkan tujuh orang partisipan yang mencoba menjalankan *task scenario*. Merujuk pada Tabel 5, rata-rata persentase keberhasilan partisipan untuk menyelesaikan *task scenario* memperoleh nilai sebesar 94%. Hal ini disebabkan karena dua partisipan, yakni P2 dan P3 gagal menyelesaikan dua *task scenario* yang mana adalah *task scenario* dengan kode TG6 dan TG7. Pada TG6, partisipan gagal melakukan pembayaran dikarenakan saat hendak melakukan simulasi pembayaran, nomor *virtual account* yang diperlukan tidak muncul. Dikarenakan TG6 gagal untuk dilakukan, partisipan tidak bisa melakukan *task scenario* TG7 karena pada *task* ini, partisipan diminta untuk menyimpan atau membagikan *bill* atas transaksi yang sudah dilakukan pada *task* sebelumnya. Untuk rata-rata persentase keberhasilan partisipan, website Cafeasy mendapat tingkat pencapaian yang sangat efektif, karena memiliki rasio di atas 80%. Selanjutnya dari sisi efektivitas, dapat dilihat pada Tabel 6, rata-rata waktu yang dibutuhkan partisipan untuk menjalankan *task scenario* yang diberikan adalah 0,117 detik. Berdasarkan interpretasi interval waktu pada indikator *time behavior,* rata-rata waktu yang didapat oleh partisipan dalam menyelesaikan sembilan *task scenario* memperoleh kualifikasi yang sangat cepat. Dari aspek kualitatif, didapat beberapa rekomendasi untuk perbaikan fitur edit catatan agar pengalaman pengguna dalam menggunakan website Cafeasy bisa lebih baik lagi. Dengan rata-rata Skor SUS sebesar 81,4, *website* Cafeasy masuk ke dalam *grade* A yang memiliki rating *excellent* yang berarti *website* sudah dapat memberikan pengalaman pengguna yang baik.

V. KESIMPULAN

A. Kesimpulan

Berdasarkan hasil analisis pada bab sebelumnya, dapat diambil kesimpulan untuk menjawab tujuan penelitian, yakni sebagai berikut.

1. Dari 74 *test case* yang terdefinisikan, semua *test case* berhasil dijalankan dengan status *passed* yang berarti sebanyak 100% *test case* unit yang dideskripsikan sudah berjalan dengan *product backlog* yang terdefinisikan.
2. Modul-modul seperti *log in, sign up*¸ data menu, data kategori, data pelanggan, data transaksi dan Google Spreadsheet sudah saling berinteraksi sesuai dengan alur bisnis aplikasi.
3. Untuk pengukuran *usability* dari website Cafeasy pada tingkat efektivitas mendapatkan hasil sebesar 94% yang berarti website sudah sangat efektif untuk digunakan. Sedangkan dari segi efisiensi, diperoleh hasil dengan rata-rata kecepatan 0,117 detik untuk setiap *task scenario* yang berarti tingkat pencapaian pengerjaannya adalah sangat cepat. Dari sisi kualitatif, didapat beberapa rekomendasi perbaikan pada fitur edit pesanan agar fitur tersebut dapat dengan mudah digunakan oleh pengguna. Serta evaluasi pengalaman pengguna yang didapat dari *usability testing* dengan skor SUS 81,4 menyatakan bahwa sistem sudah memiliki tingkat usabilitas yang tinggi serta sistem sudah efektif memenuhi kebutuhan dan harapan dari pengguna.

B. Saran

Berdasarkan hasil yang diperoleh, terdapat saran yang dapat diberikan untuk meningkatkan hasil pengujian, yakni menggunakan pendekatan *coverage* lain dalam melakukan *unit testing* untuk memastikan lebih jauh bahwa kualitas perangkat lunak yang diuji sudah baik.

REFERENSI

[1] R. B. Trengginaz, A. Yusup, D. S. Sunyoto, M. R. Jihad, and Y. Yulianti, "Pengujian Aplikasi Pemesanan Tiket Kereta berbasis Website Menggunakan Metode Black Box dengan Teknik Equivalence Partitioning," *Jurnal Teknologi Sistem Informasi dan Aplikasi*, vol. 3, no. 3, p. 144, Aug. 2020, doi: 10.32493/jtsi.v3i3.5349.

[2] B. P. Pratama, I. B. Vitriadi Ristianto, I. A. Prayogo, Nasrullah, and A. Saifudin, "Pengujian Perangkat Lunak Sistem Informasi Penilaian Mahasiswa dengan Teknik Boundary Value Analysis Menggunakan Metode Black Box Testing," Journal of Artificial Intelligence and Innovative Applications, vol. 1, no. 1, Feb. 2020.

[3] L. Liana, "Pengujian Perangkat Lunak (Software Testing)," Universitas Mercu Buana, Jakarta, 2015.

[4] T. A. Kurniawan, "Pengujian Struktur Program Dengan Pengujian Jalur Dasar (Basis Path Testing) : Teori Dan Aplikasi," *Jurnal EECCIS (electrics, electronics, communications, controls, informatics, systems)*, vol. 1, no. 1, p. 29, Jun. 2016, doi: 10.21776/jeeccis.v1i1.357.

[5] F. C. Ningrum, D. Suherman, S. Aryanti, H. A. Prasetya, and A. Saifudin, "Pengujian Black Box pada Aplikasi Sistem Seleksi Sales Terbaik Menggunakan Teknik Equivalence Partitions," *Jurnal Informatika Universitas Pamulang*, vol. 4, no. 4, p. 125, Dec. 2019, doi: 10.32493/informatika.v4i4.3782.

[6] M. E. Khan, "Different Approaches to White Box Testing Technique for Finding Errors," *International Journal of Software Engineering and Its Applications*, vol. 5, no. 3, Jul. 2011.

[7] M. Vladov, *Unit Testing: The Complete Guide*. Progress Software Corporation, 2020.

[8] M. E. Delamaro, J. C. Maidonado, and A. P. Mathur, "Interface Mutation: an approach for integration testing," *IEEE Transactions on Software Engineering*, vol. 27, no. 3, pp. 228–247, Mar. 2001, doi: 10.1109/32.910859.

[9] J. A. Whittaker, "What is software testing? And why is it so hard?," *IEEE Softw*, vol. 17, no. 1, pp. 70–79, 2000, doi: 10.1109/52.819971.

[10] S. K. Singh and A. Singh, *Software Testing*. Vandana Publications, 2012.

[11] M. Kumar, S. K. Singh, and R. K. Dwivedi, "A Comparative Study of Black Box Testing and White Box Testing Techniques," *International Journal of Advance Research in Computer Science and Management Studies (IJARCSMS)*, vol. 3, no. 10, Oct. 2015.

[12] P. Kaur, "A Research Paper on White Box Testing," *World Wide Journal of Multidisciplinary Research and Development*, 2018.

[13] T. Hamilton, "What is Usability Testing? Software UX," *guru99*, 2023. https://www.guru99.com/usability-testing-tutorial.html (accessed May 11, 2023).

[14] "A Beginner's Guide to Usability Testing." https://maze.co/guides/usability-testing/ (accessed May 11, 2023).

[15] Z. Sharfina and H. B. Santoso, "An Indonesian adaptation of the System Usability Scale (SUS)," in *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, IEEE, Oct. 2016, pp. 145–148. doi: 10.1109/ICACSIS.2016.7872776.

[16] S. Park *et al.*, "CarFast: Achieving Higher Statement Coverage Faster," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, New York, NY, USA: ACM, Nov. 2012, pp. 1–11. doi: 10.1145/2393596.2393636.