COLLEGGTION

Lecturer: Dr. Zulfany Erlisa Rasjid, B.Sc., MMSI.

Disusun Oleh:
Audrey Tabitha Ariani - 2440082812
Patrick Jonathan - 2440064791
Jocelyn Wievin - 2440063025
Samuel Sebastian Christlie - 2440085316

COMPUTER SCIENCE AND STATISTICS
SCHOOL OF COMPUTER SCIENCE
BINUS UNIVERSITY
2023

I. Latar Belakang

Software Colleggtion adalah *game* yang bertujuan untuk membawa hiburan kepada pengguna. Ada sekitar 3.09 miliar pemain *game* aktif tersebar di seluruh dunia dan diproyeksikan untuk berkembang hingga 3.32 miliar pengguna pada tahun 2024. Jumlah tersebut menunjukkan bahwa ada sangat banyak peminat *video game* dan ada banyak kesempatan untuk mengembangkan software *gaming* untuk bersaing di pasar.

Perkembangan *game* yang sangat pesat dalam beberapa tahun belakang tidak hanya memberikan nilai hiburan bagi penggemarnya tetapi juga mencetus kesempatan untuk menghasilkan pendapatan moneter. Maka bisa dilihat bahwa suatu hal menarik dapat diperoleh apabila sebuah game dirancang khusus untuk dapat menghasilkan pendapatan moneter dengan sendirinya. Dengan begitu, pengguna dapat menghasilkan pendapatan bersamaan dengan hiburan.

Walaupun ada banyak cara untuk mencari pendapatan secara digital seperti *trading* saham, *cryptocurrency*, dan lain-lain, tetapi tingkat resiko yang dimiliki oleh metode tersebut bisa diklasifikasikan sebagai tinggi, dan membutuhkan komitmen yang besar untuk mempelajari ilmu tersebut. Hal tersebut menyebabkan kecemasan bagi banyak orang yang tertarik untuk menghasilkan pendapatan secara digital tanpa *entry barrier* yang tinggi. Colleggtion dirancang agar tidak memiliki resiko tinggi yang terkait dengan aktivitas *trading* yang ada di game.

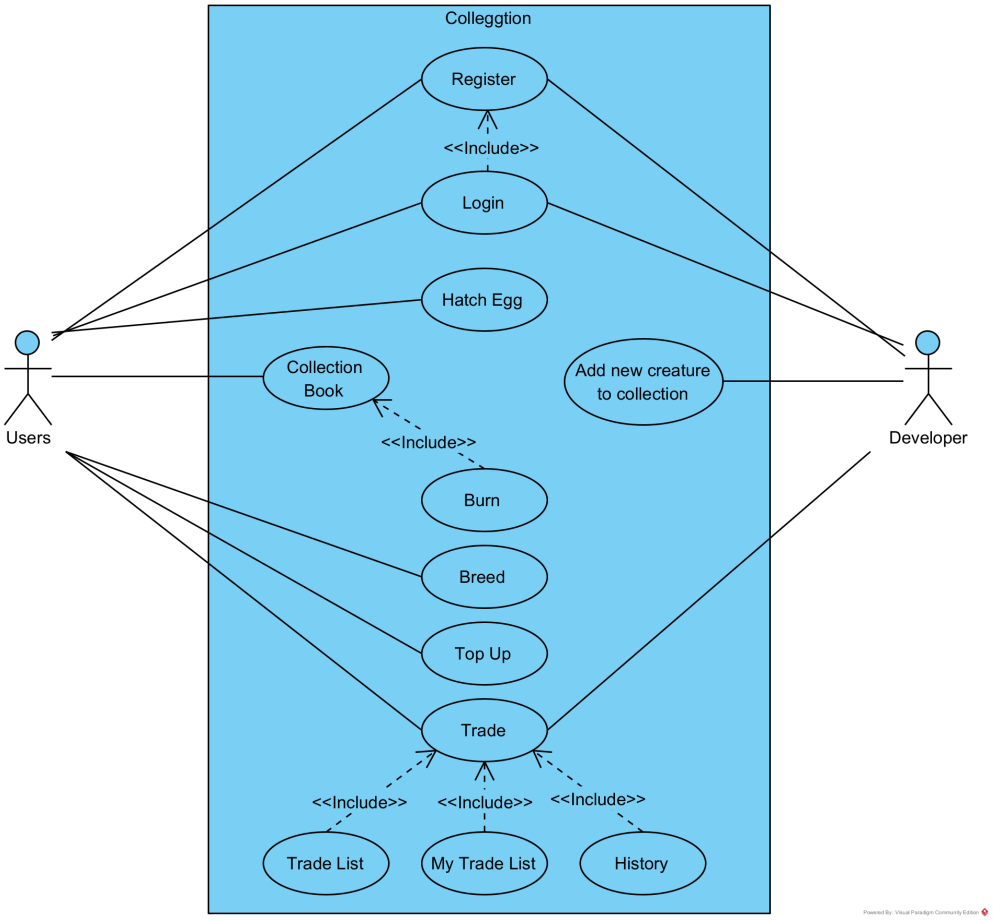
Game ini dirancang untuk memiliki game sederhana agar dapat dimainkan dalam jangka waktu pendek (5-10 menit). Ada pertimbangan bahwa game ini seharusnya dapat dimainkan dengan mudah agar tetap cocok dengan pengguna dengan gaya hidup yang sibuk, maka hanya mencakup beberapa menit saja dengan usaha rendah hingga moderat. Hal ini dapat menguntungkan pemain game serta mereka yang jarang bermain. Ada banyak game yang dapat ditemukan di pasar yang membutuhkan waktu lama untuk dimainkan setiap babaknya (>= 25 menit).

Sebagai kesimpulan kami memilih untuk menciptakan software ini agar dapat memberikan sebuah pengalaman bermain yang menyenangkan, peluang untuk menghasilkan uang, kemampuan bermain dalam jangka waktu pendek, hingga resiko rendah, "Colleggtion" memberikan kombinasi yang menarik bagi para penggemar game yang ingin mendapatkan hiburan dan kesempatan finansial sekaligus.

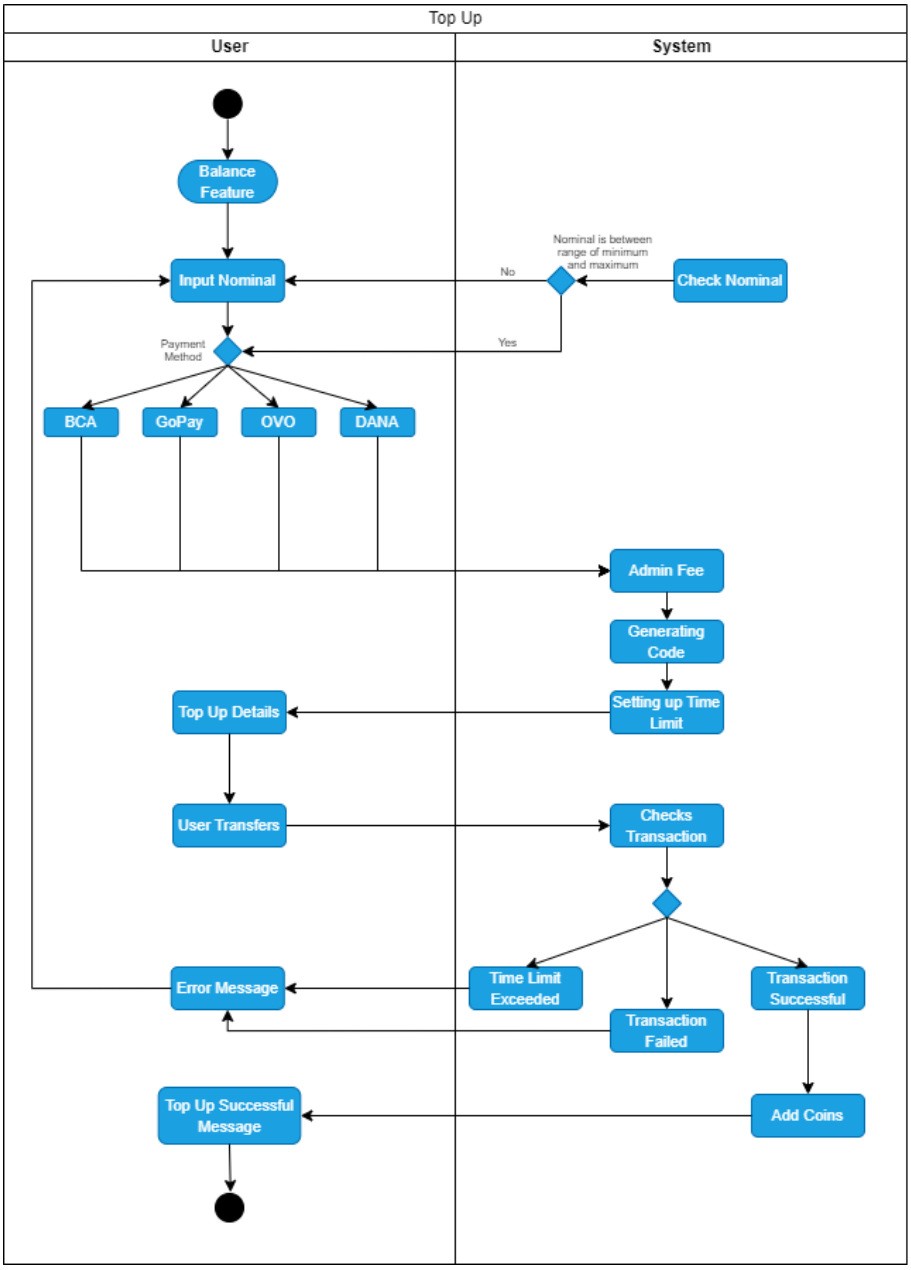
Fitur unik yang ditawarkan dalam software ini dinamakan “Breed”. Fungsi dari fitur ini sendiri adalah untuk menggabungkan dua hewan yang memiliki level masing-masing untuk menghasilkan hewan baru dengan level lebih tinggi dari atau sama dengan jumlah level kedua orang tuanya. Pemain dapat menggunakan fitur ini apabila mereka ingin mendapatkan hewan baru dengan level yang lebih tinggi.

II. Deskripsi Software

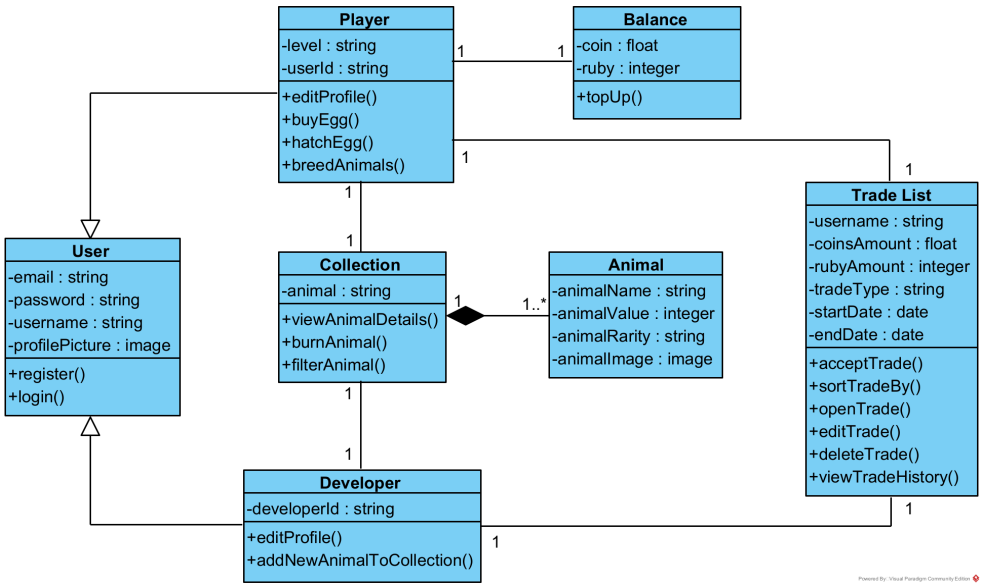
- Use Case Diagram



- Activity Diagram



- Class Diagram



III. Proses Model

Process model yang digunakan dalam *game* ini adalah model Prototype. Model Prototype merupakan proses yang menghasilkan prototipe yang dapat mereplikasi aplikasi/web serta sistem agar user dapat memberi feedback kepada tim developer.

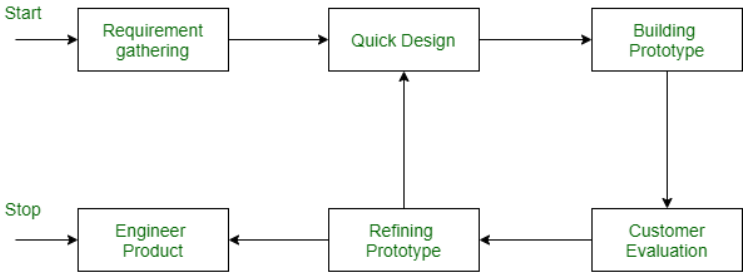


Figure - Prototype Model

Setelah mengeluarkan prototype pertama, feedback seperti tambahan desain dan preferensi yang lebih cocok dengan user dapat diterima dengan lebih baik, dan juga masalah-masalah yang ada dapat teridentifikasi di awal. Game ini juga memiliki fitur-fitur penting seperti transaksi top up dan trading, yang akan sangat membutuhkan pengetesan dan feedback dari user untuk memastikan fitur berjalan dengan baik. Dari interaksi inilah di mana developer dapat meningkatkan kelancaran aplikasi dan juga keserasiannya dengan preferensi user setiap iterasinya.

Penggunaan model Prototype ini dipilih karena memungkinkan fleksibilitas dengan perubahan-perubahan yang nantinya dibutuhkan. Perubahan ini dapat berupa perbaikan atau penambahan fitur pada software untuk meningkatkan kepuasan user, yang merupakan prioritas tim kami. Selain itu, model ini memudahkan deteksi fungsionalitas

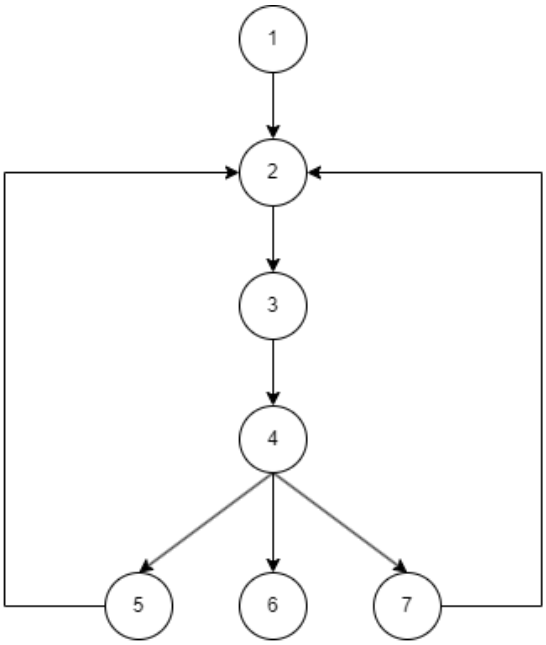
yang hilang karena dievaluasi oleh user secara langsung. Kita memprioritaskan kepuasan user dibandingkan kecepatan penyelesaian proyek (efisiensi waktu) karena Colleggtion tidak memiliki banyak pesaing di kategori game ini maka ada lebih banyak ruang untuk memastikan game ini menarik perhatian user yang nantinya menjadi loyal kepada game ini.

IV. Cyclomatic Complexity

Berikut adalah pseudocode untuk fitur “Balance”:

1. User masuk ke dalam fitur “Balance”
2. Selanjutnya, user akan diminta untuk menginput nominal top-up yang diinginkan
3. Kemudian tampilan akan diarahkan menuju top-up details, dimana kita bisa melihat jumlah yang harus dibayarkan berikut dengan biaya admin, kode pembayaran, serta batas waktu untuk melakukan pembayaran
4. User melakukan pembayaran
5. Transaksi gagal dan sistem akan menampilkan pesan error bahwa pembayaran tidak berhasil
6. Transaksi berhasil dan koin sudah berhasil ditambahkan
7. Batas waktu pembayaran sudah lewat dan sistem akan menampilkan pesan error bahwa pembayaran tidak berhasil.

Berikut adalah flow graph untuk fitur “Balance”:



Cyclomatic Complexity : V(G) = Edge – Nodes + 2 = 8 − 7 + 2 = 3

Path :
Jalur ke-1 : 1 – 2 – 3 – 4 – 6
Jalur ke-2 : 1 – 2 – 3 – 4 – 5
Jalur ke-3 : 1 – 2 – 3 – 4 – 7

V. Effort Estimation

- Effort Estimation using Albrecht Function Point:

Menghitung Unadjusted Function Point untuk setiap modul : :
Login

	Login	Count	Simple	Average	Complex	Total
Inputs	UserID, Password	2	2	0	0	6
Outputs	Error message	1	0	1	0	5
Inquiries	Check login database	1	0	0	1	5
Internal Logical Files	Login database	1	0	1	0	10
External Interface Files	None	0	0	0	0	0
Sub Total (Point)						26

Buy Egg

	Buy Egg	Count	Simple	Average	Complex	Total
Inputs	Click "Buy" button, Click confirmation button	2	2	0	0	6
Outputs	Error message, Confirmation Message, Time Limit Countdowr	3	0	2	1	16
Inquiries	Check if ruby available, Subtract ruby, Add egg	3	0	3	0	12
Internal Logical Files	Ruby database, Egg database	2	0	2	0	20
External Interface Files	None	0	0	0	0	0
Sub Total (Point)						54

Collection Book

	Collection Book	Count	Simple	Average	Complex	Total
Inputs	None	0	0	0	0	0
Outputs	Animal list	1	0	1	0	5
Inquiries	Check animal database	1	0	1	0	4
Internal Logical Files	Animal database	1	0	1	0	10
External Interface Files	None	0	0	0	0	0
Sub Total (Point)						19

Breed

	Breed	Count	Simple	Average	Complex	Total
Inputs	First Animal, Second Animal	2	0	2	0	8
Outputs	Bred Animal	1	0	0	1	6
Inquiries	Check player's animal database	1	0	1	0	4
Internal Logical Files	Animal Database	1	0	1	0	10
External Interface Files	Animal Images Folder	1	0	1	0	7
Sub Total (Point)						35

Trade

	Trade	Count	Simple	Average	Complex	Total
Inputs	Amount of Ruby, Price	2	0	2	0	8
Outputs	Trade Result	1	1	0	0	4
Inquiries	Check Trade list, Check user's balance	2	0	2	0	8
Internal Logical Files	Trade database, user balance database	2	0	2	0	20
External Interface Files	None	0	0	0	0	0
Sub Total (Point)						40

Top Up

	Top up	Count	Simple	Average	Complex	Total
Inputs	Amount of currency, Payment Method	2	0	2	0	8
Outputs	Top up result	1	1	0	0	4
Inquiries	Validate payment method, Insert into user's balance	2	0	2	0	8
Internal Logical Files	User balance database	1	0	1	0	10
External Interface Files	None	0	0	0	0	0
Sub Total (Point)						30

Profile

	Profile	Count	Simple	Average	Complex	Total
Inputs	Name, Bio, Profile	3	0	3	0	12
Outputs	Name, Bio, Profile, Collections	3	0	3	0	15
Inquiries	Check user profile, update user profile	2	0	0	2	10
Internal Logical Files	User database	1	0	1	0	10
External Interface Files	User profile folder	1	0	1	0	7
Sub Total (Point)						54

Complexity Adjustment Value:

Complexity Adjustment Value	
1. Does the system require reliable backup and recovery?	3
2. Are data communications required?	2
3. Are there distributed processing functions?	2
4. Is performance critical?	2
5. Will the system run in an existing, heavily utilized operational environment?	1
6. Does the svstem require on-line data entry?	4
7- Does the on-line data entry require the input transaction to be built over multiple screens or operator?	0
8. Are the master files updated on-line?	5
9. Are the inputs outputs, files, or inquiries complex?	5
10. Is the internal processing complex?	5
11. Is the code designed to be reusable?	1
12. Are conversion and installation included in the design?	1
13. Is the system designed for multiple installations in different organizations?	0
14. Is the application designed to facilitate change and ease of use by the user?	4

Total Adjusted Function Point:

Total Unadjusted Function Point	258
Total Complexity Adjustment Value	35
FP = Total Unadjusted FP * [0,65 + 0,01*(Total Comp Adj Value)]	
Total Adjusted Function Point	258

Gunakan metode COCOMO (Constructive Cost Model) untuk menghitung usaha yang diperlukan dengan menggunakan kategori Organic Software Projects karena ukuran tim yang cukup kecil, serta kurangnya pengalaman yang dimiliki oleh anggota tim mengenai masalah tersebut.

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Model COCOMO dapat dihitung dengan cara seperti berikut:

$$Effort = a(Final\ Function\ Point)^{b}$$
$$time = c(Effort)^{d}$$
$$PersonRequired = \frac{Effort}{time}$$

Maka kita dapat memperoleh perhitungan sebagai berikut:

$$Effort = 2.4(258)^{1.02} = 691.93\ person - month$$
$$time = 2.5(691.93)^{0.38} = 30.003\ month$$
$$PersonRequired = \frac{691.93\ person-month}{30.003\ month} = 23.062\ person$$

- Effort Estimation using Kilo Lines of Code (KLOC):

Selanjutnya kita lakukan perhitungan Effort Estimation dengan menggunakan Kilo Lines of code dari model COCOMO menggunakan kategori Organic Software Projects karena ukuran tim yang cukup kecil, serta kurangnya pengalaman yang dimiliki oleh anggota tim mengenai masalah tersebut. Dimana didapatkan KLOC setelah perhitungan yaitu 1.216

Model COCOMO dapat dihitung dengan cara seperti berikut:

$$Effort = a(KLOC)^{b}$$
$$time = c(Effort)^{d}$$
$$PersonRequired = \frac{Effort}{time}$$

Maka kita dapat memperoleh perhitungan sebagai berikut:

$$Effort = 2.4(1.216)^{1.02} = 2.93\ person - month$$
$$time = 2.5(2.93)^{0.38} = 3.76\ month$$
$$PersonRequired = \frac{2.93\ person-month}{3.76\ month} = 0.779\ person$$

Kesimpulan :
Jika kita bandingkan hasil Effort Estimation dari Function Point dan hasil Effort Estimation dari Kilo Lines of Code, dapat dilihat bahwa nilainya terbilang cukup berbeda jauh, dimana hasil Effort Estimation dari Kilo Lines of Code jauh lebih kecil dari nilai Effort Estimation menggunakan Function Point. Hal ini kemungkinan besar dikarenakan code yang hanya dibuat oleh beberapa orang saja, serta dibuat dalam waktu yang terbatas sehingga masih belum sempurna.

VI. Test case untuk Software Testing

Test Case 1 : Transaksi berhasil dan koin sudah berhasil ditambahkan

a. Description
   User berhasil melakukan transaksi dan koin telah berhasil ditambahkan
b. Steps
   1. User masuk ke dalam fitur “Balance”
   2. Selanjutnya, user akan diminta untuk menginput nominal top-up yang diinginkan
   3. Kemudian tampilan akan diarahkan menuju top-up details, dimana kita bisa melihat jumlah yang harus dibayarkan berikut dengan biaya admin, kode pembayaran, serta batas waktu untuk melakukan pembayaran
   4. User melakukan pembayaran
   5. Transaksi berhasil dan koin sudah berhasil ditambahkan
c. Test Input
   User menginput nominal top-up yang diinginkan
d. Expected Result
   Transaksi berhasil dan koin berhasil ditambahkan ke dalam akun user
e. Actual Result
   Transaksi berhasil dan koin berhasil ditambahkan ke dalam akun user

Test Case 2 : Transaksi gagal dan sistem akan menampilkan pesan error bahwa pembayaran tidak berhasil

a. Description
   Transaksi gagal dikarenakan masalah jaringan dan sistem menampilkan pesan error bahwa pembayaran tidak berhasil
b. Steps
   1. User masuk ke dalam fitur “Balance”
   2. Selanjutnya, user akan diminta untuk menginput nominal top-up yang diinginkan
   3. Kemudian tampilan akan diarahkan menuju top-up details, dimana kita bisa melihat jumlah yang harus dibayarkan berikut dengan biaya admin, kode pembayaran, serta batas waktu untuk melakukan pembayaran
   4. User melakukan pembayaran
   5. Transaksi gagal dan sistem akan menampilkan pesan error bahwa pembayaran tidak berhasil
c. Test Input

User menginput nominal top-up yang diinginkan

d. Expected Result
   Transaksi gagal dan sistem menampilkan pesan error bahwa pembayaran tidak berhasil
e. Actual Result
   Transaksi gagal dan sistem menampilkan pesan error bahwa pembayaran tidak berhasil

Test Case 3 : Batas waktu pembayaran sudah lewat dan sistem akan menampilkan pesan error bahwa pembayaran tidak berhasil.

a. Description
   Setiap transaksi memiliki batas waktu untuk melakukan pembayaran, jika user tidak melakukan pembayaran selama batas waktu tersebut, maka transaksi gagal.
b. Steps
   1. User masuk ke dalam fitur “Balance”
   2. Selanjutnya, user akan diminta untuk menginput nominal top-up yang diinginkan
   3. Kemudian tampilan akan diarahkan menuju top-up details, dimana kita bisa melihat jumlah yang harus dibayarkan berikut dengan biaya admin, kode pembayaran, serta batas waktu untuk melakukan pembayaran
   4. User melakukan pembayaran
   5. Batas waktu pembayaran sudah lewat dan sistem akan menampilkan pesan error bahwa pembayaran tidak berhasil
c. Test Input
   User menginput nominal top-up yang diinginkan
d. Expected Result
   Kode pembayaran sudah tidak berlaku diluar batas waktu pembayaran sehingga sistem akan menampilkan pesan error bahwa kode pembayaran sudah tidak berlaku
e. Actual Result
   Kode pembayaran sudah tidak berlaku diluar batas waktu pembayaran sehingga sistem akan menampilkan pesan error bahwa kode pembayaran sudah tidak berlaku

VII. Analisa Resiko Proyek

Berikut adalah tabel hasil analisa resiko proyek dari aplikasi Colleggtion :

No	Risk	Category	Probabilty	Impact	Mitigation	Monitoring	Management
1	Kurangnya minat pemain	Customer Characteristics	35%	1	Membuat game menarik bagi pemain	Player count, transaction count	Membuat fitur yang lebih menarik dan melakukan balancing mechanic permainan berdasarkan minat pemain
2	Data leaking	Development Environment	5%	2	Membuat game yang aman dan menggunakan teknologi keamanan terkini	Lalu lintas jaringan, activity log, bandwidth, security log	Mengidentifikasi sumber data leak dan memberitahu pemain agar dapat mengganti password dan hal-hal sensitif lainnya. Meningakatkan keamanan dari permainan
3	Kesalahan transaksi pengguna tidak terekam	Business Impact	1%	2	Menggunakan payment gateway yang terpercaya dan dapat diandalkan	Transaction log, laporan dari pemain	Menggantikan jumlah Ruby yang dibeli pemain yang dirugikan setelah mengkonfirmasi laporan
4	Pemain terlalu banyak	Product Size	40%	3	Membuat sistem yang scalable dan dapat menangani jumlah player yang	Player count	Meningkatkan kapasitas server dan melakukan optimisasi pada permainan
5	Mekanisme permainan yang tidak seimbang	Development Environment	25%	2	Mengecek mekanisme permainan dan harga item secara menyeluruh dan memastikan bahwa mekanisme seimbang	Mengawasi laporan dari pemain, activity log, melakukan testing secara berkala	Melakukan penyeimbangan pada mekanisme permainan dan harga item dan meminta feedback dari pemain
6	Pemain menghabiskan uang terlalu banyak dalam game	Business Impact	50%	4	Menerapkan sistem cooldown pada permainan	Transaction log	Memperpanjang waktu cooldown dan melakukan perubahan agar permainan tidak terlalu addictive

Table 1: Risk Table

VIII. Video Demonstrasi Aplikasi

Berikut adalah video demonstrasi dari aplikasi Colleggtion :
https://binusianorg-my.sharepoint.com/personal/audrey_ariani_binus_ac_id/_layouts/15/guestaccess.aspx?docid=037dafab08102415fb707cca1d55bc416&authkey=AVfSzqjMqmJnAEe2TTlPYDI&e=ULpYRJ

IX. Link Github Aplikasi

Aplikasi ini dirancang berbasis web dengan menggunakan Laravel. Berikut adalah link github yang dapat diunduh untuk menjalankan aplikasi Colleggtion :
https://github.com/AudreyTabithaAriani/Colleggtion/tree/master

X. Penutup / Referensi / Simpulan

Software Colleggtion adalah *game* yang memiliki konsep trading dimana kita dapat memperjualbelikan mata uang dalam game tersebut, sehingga dapat menghasilkan keuntungan. Beberapa fitur yang terdapat dalam aplikasi Colleggtion yaitu, home, book, breed, trade, top-up, dan profile. Dalam laporan ini, kami juga telah mencantumkan deskripsi software, yang berisi use case diagram, activity diagram, serta class diagram. Selain itu, kami menggunakan proses model Prototype, yaitu proses yang menghasilkan prototipe yang dapat mereplikasi aplikasi/web serta sistem agar user dapat memberi feedback kepada tim developer. Terakhir, kami juga memberikan penjelasan software lainnya seperti cyclomatic complexity, effort estimation, software testing, serta analisa resiko proyek. Kami berharap dengan adanya aplikasi ini, dapat memberikan kesempatan bagi pengguna untuk menikmati pengalaman bermain game sekaligus menghasilkan keuntungan melalui trading antar pengguna dengan resiko yang cukup kecil.