LAPORAN TUGAS BESAR
IF1210 DASAR PEMROGRAMAN
TUGAS BESAR SISTEM INVENTORI BNMO

Disusun Oleh:

Kelas K04 – Kelompok 4

Kenneth Ezekiel Suprantoni	16521040
M. Bharata Sri Prana Ludira H.	16521148
Melvin Kent Jonathan	16521247
Noel Christoffel Simbolon	16521355

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

TAHUN 2022

HALAMAN PERNYATAAN KELOMPOK

“Saya menyatakan bahwa saya mengerjakan tugas besar ini dengan sejujur-jujurnya, tanpa menggunakan cara yang tidak dibenarkan. Apabila di kemudian hari diketahui saya mengerjakan tugas besar ini dengan cara yang tidak jujur, saya bersedia mendapatkan konsekuensinya, yaitu mendapatkan nilai E pada mata kuliah IF1210 Dasar Pemrograman Semester 2 2020/2021.”

Kenneth Ezekiel (16521040)

M. Bharata Sri Prana Ludira H. (16521148)

Melvin Kent Jonathan (16521247)

Noel Christoffel Simbolon (16521355)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

I. DESKRIPSI PERSOALAN

Masalah yang diberikan untuk kami pecahkan dalam Tugas Besar IF1210 Dasar Pemrograman ini pada intinya adalah menemukan suatu cara untuk mengambil data dari sebuah database, memodifikasinya, lalu menyimpannya kembali kedalam database tanpa menggunakan library, alias manipulasi data tanpa library. Dimana modul-modul yang diberikan kebanyakan adalah cara-cara untuk sorting, modifikasi data, menambahkan atau mengurangi data, dan juga menyimpan data itu sendiri. Dalam background permasalahannya disebutkan bahwa kami harus memperbaiki BNMO yang rusak dibanting oleh Indra yang rugi karena *gacha*. Dalam merancang program utama untuk memecahkan permasalahan ini pun, dibutuhkan teknik programming *modular programming*, dimana sebuah program besar dapat dipecah-pecah menurut fungsionalitas-fungsionalitasnya.

F02 – Register

Subprogram register digunakan untuk menambahkan User ke database BNMO, lebih tepatnya ke user.csv. Subprogram ini hanya dapat diakses oleh Admin, dan pengguna yang ditambahkan menggunakan subprogram ini hanya dapat berupa User. Tidak bisa menambahkan pengguna dengan role Admin menggunakan subprogram ini. Untuk menambahkan pengguna dengan role Admin, dapat langsung mengedit database user.csv.

F03 – Login

Subprogram login bertugas untuk mengecek kevalidan login data yang diinput oleh pengguna. Login data tersebut berupa username dan password. Subprogram ini akan mengembalikan data bertipe boolean yang bernilai True jika username dan password benar dan terdapat pada database.

F04 – Menambah Game ke Toko Game

Subprogram ini digunakan untuk menambahakan game ke database BNMO, lebih tepatnya ke game.csv, dan hanya dapat diakses oleh Admin. Subproram akan meminta masukan atribut dari game yang perlu diinput, yakni nama, kateogri, tahun rilis, harga, dan stok awal. Apabila terdapa atribut yang belum diisi oleh Admin, akan dilakukan validasi berupa penampilan pesan error dan subprogram akan meminta input ulang. Validasi akan dilakukan berulang hingga atribut telah terisi semua.

F05 – Mengubah Game pada Toko Game

Subprogram ini digunakan untuk mengubah data game dalam database BNMO, lebih tepatnya ke game.csv, dan hanya dapat diakses oleh Admin. Subproram akan meminta masukan ID dari game yang datanya ingin diubah, lalu melakukan validasi bahwa apakah ID tersebut ada dalam database. Apabila tidak ada, subprogram akan mencetak pesan kesalahan. Apabila ada, Admin dapat mengbubah atribut nama, kategori, tahun rilis, dan harga. Admin tidak perlu mengisi semua field selain field ID, sehingga apabila tidak ingin mengubah field tertentu dari suatu game, admin dapat mengosongkannya dan value field tersebut tidak akan berubah.

F06 – Mengubah Stok Game di Toko

Subprogram ini digunakan untuk mengubah stok game dalam database BNMO, lebih tepatnya ke game.csv, dan hanya dapat diakses oleh Admin. Subproram akan meminta masukan ID dari game yang datanya ingin diubah, lalu melakukan validasi bahwa apakah ID tersebut ada dalam database. Apabila tidak ada, subprogram akan mencetak pesan kesalahan. Apabila ada, Admin akan diminta masukan jumlah perubahan stok yang ingin dilakukan. Masukan jumlah akan divalidasi oleh program agar stok game tetap valid setelah pengubahan (tidak negatif). Apabila menjadi tidak valid, stok tidak akan berubah. Apabila stok menjadi nol, game tidak perlu dihapus dari sistem.

F07 – Listing Game di Toko Berdasarkan ID, Tahun Rilis dan Harga

Subprogram ini dapat diakses oleh Admin maupun User dan digunakan untuk menampilkan daftar game yang ada di toko, dengan skema *sorting* berdasarkan tahun rilis atau harga (urutan bisa *ascending* atau *descending*). Skema sorting dari input user akan divalidasi terlebih dahulu. Apabila user mengosongkan input skema, maka daftar game akan tampil terurut (*ascending*) berdasarkan ID nya. Apabila user memasukan skema sorting yang tidak valid, subprogram akan mencetak pesan error dan meminta masukan ulang. Validasi akan dilakukan berulang hingga masukan skema sorting benar.

F08 – Membeli Game

User dapat membeli Game dengan menggunakan prosedur ini. Game yang telah dibeli akan masuk ke list Game yang dimiliki User. Game hanya dapat dibeli user yang sama sebanyak satu kali. Terdapat 1 parameter yang wajib diisi pada prosedur ini, yaitu ID Game yang akan dibeli user.

F09 – Melihat Game yang dimiliki

Prosedur ini memberikan daftar game yang dimiliki pengguna. Tidak ada aturan khusus untuk urutan game yang ditampilkan. Tampilkan pesan khusus ketika user tidak memiliki game.

F10 – Mencari Game yang dimiliki dari ID dan tahun rilis

Subprogram ini dipecah lagi menjadi beberapa subprogram untuk dapat menyelesaikan fungsinya. Terdapat subprogram yang fungsinya untuk mem-filter data game sesuai dengan game_id serta release_year yang dimasukkan oleh pengguna. Selain itu, terdapat subprogram yang menghitung panjang karakter maksimum dari tiap kolom data game yang ingin di-output. Subprogram ini berguna untuk memperapi output data game. Subprogram satu lagi berguna untuk memfilter game yang dimiliki pengguna, sekaligus berperan sebagai subprogram utama dalam modul F10 ini.

F11 – Mencari Game di Toko dari ID, Nama Game, Harga, Kategori dan Tahun Rilis

Subprogram ini dapat diakses oleh admin dan user untuk mencari game berdasarkan masukan 5 parameter, yakni ID Game, nama game, harga, kategori, dan tahun rilis game. Parameter bersifat tidak wajib diisi. Apabila pada database tidak terdapat game yang sesuai, makan akan ditampilkan pesan bahwa tidak ada game yang cocok dengan parameter.

F12 – Top Up Saldo

Subprogram top up saldo digunakan untuk menambahkan/mengurangi data saldo pada database user.csv, yang digunakan dengan cara meminta user yang akan di topup dan jumlah saldo yang akan di topup, dimana topup hanya valid jika total akhir saldo user lebih besar dari 0. Hanya Admin yang dapat memanggil subprogram ini.

F13 – Melihat Riwayat Pembelian

Subprogram melihat riwayat pembelian digunakan untuk mengeluarkan riwayat pembelian dari sang pengguna, sehingga membutuhkan masukan yaitu user yang memanggilnya, dimana subprogram ini hanya bisa dipanggil oleh user karena Admin tidak bisa membeli game. Jika user tidak pernah membeli game, riwayatnya akan kosong.

F14 – Help

Subprogram help digunakan untuk mengeluarkan list dari fungsi-fungsi yang dapat dipanggil, tergantung dari role user, apakah user biasa atau Admin.

F15 – Load

Subprogram load digunakan untuk inisialiasi folder yang akan dijadikan working database yang akan dimanipulasi oleh main program. Dimana save folder yang digunakan harus valid (termasuk dalam folder database yang sudah ada)

F16 – Save

Subprogram save digunakan untuk menyimpan data dari working database yang dimodifikasi oleh program kedalam file csv nya, dimana jika program ditutup sebelum memanggil subprogram save, data yang telah termodifikasi tidak akan masuk kedalam database dan akan hilang.

F17 – Exit

Seperti namanya, fungsi ini adalah fungsi untuk keluar dari aplikasi. Fungsi dapat menerima huruf kecil maupun besar. Pastikan masukan valid. Kalau tidak valid, bisa tanyakan kembali pertanyaannya.

II. DAFTAR PEMBAGIAN KERJA ANGGOTA KELOMPOK

Table 1 Daftar Pembagian Kerja Anggota Kelompok

MODUL	IMPLEMENTASI	CODER	DESIGNER	TESTER
F02	**Function** register	Noel Christoffel Simbolon (16521355)	Noel Christoffel Simbolon (16521355)	Kenneth Ezekiel Suprantoni (16521040)
F03	**Function** login	Noel Christoffel Simbolon (16521355)	Noel Christoffel Simbolon (16521355)	Kenneth Ezekiel Suprantoni (16521040)
F04	**Function** indeks_constructor **Function** new_game **Function** add_game	Melvin Kent Jonathan (16521247)	Melvin Kent Jonathan (16521247)	M. Bharata Sri Prana Ludira H. (16521148)
F05	**Procedure** change_game	Melvin Kent Jonathan (16521247)	Melvin Kent Jonathan (16521247)	M. Bharata Sri Prana Ludira H. (16521148)
F06	**Procedure** change_stock	Melvin Kent Jonathan (16521247)	Melvin Kent Jonathan (16521247)	M. Bharata Sri Prana Ludira H. (16521148)
F07	**Function** temporary_data **Function** modes **Function** sorting	Melvin Kent Jonathan (16521247)	Melvin Kent Jonathan (16521247)	M. Bharata Sri Prana Ludira H. (16521148)
F08	**Function** filter_str **Function** filter_int **Procedure** buy_game	M. Bharata Sri Prana Ludira H. (16521148)	M. Bharata Sri Prana Ludira H. (16521148)	Noel Christoffel Simbolon (16521355)
F09	**Procedure** list_game	M. Bharata Sri Prana Ludira H. (16521148)	M. Bharata Sri Prana Ludira H. (16521148)	Noel Christoffel Simbolon (16521355)
F10	**Function** filter_str **Function** get_max_char_length **Procedure** search_my_game	Noel Christoffel Simbolon (16521355)	Noel Christoffel Simbolon (16521355)	Kenneth Ezekiel Suprantoni (16521040)
F11	**Function** filter_str **Function** filter_int **Procedure** search_game_at_store	Melvin Kent Jonathan (16521247)	Melvin Kent Jonathan (16521247)	M. Bharata Sri Prana Ludira H. (16521148)
F12	**Function** function_topup **Function** topup	Kenneth Ezekiel Suprantoni (16521040)	Kenneth Ezekiel Suprantoni (16521040)	Melvin Kent Jonathan (16521247)
F13	**Procedure** history	Kenneth Ezekiel Suprantoni (16521040)	Kenneth Ezekiel Suprantoni (16521040)	Melvin Kent Jonathan (16521247)
F14	**Procedure** help	Kenneth Ezekiel Suprantoni (16521040)	Kenneth Ezekiel Suprantoni (16521040)	Melvin Kent Jonathan (16521247)
F15	Load (automatically loaded)	Kenneth Ezekiel Suprantoni (16521040)	Kenneth Ezekiel Suprantoni (16521040)	Melvin Kent Jonathan (16521247)

F16	**Procedure** saver **Procedure** save	Kenneth Ezekiel Suprantoni (16521040)	Kenneth Ezekiel Suprantoni (16521040)	Melvin Kent Jonathan (16521247)
F17	**Procedure** exit	M. Bharata Sri Prana Ludira H. (16521148)	M. Bharata Sri Prana Ludira H. (16521148)	Noel Christoffel Simbolon (16521355)
B01	**Function** encrypt **Function** decrypt	Noel Christoffel Simbolon (16521355)	Noel Christoffel Simbolon (16521355)	Kenneth Ezekiel Suprantoni (16521040)
B02	**Function** magicconch	Kenneth Ezekiel Suprantoni (16521040)	Kenneth Ezekiel Suprantoni (16521040)	Melvin Kent Jonathan (16521247)
B03	**Function** ask_location **Function** win_checker **Function** status **Procedure** tictactoe	Melvin Kent Jonathan (16521247)	Melvin Kent Jonathan (16521247)	M. Bharata Sri Prana Ludira H. (16521148)

III. CHECKLIST

Table 2 Checklist Pengerjaan Modul

MODUL	DESAIN	IMPLEMENTASI	TESTING
F02	✓	✓	✓
F03	✓	✓	✓
F04	✓	✓	✓
F05	✓	✓	✓
F06	✓	✓	✓
F07	✓	✓	✓
F08	✓	✓	✓
F09	✓	✓	✓
F10	✓	✓	✓
F11	✓	✓	✓
F12	✓	✓	✓
F13	✓	✓	✓
F14	✓	✓	✓
F15	✓	✓	✓
F16	✓	✓	✓
F17	✓	✓	✓
B01	✓	✓	✓
B02	✓	✓	✓
B03	✓	✓	✓

IV. DESAIN COMMAND UNTUK SETIAP PRIMITIF

(berisi nama command, masukan, dan keluaran)

F02 – Register

command: register

input: user_data: array of array of string; name, username, password: string

output: array of array of string

F03 – Login

command: login

input: user_data: array of array of string; username, password: string

output: boolean

F04 - Menambah Game ke Toko Game

command: index_constructor

input: game_data : array of array of string

ouput: array of array of string

command: new_game

input: game_data : array of array of string

ouput: array of array of string

command: add_game

input: game_data : array of array of string

ouput: array of array of string

F05 - Mengubah Game pada Toko Game

command: change_game

input: game_data : array of array of string

ouput: none

F06 - Mengubah Stok Game di Toko

command: change_stock

input: game_data : array of array of string

ouput: array of array of string

F07 - Listing Game di Toko Berdasarkan ID, Tahun Rilis dan Harga

command: temporary_data

input: game_data : array of array of string

ouput: array of array of string

command: modes

input: game_data : array of array of string

ouput: array of array of string

command: sorting

input: game_data : array of array of string

ouput: none

F08 – Membeli game

command: filter_str

input: data : array of array of string, index : integer, criteria : string

ouput: array of array of string

command: filter_int

input: data : array of array of string, index : integer, criteria : string

ouput: array of array of string

command: buy_game

input: money : integer, game_data : array of array of string, my_game : array of array of string

output: none

F09 – Melihat Game yang dimiliki

command : list_game

input: game_data : array of array of string

output: none

F10 - Mencari Game yang dimiliki dari ID dan tahun rilis

command: filter_str

input: data: array of array of string; index: integer; criteria: string output: array of array of string

command: get_max_char_length

input: filtered_game_data: array of array of string

output: array of integer

command: search_my_game

input: ownership_data, user_data, game_daya: array of array of string; game_id, release_year: string

output: - (mengoutput hasil dari filtering ke layer pengguna)

F11 – Mencari Game di Toko dari ID, Nama Game, Harga, Kategori dan Tahun Rilis

command: filter_str

input: data : array of array of string, index : integer, criteria : string

ouput: array of array of string

command: filter_int

input: data : array of array of string, index : integer, criteria : string

ouput: array of array of string

command: search_game_at_store

input: game_data : array of array of string

ouput: none

F12 – Topup

Command : function_topup

Input : username : string, balance : integer, user_data : array of array

output : array of array

Command : topup

input data : array of array

output : array of array

F13 – History

Command : history

Input : hist_data : array of array

output : none

F14 – Help

Command : help

input user : string, save_folder : string

output : none

F15 – Load

Command : -

input : folder

output : none

F16 – Save

Command : saver

input : folder, data

output : none

Command : save

input : data

output : none

F17 – Exit

Command : exit

input : array of array of array of integer

output : -

V. DESAIN KAMUS DATA

KAMUS GLOBAL

running : boolean

filenames : array of string

data : array of array of array of string

admin_callable_commands : array of string

user_callable_commands : array of string

logged_in : boolean

command : string

hist_data : array of array

F02

<u>Function</u> register (user_data : array of array of string) -> array of array of string

KAMUS LOKAL

name : string

username : string

char : string

password : string

char_pass : string

id : integer

ciphered_password : string

role : string

balance : integer

new_user : array

F03

<u>Procedure</u> login (user_data : array of array of string) -> boolean

KAMUS LOKAL

username, password: string

user_valid: boolean

F04

<u>Function</u> index_constructor (game_data : array of array of string) -> array of array of string

KAMUS LOKAL

prevoius_number, new_number : integer

new_index : string

Function new_game (game_data : array of array of string) -> array of array of string

KAMUS LOKAL

complete : boolean

name, category, release_year : string

price, stock : integer

new_index : string

new_data : array of string

Function add_game (game_data : array of array of string) -> array of array of string

KAMUS LOKAL

new_data : array of string

game_data : array of array of string

F05

Procedure change_game (**input/output** game_data : array of array of string)

KAMUS LOKAL

id : string

found : boolean

i, line_index : integer

F06

Function change_stock(game_data : list) -> list

KAMUS LOKAL

id : string

found : boolean

i, line_index, added_stock : integer

F07

Function temporary_data(game_data : array of array of string) -> array of array of string

KAMUS LOKAL

data : array of array of string

i : integer

<u>Function</u> modes(game_data : array of array of string) -> array of array of string

KAMUS LOKAL

sorted, temp : array of array of integer

i, j, index_min : integer

<u>Procedure</u> sorting (**input** game_data : array of array of string)

KAMUS LOKAL

valid : boolean

mode : string

sorted, temp : array of array of string

header : array of string

i, j, k, character_amount : integer

F08

<u>Procedure</u>

KAMUS LOKAL

ID : string

F09

<u>Procedure</u>

KAMUS LOKAL

None

F10 - Mencari Game yang dimiliki dari ID dan tahun rilis

<u>Function</u> filter_str (data : array of array of string, index : integer, criteria : string) -> array of array of string

KAMUS LOKAL

temp : array of array of string

i : integer

<u>Function</u> get_max_char_length (filtered_game_data : array of array of string) -> array of integer

KAMUS LOKAL

filtered_game_data_char_length : array

char_length_list : array

L : array of string

m, n : integer

max_length_of_column : integer

o : array

filtered_game_data_max_char_length : array

Procedure **search_my_game** (ownership_data, user_data, game_data : array of array of string)

KAMUS LOKAL

game_id, release_year : string { user inputted filter }

user_game_id : array of string

i : integer { iteration variable }

game_data_output : array of array of string

filtered_game_data_output_by_game_id : array of array of string

filtered_game_data_output_by_release_year : array of array of string

filtered_game_data : array of array of string

filtered_game_data_max_char_length : array of integer

p : integer

q : array of array of string

F11

Function **filter_str** (data : array of array of string, index : integer, criteria : string) -> array of array of string

KAMUS LOKAL

temp : array of array of string

i : integer

Function **filter_int** (data : array of array of string, index : integer, criteria : string) -> array of array of string

KAMUS LOKAL

temp : array of array of string

i : integer

Procedure **search_game_at_store** (**input** game_data : array of array of string)

KAMUS LOKAL

id, name, category, release_year, price : string

header : array of string

filtered, temp : array of array of string

i, j, k, character_amount : integer

F12

<u>Function</u> function_topup (input username **:** string**, input** balance **:** integer**, input** user_data **:** array of array**) ->** array of array

KAMUS LOKAL

user_valid : boolean

line_index, current_balance : integer

<u>Function</u> topup (input data **:** array of array**) ->** array of array

KAMUS LOKAL

username : string

balance : integer

F13

<u>Procedure</u> history (input hist_data **:** array of array**)**

KAMUS LOKAL

data_history : array of array

i, j, k, l, character_amount : integer

F14

<u>Procedure</u> help (input user **:** string**, input** save_folder **:** string**)**

KAMUS LOKAL

is_user_admin : boolean

F15

<u>Procedure</u> Load

KAMUS LOKAL

parser : function

save_folder : string

all_folder : array

F16

<u>Procedure</u> saver (input folder **:** string**, input** data **:** array**)**

KAMUS LOKAL

path : string

exist : boolean

F17

<u>Procedure</u>

KAMUS LOKAL

x : string

B01

<u>Function</u> encrypt (password : string) -> string

KAMUS LOKAL

a, b : integer

ciphered : string

<u>Function</u> decrypt (ciphered : string) -> string

KAMUS LOKAL

a, b : integer

i : integer

password : string

B02

<u>Function</u> magicconch()

KAMUS LOKAL

x, a, c, m, state : integer

B03

<u>Procedure</u> ask_location (**input/output** matrix : array of array of strings, **input** pawn : string)

KAMUS LOKAL

valid : boolean

x,y : integer

<u>Function</u> win_checker (matrix : array of array of strings, pawn : string) -> string

KAMUS LOKAL

win : string

<u>Procedure</u> status(**input/output** matrix : array of array of strings)

KAMUS LOKAL

i,j : integer

<u>Procedure</u> tictactoe()

KAMUS LOKAL

matrix : array of array of characters

turn : integer

pawn, string : string

FUNGSI TAMBAHAN

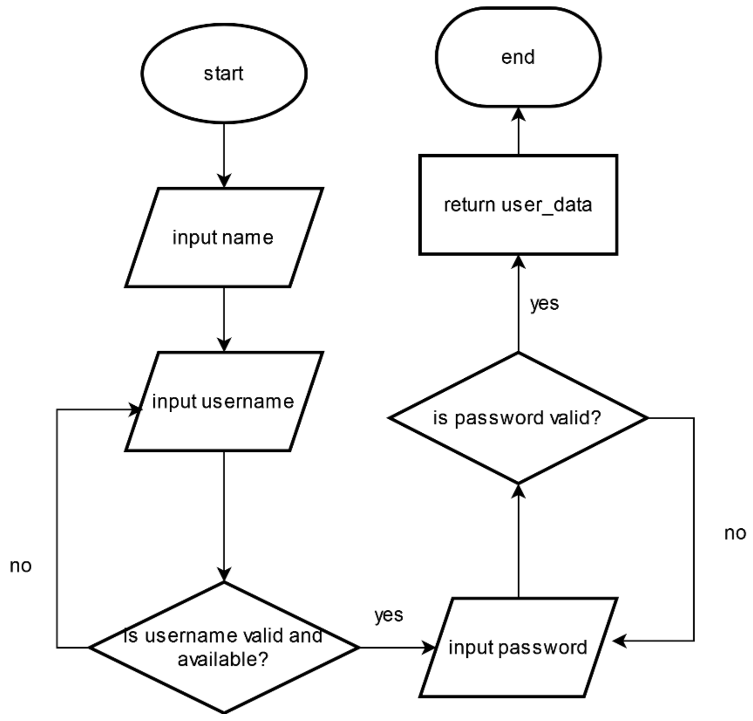
<u>Procedure</u>

KAMUS LOKAL

Lorem Ipsum : Dolor Sit Amet

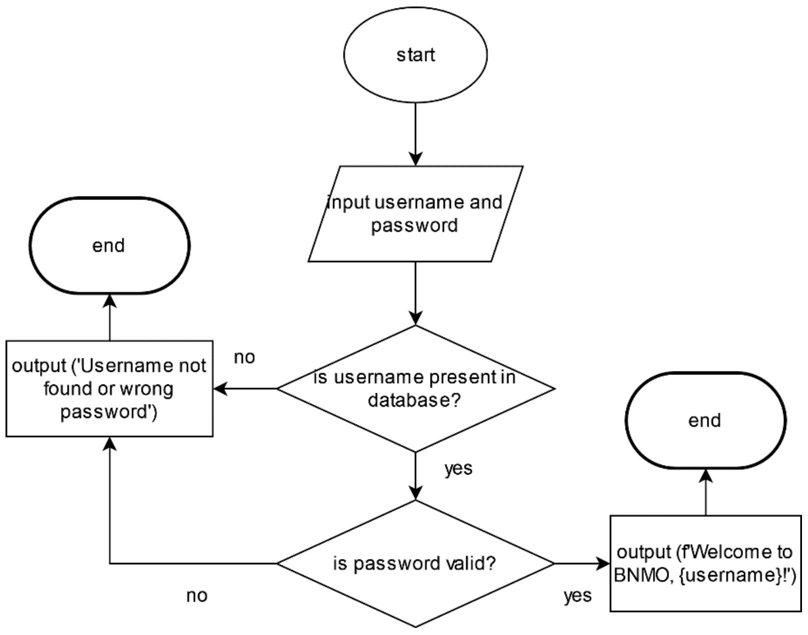
VI. DESAIN DEKOMPOSISI ALGORITMIK DAN FUNGSIONAL PROGRAM

F02 - Register



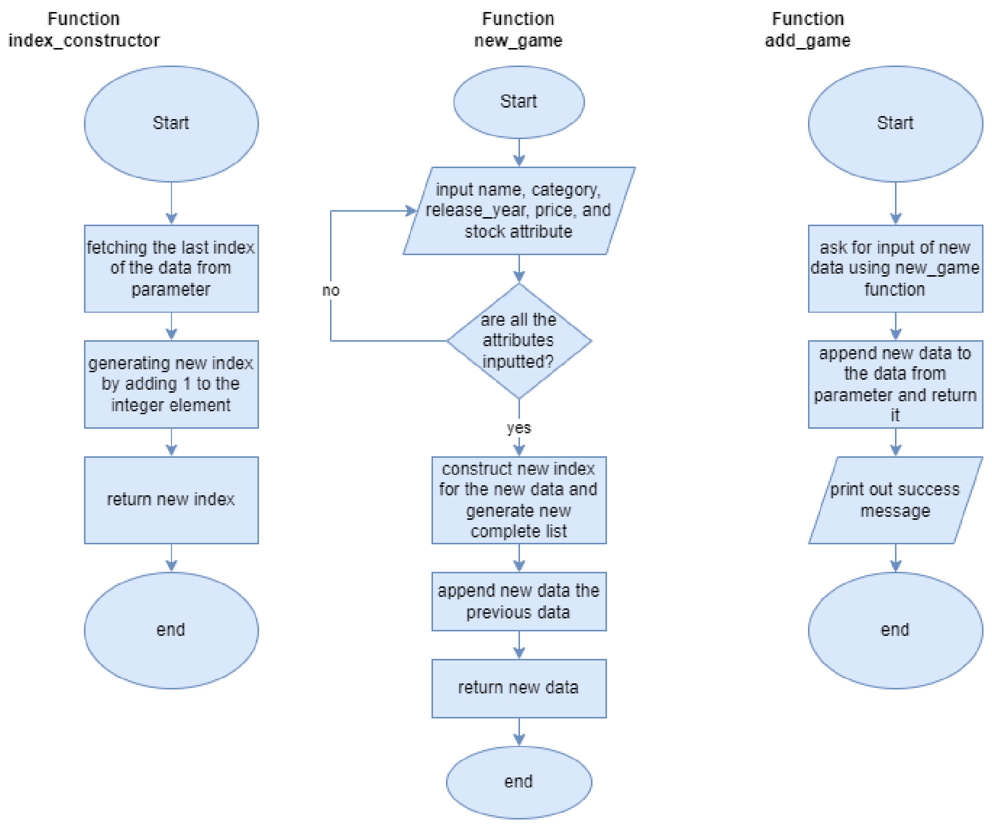
Gambar 6.2 Flowchart untuk modul 2

F03 – Login


start
input username and password
is username present in database?
no
output ('Username not found or wrong password')
end
yes
is password valid?
no
yes
output (f'Welcome to BNMO, {username}!')
end

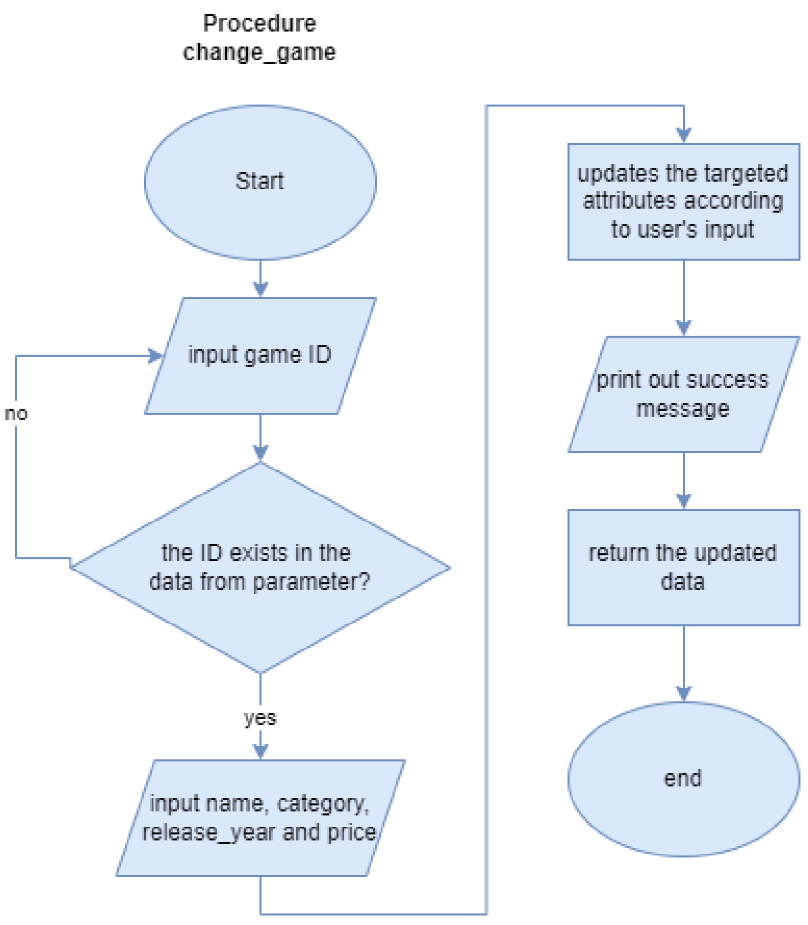
Gambar 6.3 Flowchart modul 3

F04 – Menambah Game ke Toko Game


Function index_constructor
Start
fetching the last index of the data from parameter
generating new index by adding 1 to the integer element
return new index
end
Function new_game
Start
input name, category, release_year, price, and stock attribute
are all the attributes inputted?
no
yes
construct new index for the new data and generate new complete list
append new data the previous data
return new data
end
Function add_game
Start
ask for input of new data using new_game function
append new data to the data from parameter and return it
print out success message
end

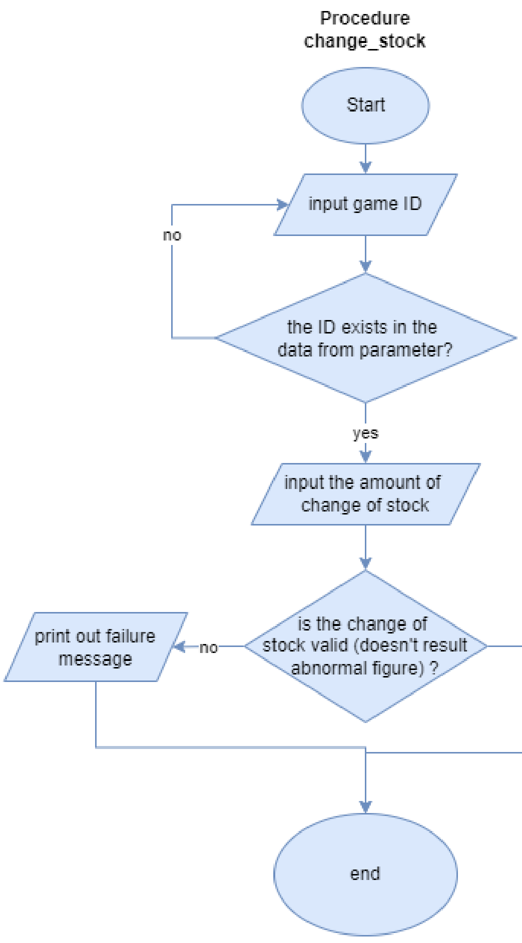
Gambar 6.4 Flowchart untuk modul 4

F05 – Mengubah Game pada Toko Game


Procedure change_game
Start
input game ID
no
the ID exists in the data from parameter?
yes
input name, category, release_year and price
updates the targeted attributes according to user's input
print out success message
return the updated data
end

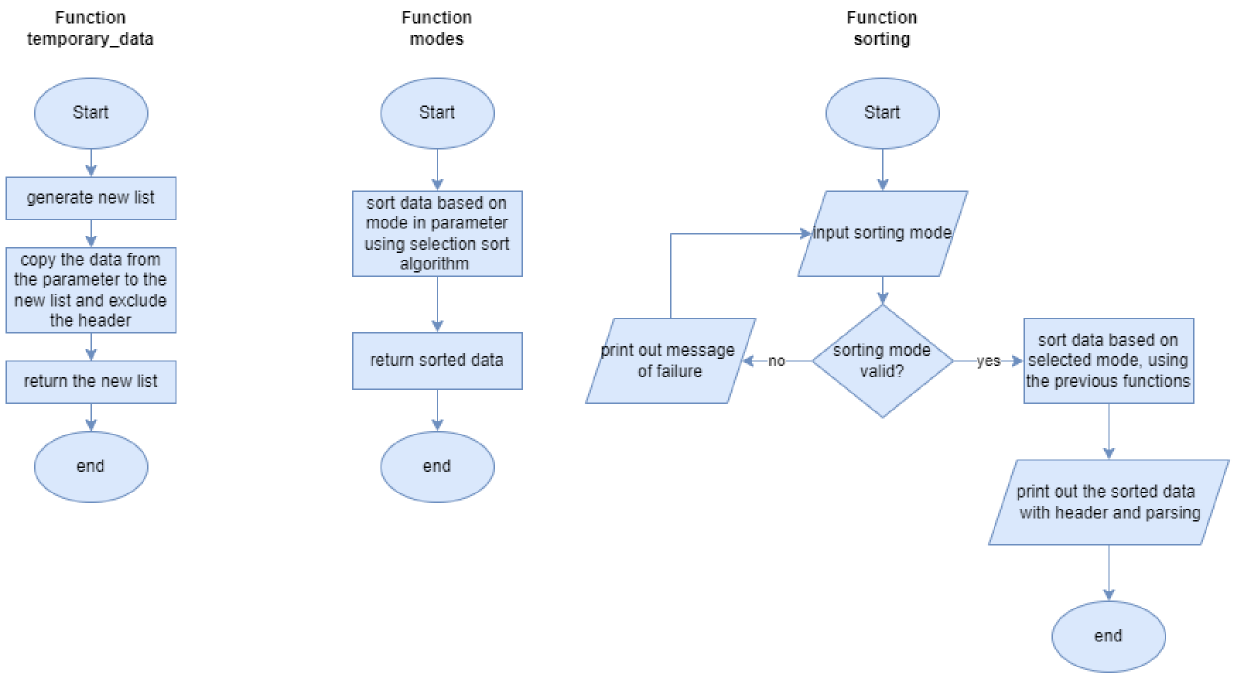
Gambar 6.5 Flowchart untuk modul 5

F06 – Mengubah Stok Game di Toko


Procedure change_stock
Start
input game ID
no
the ID exists in the data from parameter?
yes
input the amount of change of stock
is the change of stock valid (doesn't result abnormal figure) ?
no
print out failure message
yes
updates the targeted stock according to user's input
print out success message
return the updated data
end

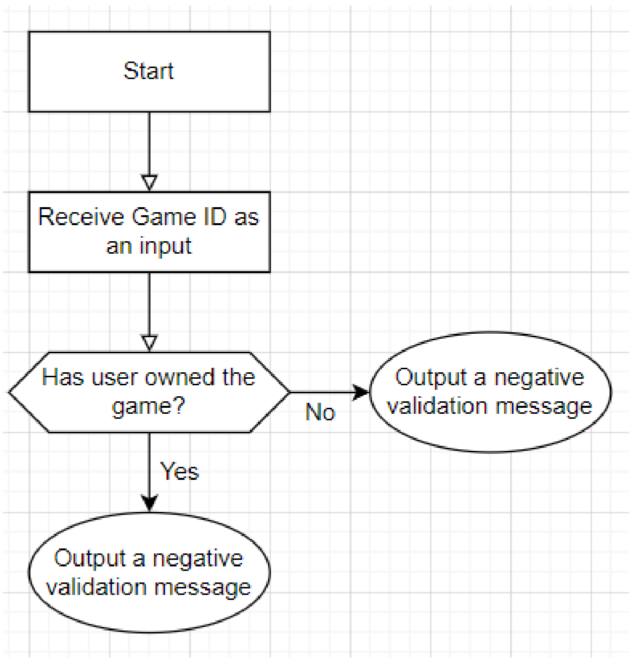
Gambar 6.6 Flowchart untuk modul 6

F07 – Listing Game di Toko Berdasarkan ID, Tahun Rilis dan Harga


Function temporary_data
Start
generate new list
copy the data from the parameter to the new list and exclude the header
return the new list
end
Function modes
Start
sort data based on mode in parameter using selection sort algorithm
return sorted data
end
Function sorting
Start
input sorting mode
sorting mode valid?
no
print out message of failure
yes
sort data based on selected mode, using the previous functions
print out the sorted data with header and parsing
end

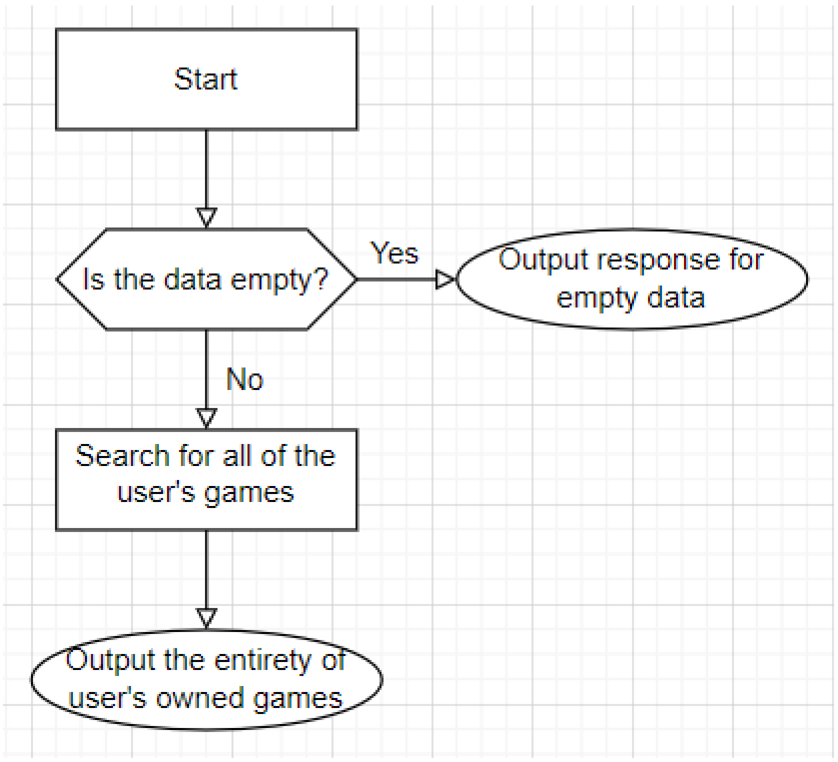
Gambar 6.7 Flowchart untuk modul 7

F08 – Buy Game


Start
Receive Game ID as an input
Has user owned the game?
No
Output a negative validation message
Yes
Output a negative validation message

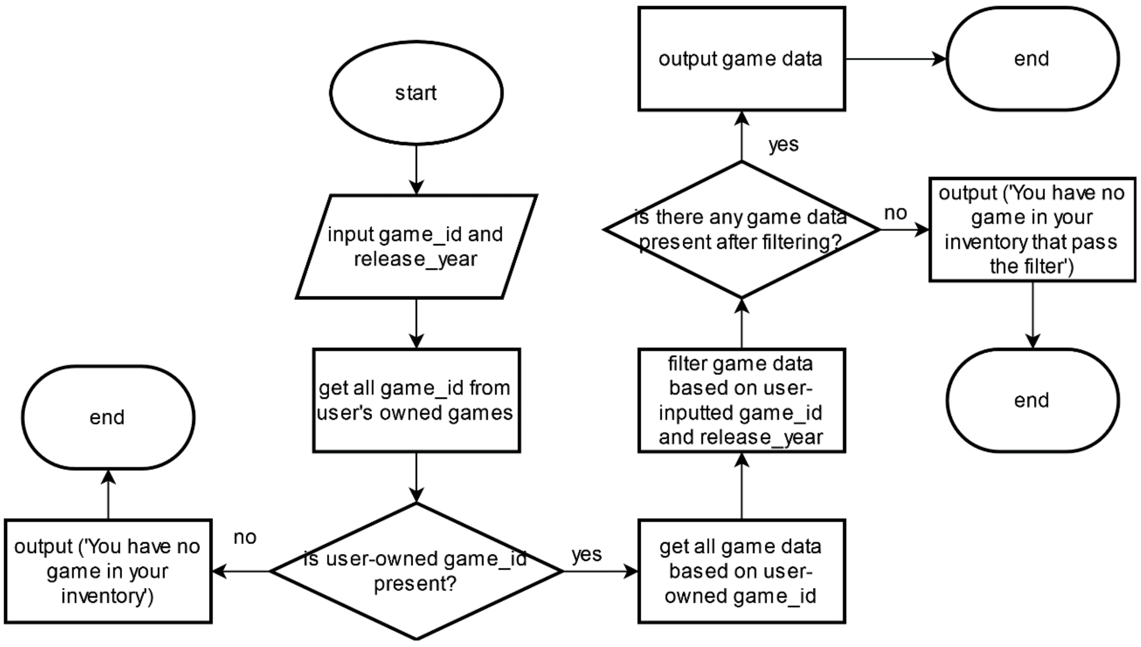
Gambar 6.8 Flowchart untuk modul 8

F09 – List Game


Start
Is the data empty?
Yes
Output response for empty data
No
Search for all of the user's games
Output the entirety of user's owned games

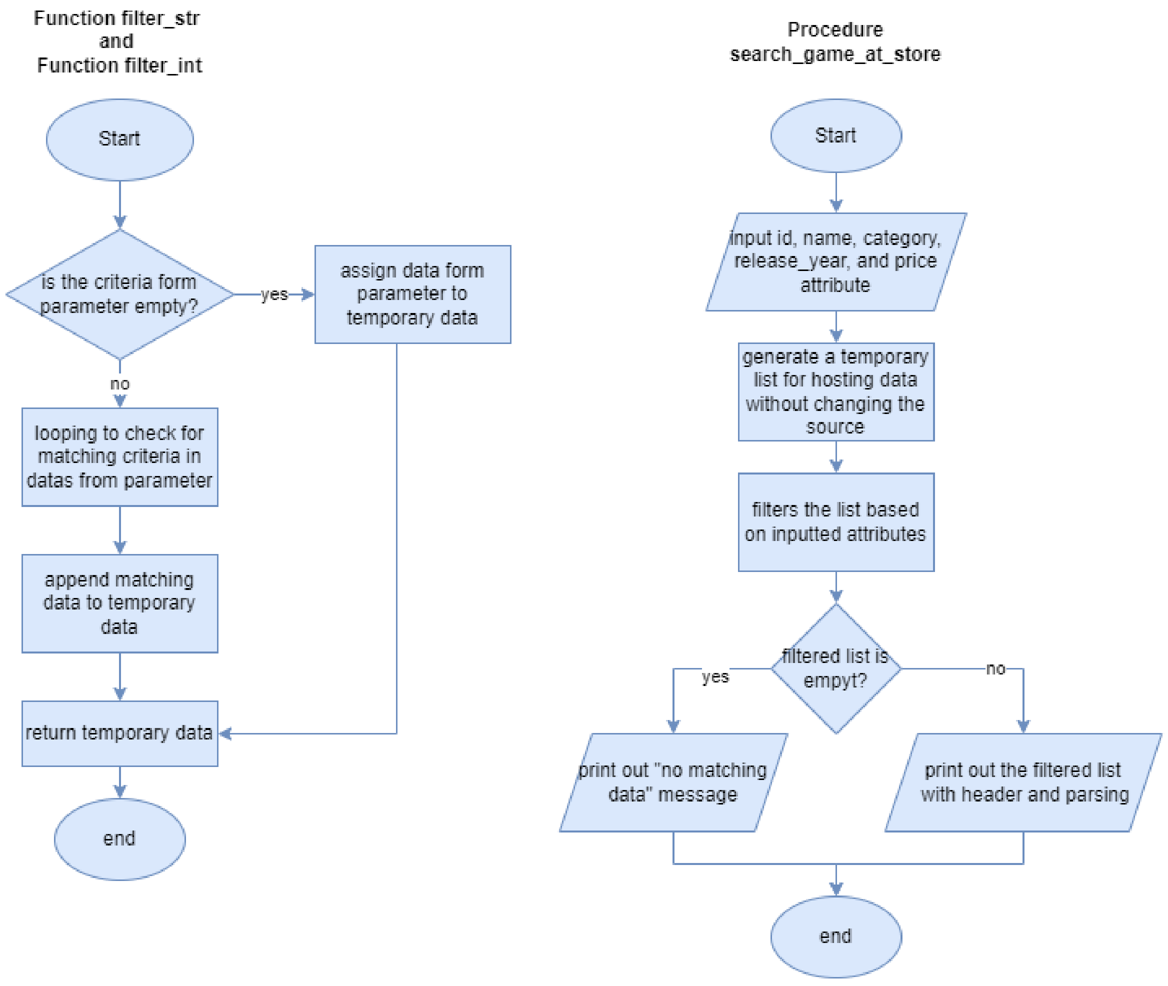
Gambar 6.9 Flowchart untuk modul 9

F10 – Mencari Game yang dimiliki dari ID dan tahun rilis


start
input game_id and release_year
get all game_id from user's owned games
is user-owned game_id present?
no
output ('You have no game in your inventory')
end
yes
get all game data based on user-owned game_id
filter game data based on user-inputted game_id and release_year
is there any game data present after filtering?
yes
output game data
end
no
output ('You have no game in your inventory that pass the filter')
end

Gambar 6.10 Flowchart untuk modul 10

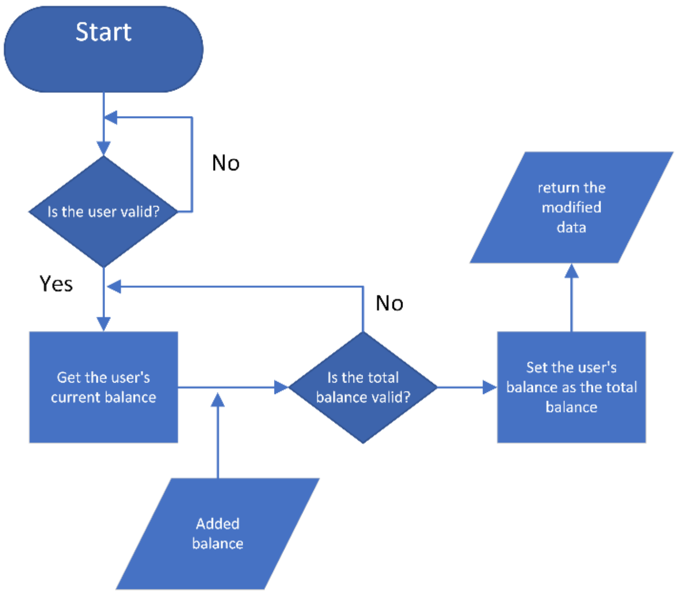
F11 – Mencari Game di Toko dari ID, Nama Game, Harga, Kategori dan Tahun Rilis


Function filter_str and Function filter_int
Start
is the criteria form parameter empty?
yes
assign data form parameter to temporary data
no
looping to check for matching criteria in datas from parameter
append matching data to temporary data
return temporary data
end
Procedure search_game_at_store
Start
input id, name, category, release_year, and price attribute
generate a temporary list for hosting data without changing the source
filters the list based on inputted attributes
filtered list is empyt?
yes
print out "no matching data" message
no
print out the filtered list with header and parsing
end

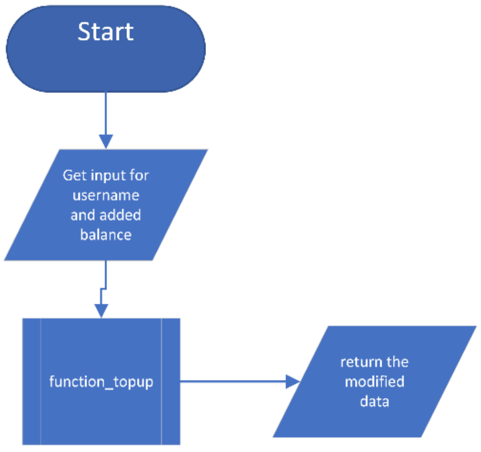
Gambar 6.11 Flowchart untuk modul 11

F12 – Topup

Function function_topup

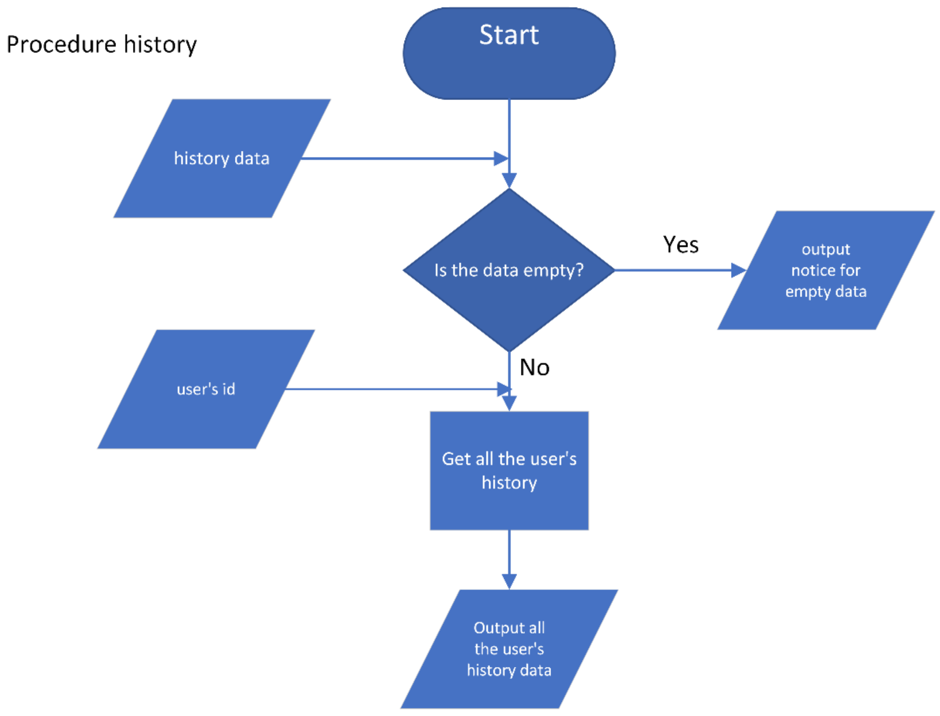

Start
Is the user valid?
No
Yes
Get the user's current balance
Added balance
Is the total balance valid?
No
Set the user's balance as the total balance
return the modified data

Function topup



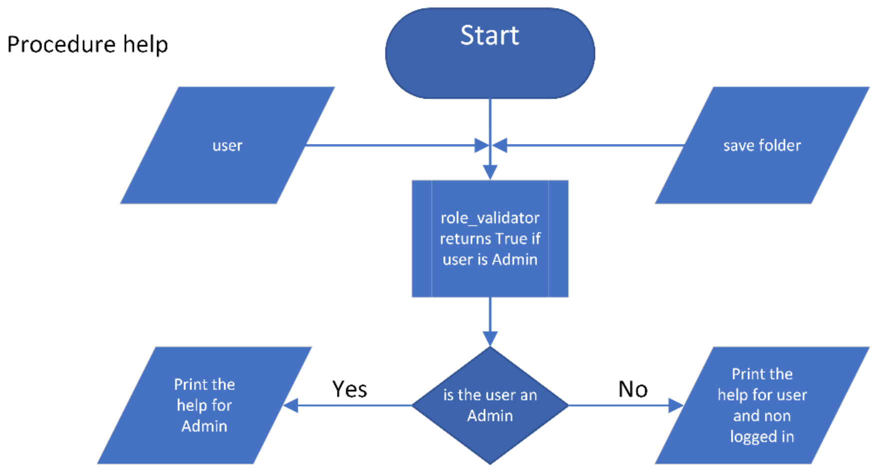
Gambar 6.12 Flowchart untuk modul 12

F13 – History


Procedure history
Start
history data
Is the data empty?
Yes
output notice for empty data
No
user's id
Get all the user's history
Output all the user's history data

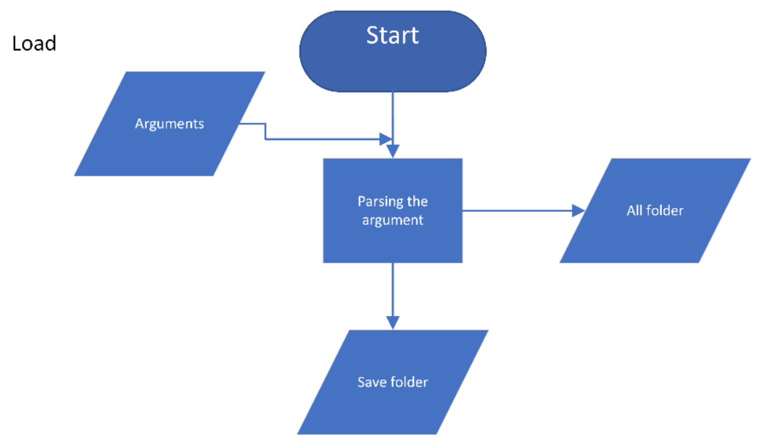
Gambar 6.13 Flowchart untuk modul 13

F14 – Help


Procedure help
Start
user
save folder
role_validator returns True if user is Admin
is the user an Admin
Yes
Print the help for Admin
No
Print the help for user and non logged in

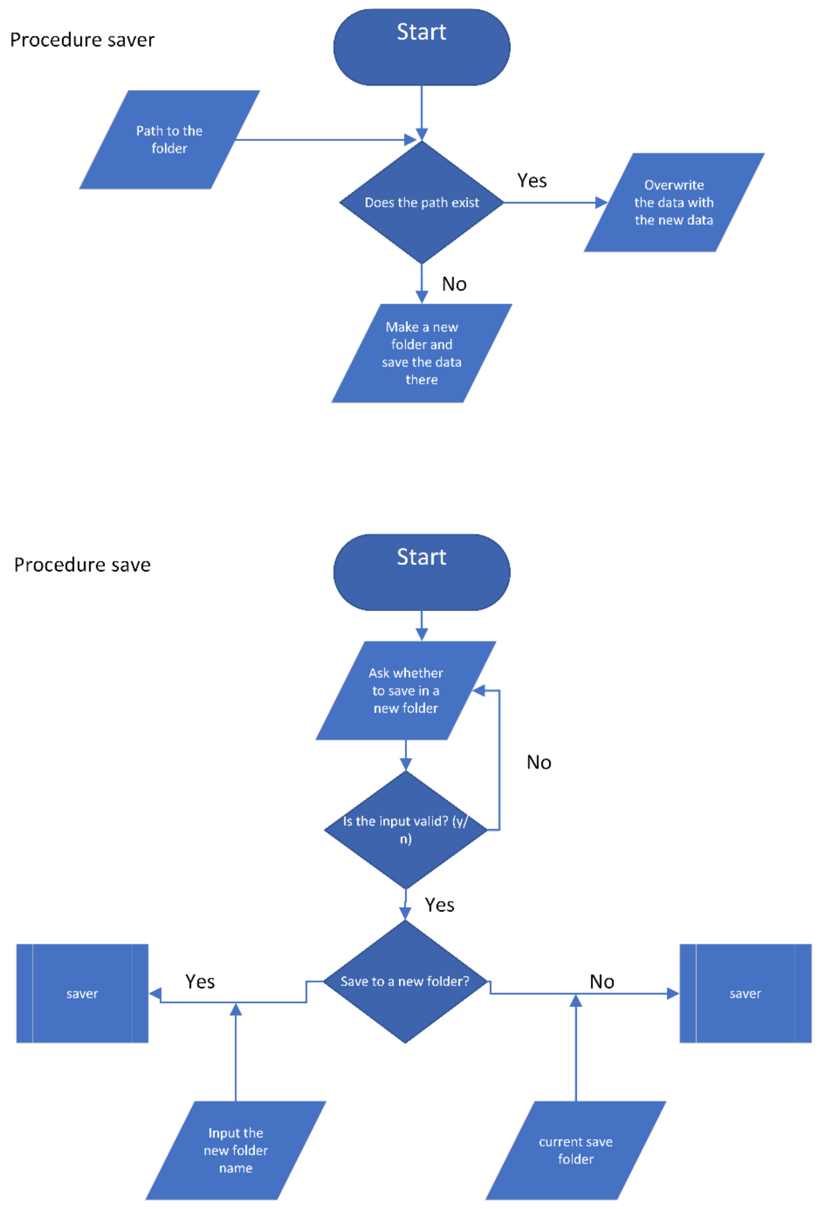
Gambar 6.14 Flowchart untuk modul 14

F15 – Load



Gambar 6.15 Flowchart untuk modul 15

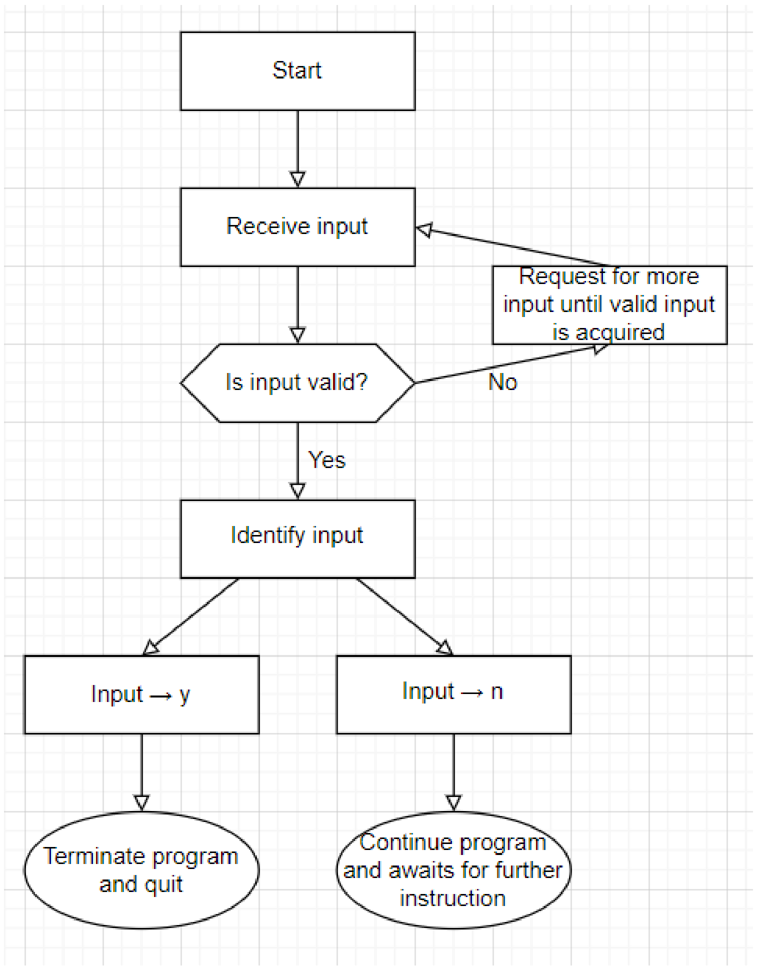
F16 – Save



Procedure saver
Start
Path to the folder
Does the path exist
Yes
Overwrite the data with the new data
No
Make a new folder and save the data there
Procedure save
Start
Ask whether to save in a new folder
No
Is the input valid? (y/n)
Yes
Save to a new folder?
Yes
saver
No
saver
Input the new folder name
current save folder

Gambar 6.16 Flowchart untuk modul 16

F17 – Exit


Start
Receive input
Request for more input until valid input is acquired
Is input valid?
No
Yes
Identify input
Input → y
Input → n
Terminate program and quit
Continue program and awaits for further instruction

Gambar 6.17 Flowchart untuk modul 17

VII. SPESIFIKASI UNTUK SETIAP MODUL YANG DIBUAT MAIN PROGRAM

F02 – Register

```
DICTIONARY
    function length (input) -> integer
    { Function to calculate the length of an object. }

    function append (list_ : array, input) -> array
    { Function to append an input to a list. }

    function encrypt (password : string) -> string
    { Encrypts user password using the Affine cipher. }


function register (user_data : array of array of string) -> array of
array of string
    { Function to add a list of id, username, name, ciphered password,
role, and balance of user
    to the loaded user.csv data on the main program (GUI.py). }

LOCAL DICTIONARY
    name, username, ciphered_password, role, char, password, char_pass :
string
    id, balance : integer

ALGORITHM
    input (name)
    input (username)

    { Loops until the username is valid }
    while (True) do

        try
            { Username valid characters (-, _, 0-9, A-Z, a-z)
validation }
            char traversal username
                if (not (ord(char) = 45 or ord(char) = 95 or 48 <=
ord(char) <= 57 or
                        65 <= ord(char) <= 90 or 97 <= ord(char) <= 122))
then  { -, _, 0-9, A-Z, a-z respectively }
                    output ('Username is not valid. Please only use
letters, numbers, underscore (_), and dash (-).')
                    raise ValueError

            { Checks if the username is already present }
            { Loop for every entry in user.csv excluding the first line }
            i traversal [1..length(user_data) - 1]
                if (username =s user_data[i][1]) then
                    output (f'Username "{username}" already exists,
please select a different username.\n')
                    raise ValueError

        except ValueError
            input (username)

        else
            output ('Username is available!')
            break

    input (password)

    { Loops until the password is valid }
    while (True) do

        try
            { Password validation to not break user.csv }
            char_pass traversal password
                if (char_pass = ';') then
                    output ('Password must not contain semicolon (;)')
                    raise ValueError

        except ValueError
            input (password)

        else
            output ('Password is valid!')
            break

    id <- length(user_data)
    ciphered_password <- encrypt(password)
    role <- 'User'  { Register can only add a user, not admin }
    balance <- 0  { Initial balance is always 0 }

    new_user <- [id, username, name, ciphered_password, role, balance]

    user_data <- append(user_data, new_user)

    -> user_data
```

F03 – Login

```
DICTIONARY
    function decrypt (ciphered : string) -> string
    { Decrypts ciphered user password using the Affine cipher. }

    function length (input) -> integer
    { Function to calculate the length of an object. }

    user_line_index : integer  { global variable on what index is the
current user's username stored in user.csv }


function login (user_data : array of array of string) -> boolean
    { Returns True if the username and password is correct and it is on
the database.
    Returns False otherwise. }

LOCAL DICTIONARY
    user_valid : boolean  { username present in database or not }
    username, password : string

ALGORITHM
    input (username)
    input (password)

    user_valid <- False
    { Checks if the username is present in database }
    { Loop for every line in file user.csv (index 3 on folder save)
(ignore the first line) }
    i traversal [1..length(user_data) - 1]
        if (username = user_data[i][1]) then
            user_valid <- True
            global user_line_index
            user_line_index <- i

    if (user_valid) then
        if (decrypt(user_data[user_line_index][3]) = password) then
            output (f'Welcome to BNMO, {username}!')
            -> True
        else { not (decrypt(user_data[user_line_index][3]) = password) }
            output ('Username not found or wrong password')
            -> False
    else { not (user_valid) }
        output ('Username not found or wrong password')
        -> False
```

F04 – Menambah Game ke Toko Game

```
DICTIONARY
    function length (input) -> integer
    { Function to calculate the length of an object. }

    function append (list_ : array, input) -> array
    { Function to append an input to a list. }


{not to be imported}
function index_constructor (game_data : array of array of string) ->
array of array of string
    {I.S. game_data array of  is ordered based on game ID}
    {F.S. new index is generated and returned}

LOCAL DICTIONARY
    prevoius_number, new_number : integer
    new_index : string

ALGORITHM
    {Fetching the integers of the last Game ID}
    previous_number <- int(game_data[length(game_data)-1][0][4]) * 100 +
int(game_data[length(game_data)-1][0][5]) * 10 +
int(game_data[length(game_data)-1][0][6])

    new_number <- previous_number + 1
    if (new_number < 10) then
        new_index <- "GAME00" + str(new_number)
    else (if new_number < 100) then
        new_index <- "GAME0" + str(new_number)
    else
        new_index <- "GAME" + str(new_number)

    -> new_index

{not to be imported}
function new_game (game_data : array of array of string) -> array of
array of string
```

```
LOCAL DICTIONARY
    complete : boolean
    name, category, release_year : string
    price, stock : integer
    new_index : string
    new_data : array of string

ALGORITHM
    {I.S. game_data is defined and have all the atrributes}
    {F.S. new game data is collected and validated, index is generated
automatically, new data is returned}

    complete <- False
    while (complete = False) do                        # loop for input
completeness validation
        input(name, category, release_year, price, stock)

        {Input validation}
        if (length(name) = 0) or (length(category) = 0) or
(length(release_year) = 0) or (length(price) = 0) or (length(stock) = 0)
then
            output("Please insert all of the game information to be saved
by BNMO.")
        else
            complete <- True

    new_index <- index_constructor(game_data)      # generating id for
the new game by fetching the latest id from the database + 1
    new_data <- [new_index , name, category, release_year, int(price),
int(stock)]

    -> new_data


function add_game (game_data : array of array of string) -> array of
array of string
    {I.S game_data is defined and have all the atrributes}
    {F.S game_data array is returned with validated new game data}

LOCAL DICTIONARY
    new_data : array of string
    game_data : array of array of string

ALGORITHM
    new_data <- new_game(game_data)
    game_data <- append(game_data, new_data)
    output("Congratulations! Adding game succeded", new_data[1] + "." )

    -> game_data
```

F05 – Mengubah Game pada Toko Game

```
DICTIONARY
    function length (input) -> integer
    { Function to calculate the length of an object. }

procedure change_game (input/output game_data : array of array of string)
```

```
    {I.S. game_data array is ordered based on game ID}
    {F.S. new index is generated and returned}

LOCAL DICTIONARY
    id : string
    found : boolean
    i, line_index : integer

ALGORITHM
    input(id)

    {finding matching game ID in game data matrix}
    found <- False
    i <- 1
    while (found == False) and (i <= length(game_data)) do     {Loop for
every line in file game.csv (index 3 on folder save) (ignore the first
line)}
        if  (id = game_data[i][0]) then
            line_index <- i
            found <- True
        else
            i <- i + 1

    if (found = False) then
        output("There's no game with that ID!")

    else       # ID is found
        input(name, category, release_year, price)
        if name != "" then
            game_data[line_index][1] <- name

        if category != "" then
            game_data[line_index][2] <- category

        if release_year != "" then
            game_data[line_index][3] <- release_year


        if price != "" then
            game_data[line_index][4] <- price


        -> game_data
```

F06 – Mengubah Stok Game di Toko

```
DICTIONARY
    function length (input) -> integer
    { Function to calculate the length of an object. }


function change_stock(game_data : array of array of string ) -> array of
array of string
    {Function to change the amount of stock of an existing game, complete
with input validation}


LOCAL DICTIONARY
```

```
    id : string
    found : boolean
    i, line_index, added_stock : integer

ALGORITHM
    input(id)

    {finding matching game ID in game data matrix}
    found <- False
    i <- 1
    while (found = False and i < length(game_data)) do     {Loop for
every line in file game.csv (index 3 on folder save) (ignore the first
line)}
        if  (id = game_data[i][0]) then
            line_index <- i
            found <- True
        else
            i <- i + 1

    if (found == False) then
        output("There's no game with that ID!")

    else     {ID is found}
        input(added_stock)

        if int(game_data[line_index][5]) + added_stock < 0 then
            output(game_data[line_index][1], "stock substraction failed
due to not enough stock. Current stock:", game_data[line_index][5],
"(<" , str(abs(added_stock)) , ")" )
        else
            game_data[line_index][5] <- int(game_data[line_index][5]) +
added_stock
            if (added_stock = 0)
                output("No changes were made to the amount of",
game_data[line_index][1] + "'s stock. Current stock:",
game_data[line_index][5])
            else (if added_stock > 0) then
                output(game_data[line_index][1], "stock addition
succeeded. Current stock:", game_data[line_index][5])
            else       {added_stock < 0}
                output(game_data[line_index][1], "stock substraction
succeeded. Current stock:", game_data[line_index][5])
    -> game_data
```

F07 – Listing Game di Toko Berdasarkan ID, Tahun Rilis dan Harga

```
DICTIONARY
    function length (input) -> integer
    { Function to calculate the length of an object. }


{not to be imported}
function temporary_data(game_data : array of array of string) -> array of
array of string
    {Function to generate temporary list for hosting data without
changing the source}

LOCAL DICTIONARY
```

```
    data : array of array of string
    i : integer

ALGORITHM
    data <- ["*" i traversal [0..(length(game_data)-1)]]
    i traversal [1..length(game_data))]                {traversing from 1
to skip data header}
        data[i-1] <- game_data[i]
    -> data


{not to be imported}
function modes(game_data : array of array of string, mode : string) ->
array of array of string
    {Function to sort data based on modes}

LOCAL DICTIONARY
    sorted, temp : array of array of integer
    i, j, index_min : integer

ALGORITHM
    sorted <- temporary_data(game_data)

    i traversal [0..(length(sorted)-1)]                    {sorting data
using selection sort algortihm}
        index_min <- i

        j traversal [(i+1)..(length(sorted))]
            if (mode = "year+") then
                if (int(sorted[index_min][3]) > int(sorted[j][3])) then
                    index_min <- j
            else if (mode = "year-") then
                if (int(sorted[index_min][3]) < int(sorted[j][3])) then
                    index_min <- j
            else if (mode ="price+") then
                if int(sorted[index_min][4]) > int(sorted[j][4]) :
                    index_min <- j
            else if (mode = "price-") then
                if int(sorted[index_min][4]) < int(sorted[j][4]) :
                    index_min <- j

        temp <- sorted[index_min]
        sorted[index_min] <- sorted[i]
        sorted[i] <- temp

    -> sorted


procedure sorting (input game_data : array of array of string)
    {Procedure to print out parsed and sorted game data}

LOCAL DICTIONARY
    valid : boolean
    mode : string
    sorted, temp : array of array of string
    header : array of string
    i, j, k, character_amount : integer

ALGORITHM
    valid <- False
    while (valid = False) do
        input(mode)

        if (mode == "year+") or (mode == "year-") or (mode == "price+")
or (mode == "price-") then
            sorted = modes(game_data, mode)
            valid = True
        else
            output("Invalid sorting mode. Please try again!")

    {adding header to sorted data}
    header <- ["ID", "NAME", "CATEGORY", "RELEASE YEAR", "PRICE",
"STOCK"]


    temp <- ["*" i traversal[0..(length(sorted)+1)]]
    temp[0] <- header
    i traversal [0..length(sorted)]
        temp[i + 1] <- sorted[i]

    sorted <- temp

    {Generate parsing for sorted data}
    i traversal [0..length(sorted)]
        if (i = 0) then                         {skip numbering for the
header}
            output("  ", end= " ")
        else                                {numbering for the list}
            output(str(i) + ".", end=" ")


        j traversal [0..6]
            output(sorted[i][j], end="")
            character_amount <- 0
            k traversal [0..length(sorted)]
                if (length(str(sorted[k][j])) > character_amount) then
                    character_amount <- length(str(sorted[k][j]))

            output((" " * (character_amount -
length(str(sorted[i][j])))), "| ", end="")
        output("")
```

F08 – Membeli Game

```
DICTIONARY

    function is (list : list) -> (list : list) -> string -> string



LOCAL DICTIONARY

game_data: list of lists

my_game: list of lists

ID : string
```

```
ALGORITHM

ID <- String

if ID in game_data then

      if ID in my_game then

            output("You have owned that game!")

else

            output("Game is succesfully bought!")

else

output("Game doesn't exist")
```

F09 – Melihat Game yang dimiliki

```
DICTIONARY

    function is (list : list) -> (list : list)


LOCAL DICTIONARY

    Game_data : list of lists


ALGORITHM

      If game_data length = 0

            output("You haven't bought any game")

      else:

            output(game_data)
```

F10 – Mencari Game yang dimiliki dari ID dan tahun rilis

```
DICTIONARY
    function length (input) -> integer
    { Function to calculate the length of an object. }

    function append (list_ : array, input) -> array
    { Function to append an input to a list. }

    function enum (list_ : array, start=0)
    { Works the same way as the built-in function enumerate(). Returns an
enumerate object. }
```

```
    user_line_index : integer  { global variable on what index is the
current user's username stored in user.csv }


{ not to be imported }
[ data is filtered list, index is the location of the targetted
attribute, criteria is the previously asked input by search_game_at_store
procedure }
function filter_str (data : array of array of string, index : integer,
criteria : string) -> array of array of string
    { Function to create a list filtered by a string attribute }

LOCAL DICTIONARY
    temp : array of array of string
    i : integer  { iteration variable }

ALGORITHM
    if (criteria = "") then
        temp <- data
    else { not (criteria = "") }
        temp <- [] { temp is for hosting matching datas }

        { traversing to find matching attribute and appending the list to
temp }
        i traversal [0..length(data) - 1]
            if (criteria = data[i][index]) then
                temp <- append(temp, data[i])

    -> temp


{ not to be imported }
function get_max_char_length (filtered_game_data : array of array of
string) -> array of integer
    { Function to get the maximum character length for each column in the
filtered game data
    (i.e. filtered game.csv according to kepemilikan.csv and user-
inputted game_id and release_year) }

LOCAL DICTIONARY
    filtered_game_data_char_length : array
    char_length_list : array
    L : array of string
    m, n : integer
    max_length_of_column : integer
    o : array
    filtered_game_data_max_char_length : array

ALGORITHM
    filtered_game_data_char_length <- []

    L traversal filtered_game_data
        char_length_list <- []
        m traversal [0..4]  { Don't index the stock of game }

            { char_length_list is a list of character length for each
game entry }
            char_length_list <- append(char_length_list, length(L[m]))
        { filtered_game_data_char_length is the complete list of list of
character length for the filtered game data }
        filtered_game_data_char_length <-
append(filtered_game_data_char_length, char_length_list)


    filtered_game_data_max_char_length <- []

    n traversal [0..4]

        max_length_of_column <- 0
        o traversal filtered_game_data_char_length
            if (o[n] > max_length_of_colum) then
                max_length_of_column <- o[n]

        filtered_game_data_max_char_length <-
append(filtered_game_data_max_char_length, max_length_of_column)

    -> filtered_game_data_max_char_length


{ ownership_data is the kepemilikan.csv of a save folder, user_data is
the user.csv, while game_data is the game.csv }
procedure search_my_game (ownership_data, user_data, game_data : array of
array of string)
    { Procedure that prints user-owned games based on its ID and release
year. }

LOCAL DICTIONARY
    game_id, release_year : string  { user inputted filter }
    user_game_id : array of string
    i : integer  { iteration variable }
    game_data_output : array of array of string
    filtered_game_data_output_by_game_id : array of array of string
    filtered_game_data_output_by_release_year : array of array of string
    filtered_game_data : array of array of string
    filtered_game_data_max_char_length : array of integer
    p : integer
    q : array of array of string

ALGORITHM
    input (game_id.upper())
    input(release_year)

    user_game_id <- []  { All Game ID of the currently logged in user }

    { Loop for every entry in kepemilikan.csv excluding the first line }
    i traversal [1..length(ownership_data) - 1]

        { if user id in kepemilikan.csv == user id of currently logged in
user, then append the game id of the game with matchin user id }
        if (ownership_data[i][1] = user_data[user_line_index][0]) then
            user_game_id <- append(user_game_id, ownership_data[i][0])

    if (length(user_game_id) = 0) then
        output ('You have no game in your inventory')

    else { not (length(user_game_id) = 0) }

        game_data_output <- []
        j traversal user_game_id
            { Loop for every entry in game.csv excluding the first line }
            k traversal [1..length(game_data) - 1]

                if (j = game_data[k][0]) then  { if user game id matches
game id in game.csv }
                    game_data_output <- append(game_data_output,
game_data[k])

        { Filter based on game_id and release_year }
        filtered_game_data_output_by_game_id <-
filter_str(game_data_output, 0, game_id)
        filtered_game_data_output_by_release_year <-
filter_str(filtered_game_data_output_by_game_id, 3, release_year)
        filtered_game_data <- filtered_game_data_output_by_release_year

        if (length(filtered_game_data) = 0) then
            output ('You have no game in your inventory that pass the
filter')

        else { not (length(filtered_game_data) = 0) }

            filtered_game_data_max_char_length <-
get_max_char_length(filtered_game_data)

            output ('\nGames in your inventory that meet the filter:')

            p, q traversal enum(filtered_game_data, start=1)
                output (f'{p}.
{q[0].ljust(filtered_game_data_max_char_length[0])} |
{q[1].ljust(filtered_game_data_max_char_length[1])} |
{q[4].ljust(filtered_game_data_max_char_length[4])} |
{q[2].ljust(filtered_game_data_max_char_length[2])} |
{q[3].ljust(filtered_game_data_max_char_length[3])}')
```

F11 – Mencari Game di Toko dari ID, Nama Game, Harga, Kategori dan Tahun Rilis

```
DICTIONARY
    function length (input) -> integer
    { Function to calculate the length of an object. }

    function append (list_ : array, input) -> array
    { Function to append an input to a list. }



{not to be imported}
{data is filtered list, index is the location of the targetted attribute,
criteria is the previously asked input by search_game_at_store procedure}
function filter_str (data : array of array of string, index : integer,
criteria : string) -> array of array of string
    {Function to create a list filtered by a string attribute}

LOCAL DICTIONARY
    criteria : string
    temp : array of array of string
    i : integer
```

```
ALGORITHM
    if (criteria = "") then
        temp <- data
    else
        temp <- [] # temp is for hosting matching datas

        {traversing to find matching attribute and appending the list to
temp}
        i traversal[0..(length(data))]
            if criteria == data[i][index] then
                temp = append(temp, data[i])
    -> temp


{not to be imported}
{data is filtered list, index is the location of the targetted attribute,
criteria is the previously asked input by search_game_at_store procedure}
function filter_int (data : array of array of string, index : integer,
criteria : string) -> array of array of string
    {Function to create a list filtered by an integer attribute}
LOCAL DICTIONARY
    temp : array of array of string
    i : integer

ALGORITHM
    if (criteria = "") then
        temp <- data
    else
        temp <- [] { temp is for hosting matching datas }

        {traversing to find matching attribute and appending the list to
temp}
        i traversal[0..(length(data))]
            if [int(criteria) = data[i][index]]
                temp <- append(temp, data[i])

    -> temp


{game_data is game.csv}
procedure search_game_at_store (input game_data : array of array of
string)
    { Procedure to print out a filtered list based on criteria from user
input }

LOCAL DICTIONARY
    id, name, category, release_year, price : string
    header : array of string
    filtered, temp : array of array of string
    i, j, k, character_amount : integer

ALGORITHM
    input(id, name, category, release_year, price)

    {generate a temporary list for hosting data without changing the
source}
    filtered <- ["*" i traversal [0..(length(game_data)-1)]]
    i traversal [1..length(game_data)]                [traversing from 1
to skip data heading]
```

```
        filtered[i-1] <- game_data[i]

    {each line creates a new filtered list from the previous filtered
list}
    filtered <- filter_str(filtered, 0, id)
    filtered <- filter_str(filtered, 1, name)
    filtered <- filter_str(filtered, 2, category)
    filtered <- filter_str(filtered, 3, release_year)
    filtered <- filter_int(filtered, 4, price)

    output("List of games at store that match the criteria: ")

    if (length(filtered) = 0) then
        output("There is no game at store that matches the criteria.")
    else
        {adding header to filtered data}
        header <- ["ID", "NAME", "CATEGORY", "RELEASE YEAR", "PRICE",
"STOCK"]

        temp <-  ["*" i traversal [0..(length(filtered)+1)]]
        temp[0] <- header
        i traversal [0..length(filtered)]
            temp[i + 1] <- filtered[i]

        filtered <- temp

        {Generate parsing for non-empty filtered data}
        i traversal [0..length(filtered)]
            if (i = 0) then                          {skip numbering for
the header}
                output("  ", end= " ")
            else                                  {numbering for the list}
                output(str(i) + ".", end=" ")

            j traversal [0..6]
                output(filtered[i][j], end="")
                character_amount <- 0

                k traversal [0..length(filtered)]
                    if (length(str(filtered[k][j])) > character_amount)
                        character_amount <- length(str(filtered[k][j]))

                output((" " * (character_amount -
length(str(filtered[i][j])))), "| ", end="")
            output("")
```

F12 – Topup

```
DICTIONARY
    {
    function length (a) -> integer
    }

FUNCTION/PROCEDURE DEFINITION
Function function_topup (input username : string, input balance :
integer, input user_data : array of array) -> array of array
    {Function to topup the user's balance}
    LOCAL DICTIONARY
    {
    user_valid : boolean
    line_index, current_balance : integer
    }

    ALGORITHM
    user_valid <- False
    line_index <- 0

    i traversal [2..length(user_data)]

        {Checks if the user is a valid user or not}
        if username = user_data[i][2] then
            user_valid <- True
            line_index <- i

    if user_valid = True then

        current_balance <- user_data[line_index][6]

        if balance + current_balance < 0 then
            output("Input not valid")
        else
            current_balance <- current_balance + balance
            user_data[line_index][6] <- current_balance
            -> user_data

    else
        output("Username", username, "not found")

Function topup (input data : array of array) -> array of array
    {Function to get input and inputs it into the function_topup}

    LOCAL DICTIONARY
    {
    username : string
    balance : integer
    }

    ALGORITHM

    input(username, balance)
    data <- function_topup(username, balance, data)
    -> data
```

F13 – History

```
DICTIONARY
    {
    history_data : array of array

    function length (a) -> integer
    function append (a, array) -> array
    }

FUNCTION/PROCEDURE DEFINITION
```

```
Procedure history (input hist_data : array of array)

    {Procedure to print the content of riwayat.csv array in the working
data_history (temporary data matrix)}

    {I.S. hist_data is defined and not empty (minimum 1 element)
    F.S. hist_data is printed}

    LOCAL DICTIONARY
    {
    user_hist_data : array of array
    data_history : array of array
    i, j, k, l, character_amount : integer
    }

    ALGORITHM

    {Loop to check for all the user's history data}
    i traversal [1..length(history_data)]
        if history_data[i][4] = user_id then
            user_hist_data += history_data[i]

    if length(hist_data) = 1 then
        output("Sorry, you haven't bougth any game yet. Enter buy_game to
buy some game.)
    else
        {Generating a temporary list to host data without changing the
original source}
        data_history <- array [1..length(hist_data)] of "*"
        i traversal [2..length(hist_data)]
            data_history[i] <- hist_data[i+1]

        data_history <- append (["HEADING"], data_history)

        {Generating parsing for non-empty data_history list}
        i traversal [1..length(data_history)]
            output(i, ".", end: " ")

            j traversal [1..6]
                output(data_history[i][j], end: "")
                character_amount <- 0

                k traversal [1..length(data_history)]
                    if length(data_history[k][j]) > character_amount then
                        character_amount <- length(data_history[k][j])

                l traversal [1..character_amount-
length(data_history[i][j])]
                    output(" ", end: "")
                output("| ", end:"")
                output("\n")
```

F14 – Help

```
DICTIONARY
    {
    function is_admin (user, save_folder) -> Boolean
    }
```

```
FUNCTION/PROCEDURE DEFINITION
Procedure help (input user : string, input save_folder : string)

    {Procedure to print the instructions for the main program}

    {I.S. user is defined, save-folder is defined, role_validator
function is defined
    F.S. Help instructions are printed}

    LOCAL DICTIONARY
    {
    is_user_admin : boolean
    }

    ALGORITHM
    is_user_admin = is_admin (user, save_folder)

    if is_user_admin = True then
        output("========== HELP ==========")
        output("")
        output("1. register - Register a new user")
        output("2. login - Log in to the program")
        output("3. add_game - Adding a game to the database")
        output("4. change_game - Changing a game in the database")
        output("5. change_stock - Changing the stock of a game in the
database")
        output("6. list_available_game - Gives a list of all the
available game in the store")
        output("7. search_at_store - Searches the store for a game")
        output("8. topup - Top ups the balance of a user")
        output("9. help - Prints this menu")
        output("10. save - Saves the current working database")
        output("11. exit - Exits the program")
        print("12. magicconch : Hears what the great magic conch has to
say")
        print("13. tictactoe : Play TicTacToe")
    else
        output("========== HELP ==========")
        output("")
        output("before logging in:")
        output("1. login - Log in to the program")
        output("2. help - prints this menu")
        output("")
        output("after logging in:")
        output("1. list_available_game - Gives a list of all the
available game in the store")
        output("2. buy_game - Buys a game with the current balance")
        output("3. list_my_game - Lists owned games")
        output("4. search_my_game - Searches owned games")
        output("5. search_at_store - Searches the store for a game")
        output("6. history - Prints the transaction history")
        output("7. help - Prints this menu")
        output("8. save - Saves the current working database")
        output("9. exit - Exits the program")
        print("10. magicconch : Hears what the great magic conch has to
say")
        print("11. tictactoe : Play TicTacToe")
```

F15 – Load

```
DICTIONARY
    {

    function ArgumentParser
    function walk
    }

ALGORITHM
parser <- ArgumentParser()
parser.add_argument("folder", help="the save file that is want to be
loaded")

args <- parser.parse_args()

save_folder <- args.folder
all_folder <- next(walk("Database"))[1]
```

F16 – Save

```
DICTIONARY
    {
    procedure writeline
    function os
    save_folder : string
    }

FUNCTION/PROCEDURE DEFINITION
Procedure saver (input folder : string, input data : array)

    {Procedure to save the data in the program to the database}

    {I.S. folder is defined, data (matrix) is defined
    F.S. The working database is saved to the csv}

    LOCAL DICTIONARY
    {
    path : string
    exist : boolean
    }

    ALGORITHM
    path <- "Database/{folder}"
    exist <- os.path.exists(path)

    if exist then
    {overwrite the data}
        writeline(folder, "game.csv", data[0])
        writeline(folder, "kepemilikan.csv", data[1])
        writeline(folder, "riwayat.csv", data[2])
        writeline(folder, "user.csv", data[3])
    else
    {make a new folder}
        open(path/files, "w")   # Make a new file for every files
        writeline(folder, "game.csv", data[0])
        writeline(folder, "kepemilikan.csv", data[1])
        writeline(folder, "riwayat.csv", data[2])
        writeline(folder, "user.csv", data[3])
```

```
Procedure save (input data : list)
    {Procedure to ask whether to save in the same save folder or a
different one}

    {I.S. saver is defined, data is defined
    F.S. the saver runs with a folder defined (new/existing)}

    LOCAL DICTIONARY
    {
    is_new_folder, new_folder, folder : string
    }

    ALGORITHM
    is_new_folder <- input("Do you wish to save to a new folder? (y/n) ")

    while (is_new_folder != "y") and (is_new_folder != "Y") and
(is_new_folder != "n") and (is_new_folder != "N") do
        {Input Validation}

        output("Unknown input. Please choose between (y/n)")
        is_new_folder <- input("Do you wish to save to a new folder?
(y/n) ")

    if (is_new_folder == "y") or (is_new_folder == "Y") then
        new_folder <- input("folder name: ")
        saver(new_folder, data)

    elif (is_new_folder == "n") or (is_new_folder == "N") then
        folder <- save_folder
        saver(folder, data)
```

F17 – Exit

```
DICTIONARY

    function is string -> quit()


LOCAL DICTIONARY

    x : string


ALGORITHM

      X <- String

            if (x = Y) or (x = y) then

            quit()

if (x = N) or (x = n) then

                  Ø

    -> quit()
```

B01 – Cipher

```
DICTIONARY
    function is_lower (string : string) -> boolean
    { Function to check if a string consists entirely of lowercase
letters. }


function encrypt (password : string) -> string
    { Encrypts user password using the Affine cipher. }

LOCAL DICTIONARY
    a, b : integer  { cipher keys }
    ciphered : string  { ciphered password }

ALGORITHM
    { Hardcoded key }
    a <- 17
    b <- 9
    ciphered <- ''

    { e(x) = (ax + b) mod m }
    char traversal password

        if (97 <= ord(char) <= 122 or 65 <= ord(char) <= 90) then
            if (is_lower(char)) then
                ciphered <- ciphered + chr(((a * (ord(char) - 97) + b) %
26) + 97)
            else  { is_upper(char) }
                ciphered <- ciphered + chr(((a * (ord(char) - 65) + b) %
26) + 65)

        else  { not (97 <= ord(char) <= 122 or 65 <= ord(char) <= 90) }
            ciphered <- ciphered + char

    -> ciphered


function decrypt (ciphered : string) -> string
    { Decrypts ciphered user password using the Affine cipher. }

LOCAL DICTIONARY
    a, b : integer  { cipher keys }
    i : integer
    password : string

ALGORITHM
    { Hardcoded key }
    a <- 17
    b <- 9
    i <- 0
    password <- ''

    { Finding a^(-1) which is the multiplicative inverse of a }
    multiplicative_inverse <- None
    while (multiplicative_inverse = None) do
        if (((i * 26) + 1) / a = ((i * 26) + 1) // a) then
            multiplicative_inverse <- int(((i * 26) + 1) / a)
            break
        else { not (((i * 26) + 1) / a = ((i * 26) + 1) // a) }
            i <- i + 1
    { d(x) = a^(-1)(x - b) mod m }
    char traversal ciphered

        if (97 <= ord(char) <= 122 or 65 <= ord(char) <= 90) then
            if (is_lower(char)) then
                password  <-  password  +  chr(((multiplicative_inverse  *
((ord(char) - 97) - b)) % 26) + 97)
            else  { is_upper(char) }
                password  <-  password  +  chr(((multiplicative_inverse  *
((ord(char) - 65) - b)) % 26) + 65)

        else { not (97 <= ord(char) <= 122 or 65 <= ord(char) <= 90) }
            password <- password + char

    -> password
```

B02 – Magic conch

```
DICTIONARY
    {
    function time () -> integer
    }

FUNCTION/PROCEDURE DEFINITION
Function magicconch () -> string
    {Function that generates a random number with LCG and returns a
string based on the random number}

    LOCAL DICTIONARY
    {
    x, a, c, m, state : integer
    }

    ALGORITHM
    x <- time.time()
    a <- 3
    c <- 1
    m <- 7
    state <- round(((a*x) + c) mod 7)

    {States and outputs}
    if (state == 0) then
        -> "Coba lagi."
    else if (state == 1) then
        -> "Ya."
    else if (state == 2) then
        -> "Tidak."
    else if (state == 3) then
        -> "Mungkin."
    else if (state == 4) then
        -> "Jangan deh."
    else if (state == 5) then
        -> "Tanya lagi nanti."
    else if (state == 6) then
        -> "Terserah dah."
    else if (state == 7) then
        -> "Coba tanya doswal."
    else
        do nothing
```

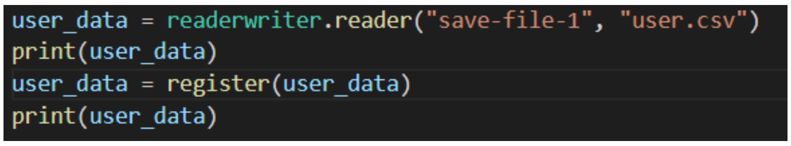
B03 – TicTacToe

```
DICTIONARY

{ not to be imported }
procedure ask_location(input/output matrix : array of array of strings,
input pawn : string)
LOCAL ALGORITHM
    valid : boolean
    x, y : integer

ALGORITHM

    { Procedure to ask input for pawn location and validate it. }
    valid <- False
    while (valid = False) do
        output("["+ pawn + "] turn: ")
        input (x, y)


        # Location validation
        if not ((1<=x<=3) and (1<=y<=3) ) then   {the location does not
exist}
            output("Invalid location. Please try again!")
        else                                {the location exists}


            if (matrix[y-1][x-1] != "#") then     {the location is alrady
occupied}
                output("Location is already filled. Please try again!")
            else                          {teh location is empty}
                matrix[y-1][x-1] <- pawn
                valid <- True


{ not to be imported }
function win_checker(matrix : array of array of strings, pawn : string) -
> string
    {Function to return state of winning of a pawn}


LOCAL DICTIONARY
    win : string


ALGORITHM
    win <- ""        {win = "" --> pawn haven't won yet}


    {horizontal win checker}
    if ((matrix [0][0] = pawn) and (matrix [0][1] = pawn) and (matrix
[0][2] = pawn)) or ((matrix [1][0] = pawn) and (matrix [1][1] = pawn) and
(matrix [1][2] = pawn)) or ((matrix [2][0] = pawn) and (matrix [2][1] =
pawn) and (matrix [2][2] = pawn)) then
        win <- "horizontally"

    {vertical win checker}
    else if ((matrix [0][0] = pawn) and (matrix [1][0] = pawn) and
(matrix [2][0] = pawn)) or ((matrix [0][1] = pawn) and (matrix [1][1] =
pawn) and (matrix [2][1] = pawn)) or ((matrix [0][2] = pawn) and (matrix
[1][2] = pawn) and (matrix [2][2] = pawn)) then
        win <- "vertically"

    # diagonal win checker
    else if ((matrix [0][0] = pawn) and (matrix [1][1] = pawn) and
(matrix [2][2] = pawn)) or ((matrix [0][2] = pawn) and (matrix [1][1] =
pawn) and (matrix [2][0] = pawn)) then
        win <- "diagonally"

    -> win


{ not to be imported }
procedure status(input/output matrix : array of array of strings)
    {Procedure to print out board status.}

LOCAL DICTIONARY
    i,j : integer

ALGORITHM
    output("============")
    output("Board Status:")

    {Generate parsing for matrix}
    i traversal [0..2]
        output("|", end=" ")
        j taversal [0..2]
            output(matrix[i][j], end=" ")
            output("|", end=" ")
        output("")


procedure tictactoe ()
    {Procedure to simulate tic tac toe game}

LOCAL DICTIONARY
    matrix : array of array of characters
    turn : integer
    pawn, string : string

ALGORITHM
    matrix <- [["#", "#", "#"], ["#", "#", "#"], ["#", "#", "#"]]
    turn <- 0

    while (turn<=9) do
        turn <- turn + 1

        if turn%2 = 1 then
            pawn <- "X"
        else
            pawn <- "O"

        status(matrix)                    { print out board status }
        ask_location(matrix, pawn)        { ask for user input of pawn
location }

        { check if pawn wins }
        win <- win_checker(matrix, pawn)
        if win != "" then                  {(win = "")--> meaning pawn
haven't won yet }
            status(matrix)
            if (win = "horizontally") then
                output(pawn, "won horizontally. Victory applies to other
row.")
            else if (win = "vertically") then
                output(pawn, "won vertically. Victory applies to other
column.")
            else           {win = "diagonally"}
                output(pawn, "won diagonally. Victory applies to the
opposite diagonal.")

            break

        else             { win == "" --> pawn haven't won }
            pass


        { turn == 9 is the last turn; tie statement will be skipped if
there is already a winner }
        if (turn = 9) then
            status(matrix)
            output("Tie. There is no winner.")
            break
```

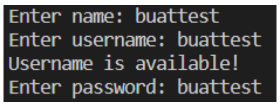
VIII. HASIL SCREENSHOT PENGUJIAN PROGRAM BERDASARKAN FITUR-FITUR PADA SPESIFIKASI

F02 – Register

Input

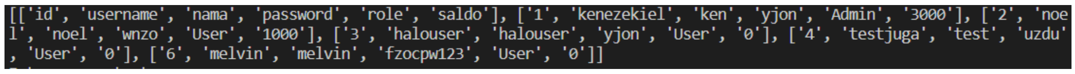

```python
user_data = readerwriter.reader("save-file-1", "user.csv")
print(user_data)
user_data = register(user_data)
print(user_data)
```

Gambar 9.2.1 Cara memanggil prosedur dalam modul F02

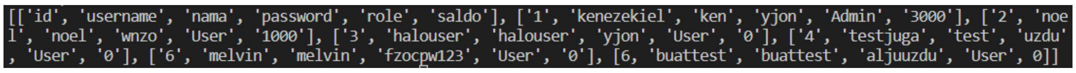

```
Enter name: buattest
Enter username: buattest
Username is available!
Enter password: buattest
```

Gambar 9.2.2 Input untuk modul F02

Output

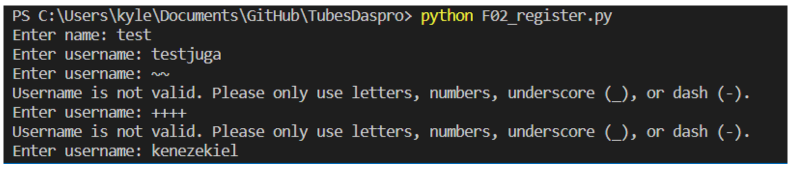

```
[['id', 'username', 'nama', 'password', 'role', 'saldo'], ['1', 'kenezekiel', 'ken', 'yjon', 'Admin', '3000'], ['2', 'noe
l', 'noel', 'wnzo', 'User', '1000'], ['3', 'halouser', 'halouser', 'yjon', 'User', '0'], ['4', 'testjuga', 'test', 'uzdu'
, 'User', '0'], ['6', 'melvin', 'melvin', 'fzocpw123', 'User', '0']]
```

Gambar 9.2.3 Sebelum modifikasi register

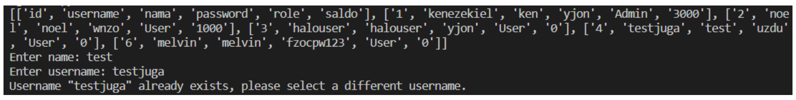

```
[['id', 'username', 'nama', 'password', 'role', 'saldo'], ['1', 'kenezekiel', 'ken', 'yjon', 'Admin', '3000'], ['2', 'noe
l', 'noel', 'wnzo', 'User', '1000'], ['3', 'halouser', 'halouser', 'yjon', 'User', '0'], ['4', 'testjuga', 'test', 'uzdu'
, 'User', '0'], ['6', 'melvin', 'melvin', 'fzocpw123', 'User', '0'], [6, 'buattest', 'buattest', 'aljuuzdu', 'User', 0]]
```

Gambar 9.2.4 Output untuk modul F02

Validasi

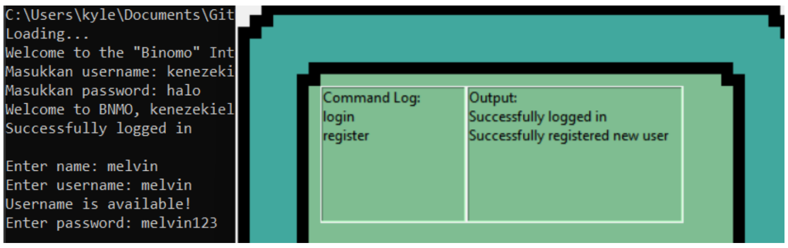

```
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python F02_register.py
Enter name: test
Enter username: testjuga
Enter username: ~~
Username is not valid. Please only use letters, numbers, underscore (_), or dash (-).
Enter username: ++++
Username is not valid. Please only use letters, numbers, underscore (_), or dash (-).
Enter username: kenezekiel
```

Gambar 9.2.5 Testing validasi input untuk input modul F02


```
[['id', 'username', 'nama', 'password', 'role', 'saldo'], ['1', 'kenezekiel', 'ken', 'yjon', 'Admin', '3000'], ['2', 'noe
l', 'noel', 'wnzo', 'User', '1000'], ['3', 'halouser', 'halouser', 'yjon', 'User', '0'], ['4', 'testjuga', 'test', 'uzdu'
, 'User', '0'], ['6', 'melvin', 'melvin', 'fzocpw123', 'User', '0']]
Enter name: test
Enter username: testjuga
Username "testjuga" already exists, please select a different username.
```

Gambar 9.2.6 Testing validasi username untuk input modul F02

Implementasi di Main Program

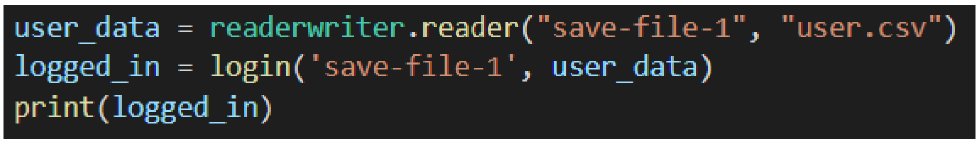

```
C:\Users\kyle\Documents\Git
Loading...
Welcome to the "Binomo" Int
Masukkan username: kenezeki
Masukkan password: halo
Welcome to BNMO, kenezekiel
Successfully logged in

Enter name: melvin
Enter username: melvin
Username is available!
Enter password: melvin123
```

```
Command Log:   Output:
login          Successfully logged in
register       Successfully registered new user
```

Gambar 9.2.7 Contoh implementasi modul F02 pada main program

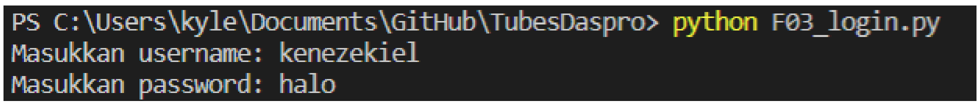
F03 – Login

Input



```
user_data = readerwriter.reader("save-file-1", "user.csv")
logged_in = login('save-file-1', user_data)
print(logged_in)
```

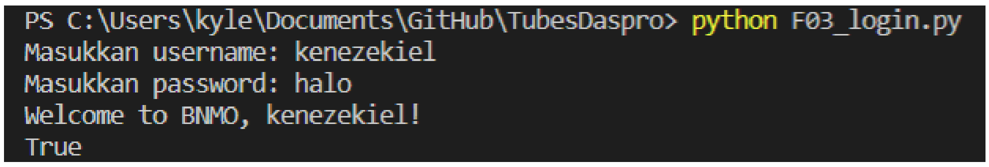
Gambar 9.3.1 Cara memanggil fungsi dalam modul F03



```
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python F03_login.py
Masukkan username: kenezekiel
Masukkan password: halo
```

Gambar 9.3.2 Input untuk modul F03

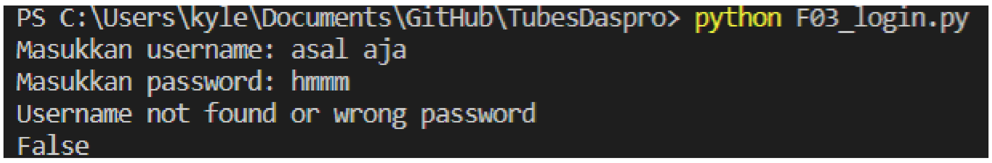
Output



```
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python F03_login.py
Masukkan username: kenezekiel
Masukkan password: halo
Welcome to BNMO, kenezekiel!
True
```

Gambar 9.3.3 Output untuk modul F03

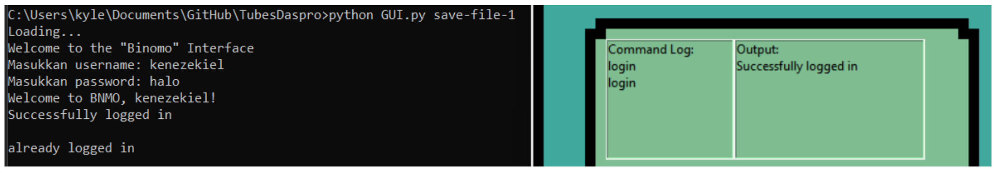
Validasi



```
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python F03_login.py
Masukkan username: asal aja
Masukkan password: hmmm
Username not found or wrong password
False
```

Gambar 9.3.4 Testing validasi untuk input modul F03

Implementasi di Main Program



```
C:\Users\kyle\Documents\GitHub\TubesDaspro>python GUI.py save-file-1
Loading...
Welcome to the "Binomo" Interface
Masukkan username: kenezekiel
Masukkan password: halo
Welcome to BNMO, kenezekiel!
Successfully logged in

already logged in
```

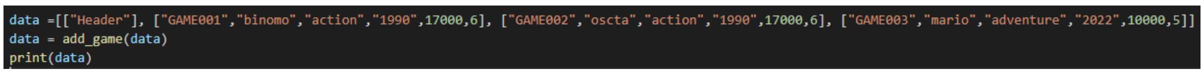
```
Command Log:
login
login
```

```
Output:
Successfully logged in
```

Gambar 9.3.5 Contoh implementasi modul F03 pada main program

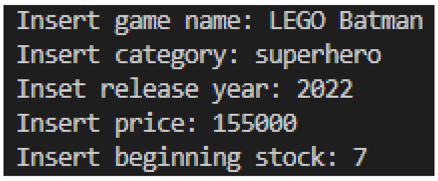
F04 – Menambahkan Game

Input



```
data =[["Header"], ["GAME001","binomo","action","1990",17000,6], ["GAME002","oscta","action","1990",17000,6], ["GAME003","mario","adventure","2022",10000,5]]
data = add_game(data)
print(data)
```

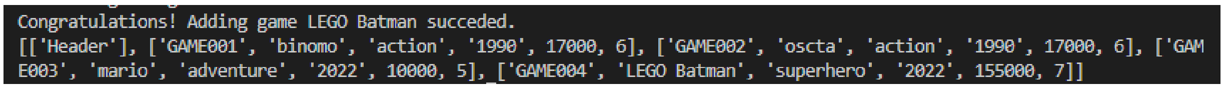
Gambar 9.4.1 Cara memanggil modul F04



```
Insert game name: LEGO Batman
Insert category: superhero
Inset release year: 2022
Insert price: 155000
Insert beginning stock: 7
```

Gambar 9.4.2 Input untuk modul F04

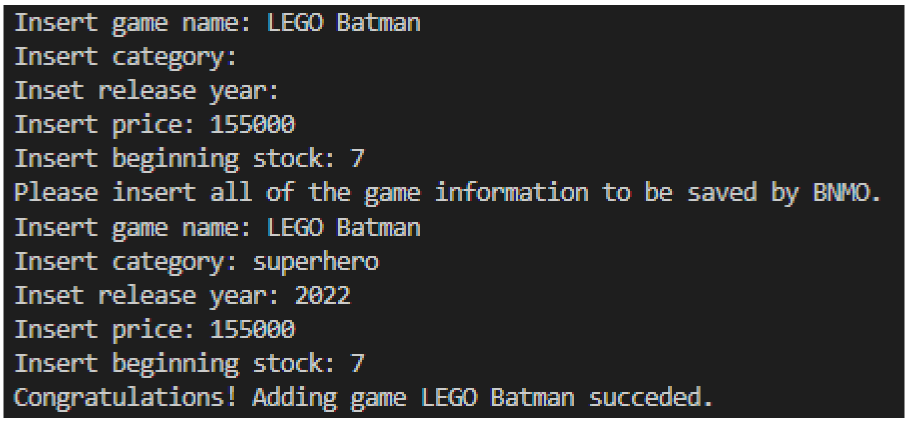
Output



```
Congratulations! Adding game LEGO Batman succeded.
[['Header'], ['GAME001', 'binomo', 'action', '1990', 17000, 6], ['GAME002', 'oscta', 'action', '1990', 17000, 6], ['GAM
E003', 'mario', 'adventure', '2022', 10000, 5], ['GAME004', 'LEGO Batman', 'superhero', '2022', 155000, 7]]
```

Gambar 9.4.3 Output untuk modul F04

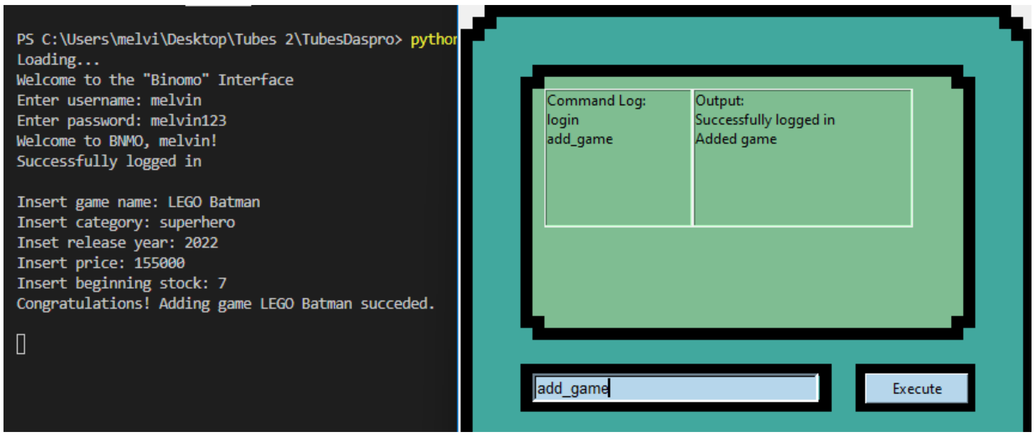
Validasi



```
Insert game name: LEGO Batman
Insert category:
Inset release year:
Insert price: 155000
Insert beginning stock: 7
Please insert all of the game information to be saved by BNMO.
Insert game name: LEGO Batman
Insert category: superhero
Inset release year: 2022
Insert price: 155000
Insert beginning stock: 7
Congratulations! Adding game LEGO Batman succeded.
```

Gambar 9.4.4 Validasi untuk modul F04

Implementasi di Main Program



```
PS C:\Users\melvi\Desktop\Tubes 2\TubesDaspro> pythor
Loading...
Welcome to the "Binomo" Interface
Enter username: melvin
Enter password: melvin123
Welcome to BNMO, melvin!
Successfully logged in

Insert game name: LEGO Batman
Insert category: superhero
Inset release year: 2022
Insert price: 155000
Insert beginning stock: 7
Congratulations! Adding game LEGO Batman succeded.
```

```
Command Log:
login
add_game
```

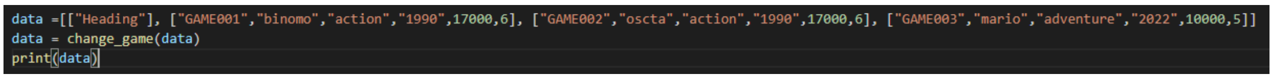
```
Output:
Successfully logged in
Added game
```

```
add_game
```

Gambar 9.4.5 Contoh implementasi modul F04 pada main program

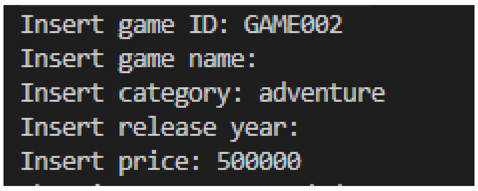
F05 – Mengubah Game

Input



```python
data =[["Heading"], ["GAME001","binomo","action","1990",17000,6], ["GAME002","oscta","action","1990",17000,6], ["GAME003","mario","adventure","2022",10000,5]]
data = change_game(data)
print(data)
```

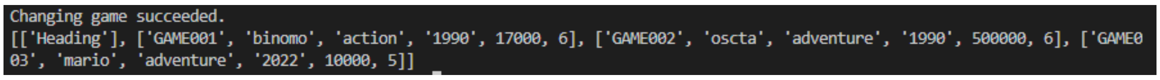
Gambar 9.5.1 Cara memanggil modul F05



```
Insert game ID: GAME002
Insert game name:
Insert category: adventure
Insert release year:
Insert price: 500000
```

Gambar 9.5.2 Input untuk modul F05

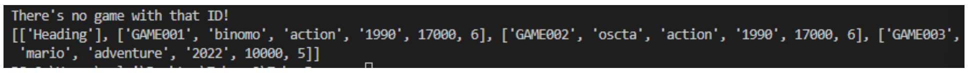
Output



```
Changing game succeeded.
[['Heading'], ['GAME001', 'binomo', 'action', '1990', 17000, 6], ['GAME002', 'oscta', 'adventure', '1990', 500000, 6], ['GAME0
03', 'mario', 'adventure', '2022', 10000, 5]]
```

Gambar 9.5.3 Output untuk modul F05

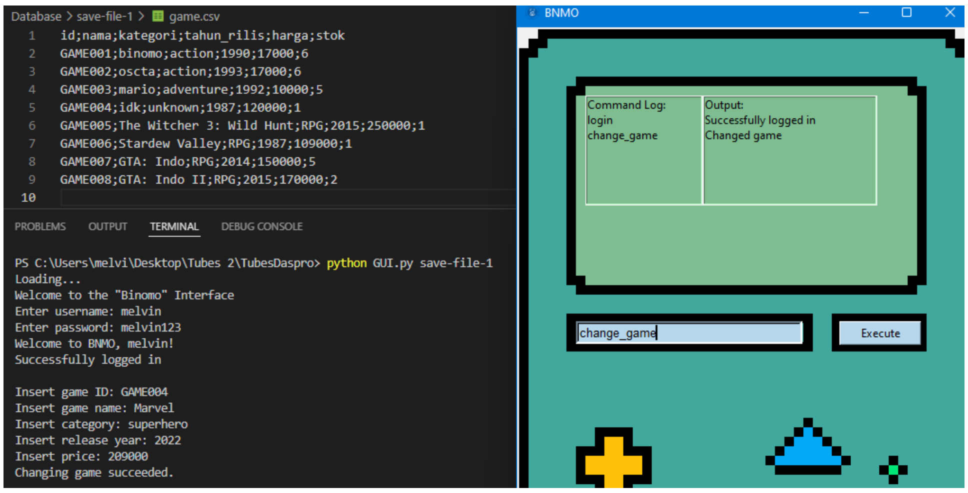
Validasi



```
There's no game with that ID!
[['Heading'], ['GAME001', 'binomo', 'action', '1990', 17000, 6], ['GAME002', 'oscta', 'action', '1990', 17000, 6], ['GAME003',
 'mario', 'adventure', '2022', 10000, 5]]
```

Gambar 9.5.4 Validasi untuk modul F05

Implementasi di Main Program



```
id;nama;kategori;tahun_rilis;harga;stok
GAME001;binomo;action;1990;17000;6
GAME002;oscta;action;1993;17000;6
GAME003;mario;adventure;1992;10000;5
GAME004;idk;unknown;1987;120000;1
GAME005;The Witcher 3: Wild Hunt;RPG;2015;250000;1
GAME006;Stardew Valley;RPG;1987;109000;1
GAME007;GTA: Indo;RPG;2014;150000;5
GAME008;GTA: Indo II;RPG;2015;170000;2
```

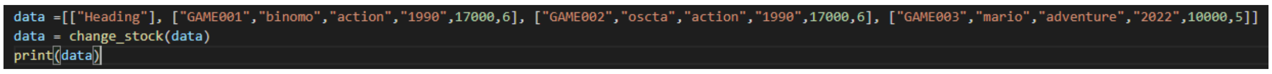
```
PS C:\Users\melvi\Desktop\Tubes 2\TubesDaspro> python GUI.py save-file-1
Loading...
Welcome to the "Binomo" Interface
Enter username: melvin
Enter password: melvin123
Welcome to BNMO, melvin!
Successfully logged in

Insert game ID: GAME004
Insert game name: Marvel
Insert category: superhero
Insert release year: 2022
Insert price: 209000
Changing game succeeded.
```

```
Command Log:        Output:
login               Successfully logged in
change_game         Changed game
```

Gambar 9.5.5 Contoh implementasi modul F05 pada main program

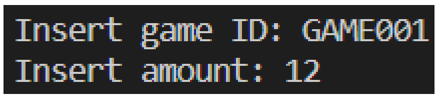
F06 – Mengubah Stok Game

Input



```
data =[["Heading"], ["GAME001","binomo","action","1990",17000,6], ["GAME002","oscta","action","1990",17000,6], ["GAME003","mario","adventure","2022",10000,5]]
data = change_stock(data)
print(data)
```

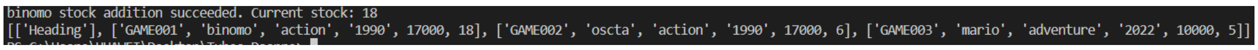
Gambar 9.6.1 Cara memanggil modul F06



```
Insert game ID: GAME001
Insert amount: 12
```

Gambar 9.6.2 Input untuk modul F06

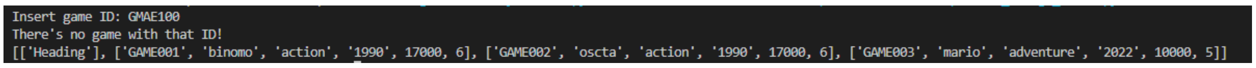
Output



```
binomo stock addition succeeded. Current stock: 18
[['Heading'], ['GAME001', 'binomo', 'action', '1990', 17000, 18], ['GAME002', 'oscta', 'action', '1990', 17000, 6], ['GAME003', 'mario', 'adventure', '2022', 10000, 5]]
```

Gambar 9.6.3 Output untuk modul F06

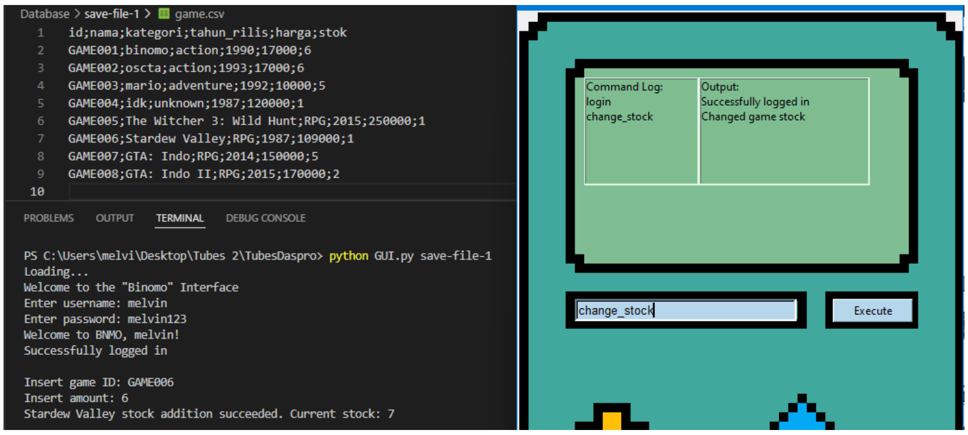
Validasi



```
Insert game ID: GMAE100
There's no game with that ID!
[['Heading'], ['GAME001', 'binomo', 'action', '1990', 17000, 6], ['GAME002', 'oscta', 'action', '1990', 17000, 6], ['GAME003', 'mario', 'adventure', '2022', 10000, 5]]
```

Gambar 9.6.4 Validasi untuk modul F06

Implementasi di Main Program


```
Database > save-file-1 > game.csv
id;nama;kategori;tahun_rilis;harga;stok
GAME001;binomo;action;1990;17000;6
GAME002;oscta;action;1993;17000;6
GAME003;mario;adventure;1992;10000;5
GAME004;idk;unknown;1987;120000;1
GAME005;The Witcher 3: Wild Hunt;RPG;2015;250000;1
GAME006;Stardew Valley;RPG;1987;109000;1
GAME007;GTA: Indo;RPG;2014;150000;5
GAME008;GTA: Indo II;RPG;2015;170000;2
```

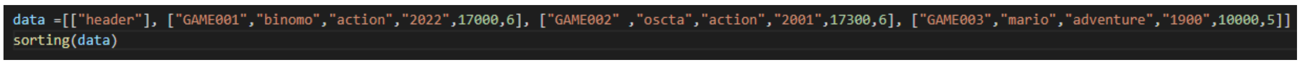
```
PS C:\Users\melvi\Desktop\Tubes 2\TubesDaspro> python GUI.py save-file-1
Loading...
Welcome to the "Binomo" Interface
Enter username: melvin
Enter password: melvin123
Welcome to BNMO, melvin!
Successfully logged in

Insert game ID: GAME006
Insert amount: 6
Stardew Valley stock addition succeeded. Current stock: 7
```

```
Command Log:      Output:
login             Successfully logged in
change_stock      Changed game stock
```

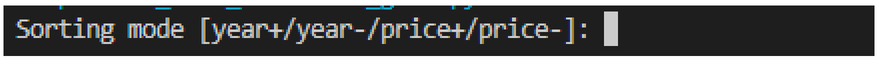
Gambar 9.6.5 Contoh implementasi modul F06 pada main program

F07 – Listing Game di Toko Berdasarkan ID, Tahun Rilis dan Harga

Input

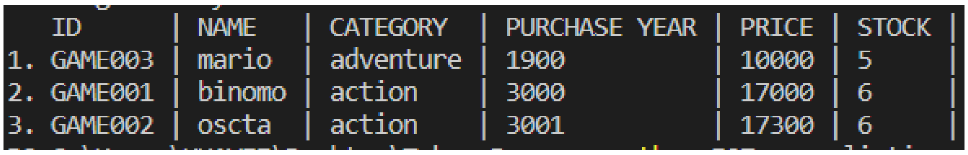

```
data =[["header"], ["GAME001","binomo","action","2022",17000,6], ["GAME002" ,"oscta","action","2001",17300,6], ["GAME003","mario","adventure","1900",10000,5]]
sorting(data)
```

Gambar 9.7.1 Cara memanggil modul F07


```
Sorting mode [year+/year-/price+/price-]:
```

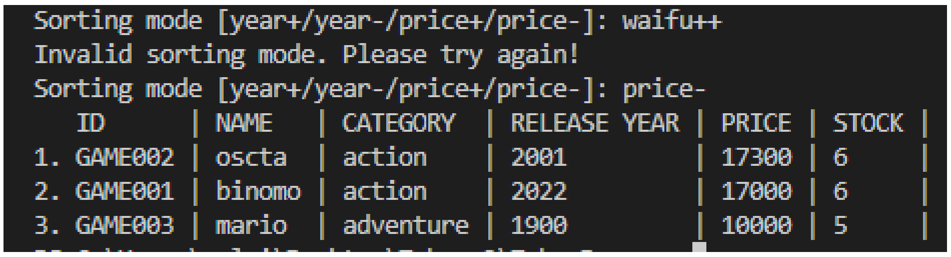
Gambar 9.7.2 Input untuk modul F07

Output


```
   ID        | NAME    | CATEGORY   | PURCHASE YEAR  | PRICE  | STOCK  |
1. GAME003  | mario   | adventure  | 1900           | 10000  | 5      |
2. GAME001  | binomo  | action     | 3000           | 17000  | 6      |
3. GAME002  | oscta   | action     | 3001           | 17300  | 6      |
```

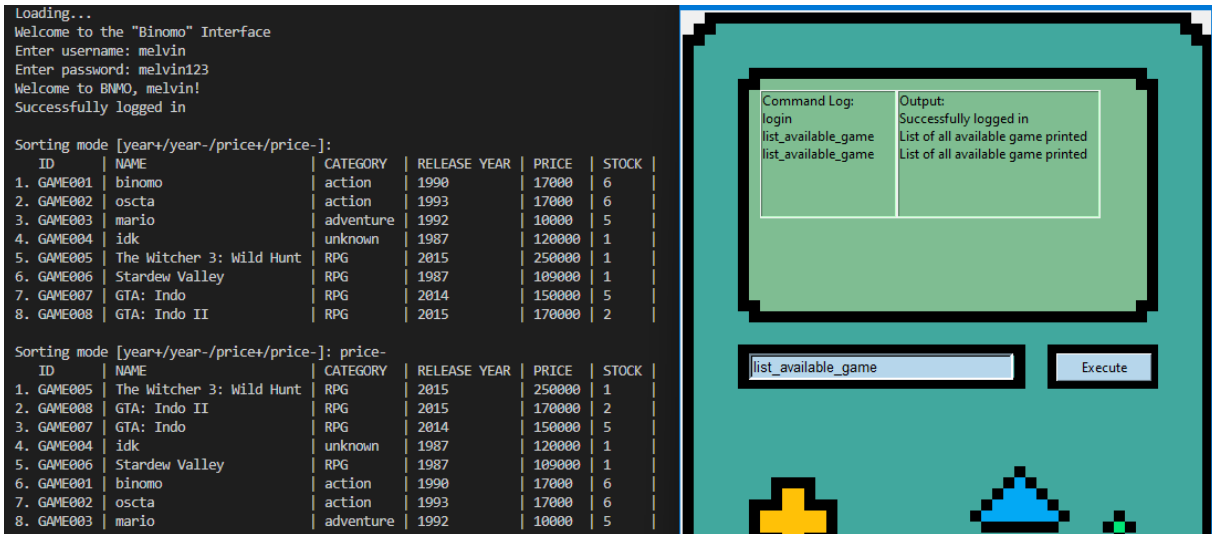
Gambar 9.7.3 Output untuk modul F07

Validasi


```
Sorting mode [year+/year-/price+/price-]: waifu++
Invalid sorting mode. Please try again!
Sorting mode [year+/year-/price+/price-]: price-
   ID        | NAME    | CATEGORY   | RELEASE YEAR  | PRICE  | STOCK  |
1. GAME002  | oscta   | action     | 2001          | 17300  | 6      |
2. GAME001  | binomo  | action     | 2022          | 17000  | 6      |
3. GAME003  | mario   | adventure  | 1900          | 10000  | 5      |
```

Gambar 9.7.4 Validasi untuk modul F07

Implementasi di Main Program



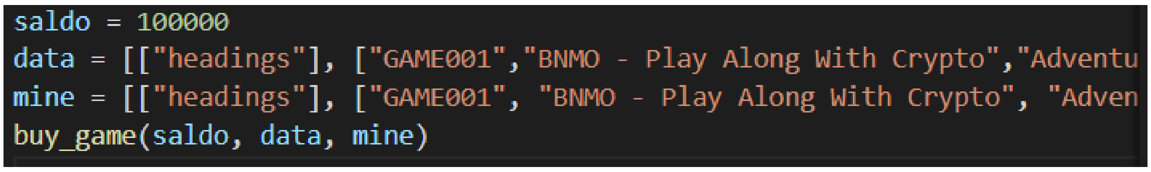
```
Loading...
Welcome to the "Binomo" Interface
Enter username: melvin
Enter password: melvin123
Welcome to BNMO, melvin!
Successfully logged in

Sorting mode [year+/year-/price+/price-]:
   ID       | NAME                      | CATEGORY  | RELEASE YEAR | PRICE  | STOCK |
1. GAME001 | binomo                    | action    | 1990         | 17000  | 6     |
2. GAME002 | oscta                     | action    | 1993         | 17000  | 6     |
3. GAME003 | mario                     | adventure | 1992         | 10000  | 5     |
4. GAME004 | idk                       | unknown   | 1987         | 120000 | 1     |
5. GAME005 | The Witcher 3: Wild Hunt | RPG       | 2015         | 250000 | 1     |
6. GAME006 | Stardew Valley            | RPG       | 1987         | 109000 | 1     |
7. GAME007 | GTA: Indo                 | RPG       | 2014         | 150000 | 5     |
8. GAME008 | GTA: Indo II              | RPG       | 2015         | 170000 | 2     |

Sorting mode [year+/year-/price+/price-]: price-
   ID       | NAME                      | CATEGORY  | RELEASE YEAR | PRICE  | STOCK |
1. GAME005 | The Witcher 3: Wild Hunt | RPG       | 2015         | 250000 | 1     |
2. GAME008 | GTA: Indo II              | RPG       | 2015         | 170000 | 2     |
3. GAME007 | GTA: Indo                 | RPG       | 2014         | 150000 | 5     |
4. GAME004 | idk                       | unknown   | 1987         | 120000 | 1     |
5. GAME006 | Stardew Valley            | RPG       | 1987         | 109000 | 1     |
6. GAME001 | binomo                    | action    | 1990         | 17000  | 6     |
7. GAME002 | oscta                     | action    | 1993         | 17000  | 6     |
8. GAME003 | mario                     | adventure | 1992         | 10000  | 5     |
```

```
Command Log:          Output:
login                 Successfully logged in
list_available_game   List of all available game printed
list_available_game   List of all available game printed
```

Gambar 9.7.5 Contoh implementasi modul F07 pada main program

F08 – Membeli Game

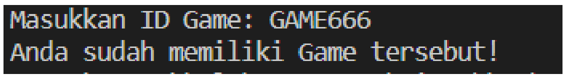
Input



```
saldo = 100000
data = [["headings"], ["GAME001","BNMO - Play Along With Crypto","Adventu
mine = [["headings"], ["GAME001", "BNMO - Play Along With Crypto", "Adven
buy_game(saldo, data, mine)
```

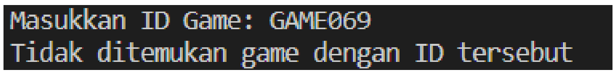
Gambar 9.8.1 Input untuk modul F08

Output



```
Masukkan ID Game: GAME666
Anda sudah memiliki Game tersebut!
```

Gambar 9.8.2 Output untuk modul F08



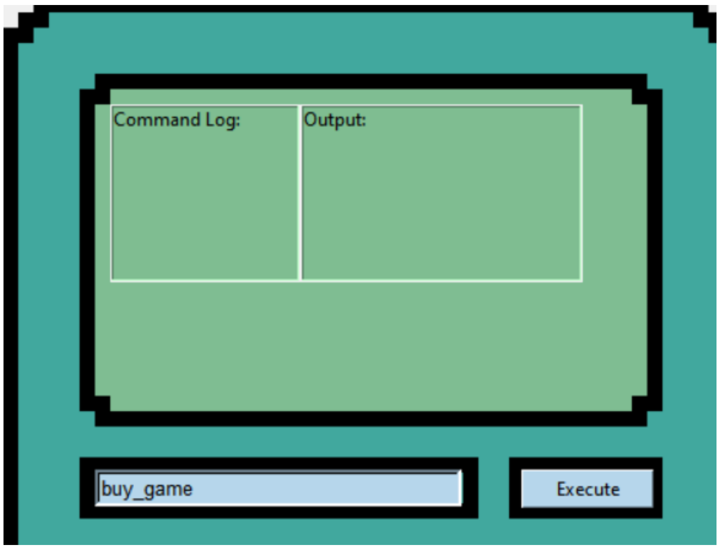
```
Masukkan ID Game: GAME069
Tidak ditemukan game dengan ID tersebut
```

Gambar 9.8.3 Input untuk modul F08

Validasi

Tidak terdapat validasi pada modul ini

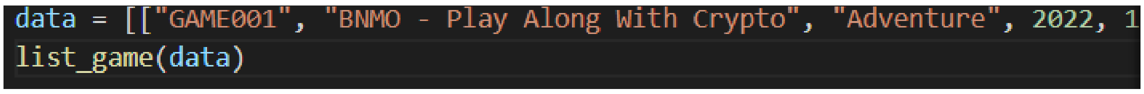
Implementasi di Main Program



Gambar 9.8.4 Input untuk modul F08

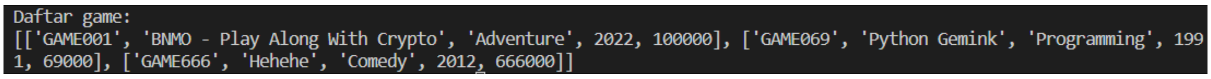
F09 – Melihat Game yang dimiliki

Input - Output



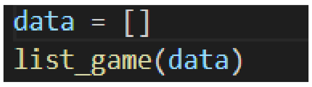
```
data = [["GAME001", "BNMO - Play Along With Crypto", "Adventure", 2022, 1
list_game(data)
```

Gambar 9.9.1 Input pertama untuk modul F09



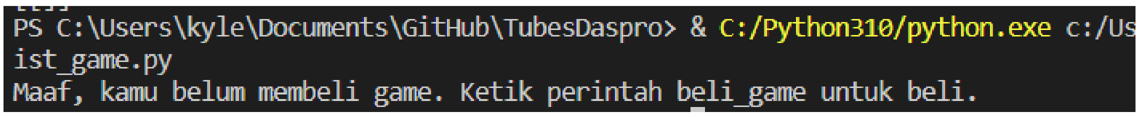
```
Daftar game:
[['GAME001', 'BNMO - Play Along With Crypto', 'Adventure', 2022, 100000], ['GAME069', 'Python Gemink', 'Programming', 199
1, 69000], ['GAME666', 'Hehehe', 'Comedy', 2012, 666000]]
```

Gambar 9.9.2 Output pertama untuk modul F09



```
data = []
list_game(data)
```

Gambar 9.9.3 Input kedua untuk modul F09



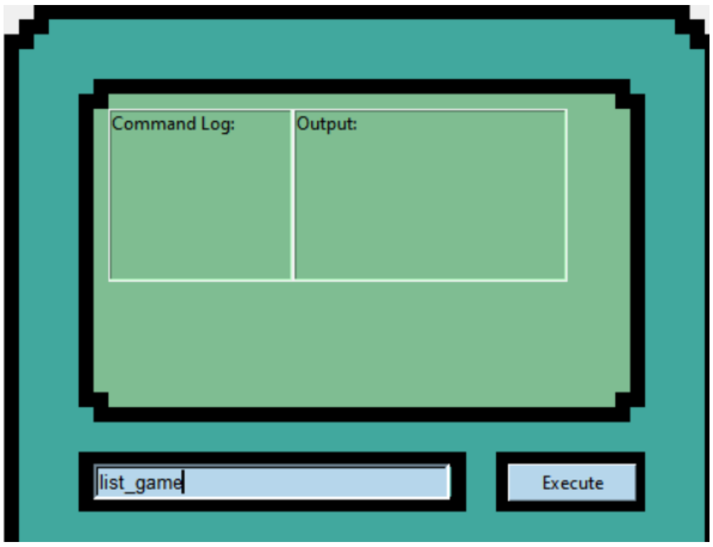
```
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> & C:/Python310/python.exe c:/Us
ist_game.py
Maaf, kamu belum membeli game. Ketik perintah beli_game untuk beli.
```

Gambar 9.9.4 Output kedua untuk modul F09

Validasi

Tidak terdapat validasi pada modul ini

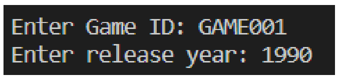
Implementasi di Main Program



Gambar 9.9.5 Implementasi untuk modul F09

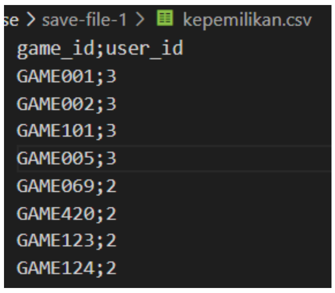
F10 – Mencari Game yang Dimiliki

Input



```
Enter Game ID: GAME001
Enter release year: 1990
```

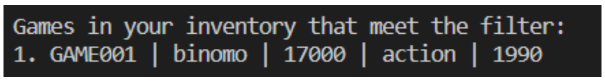
Gambar 9.10.1 Input untuk modul F10



```
game_id;user_id
GAME001;3
GAME002;3
GAME101;3
GAME005;3
GAME069;2
GAME420;2
GAME123;2
GAME124;2
```

Gambar 9.10.2 Initial Database untuk pengecekan

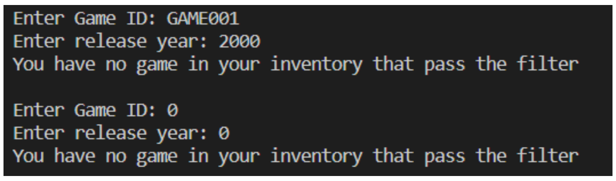
Output



```
Games in your inventory that meet the filter:
1. GAME001 | binomo | 17000 | action | 1990
```

Gambar 9.10.3 Output untuk modul F10

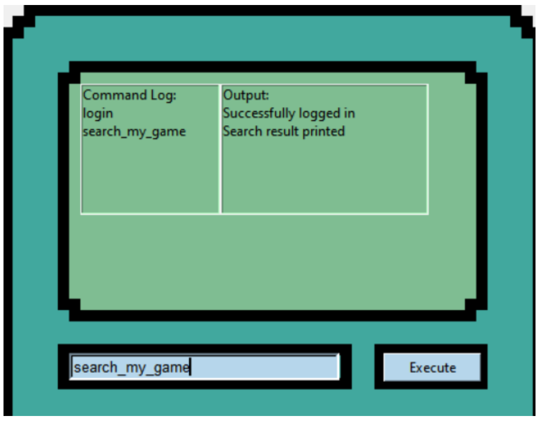
Validasi



```
Enter Game ID: GAME001
Enter release year: 2000
You have no game in your inventory that pass the filter

Enter Game ID: 0
Enter release year: 0
You have no game in your inventory that pass the filter
```

Gambar 9.10.4 Validasi untuk modul F10

Implementasi di Main Program



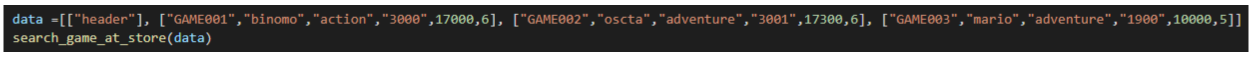
```
Command Log:
login
search_my_game
```

```
Output:
Successfully logged in
Search result printed
```

Gambar 9.10.5 Implementasi pemanggilan modul F10 pada main program

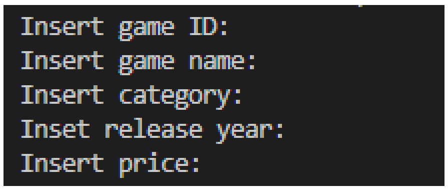
F11 – Mencari Game di Toko dari ID, Nama Game, Harga, Kategori dan Tahun Rilis

Input



```
data =[["header"], ["GAME001","binomo","action","3000",17000,6], ["GAME002","oscta","adventure","3001",17300,6], ["GAME003","mario","adventure","1900",10000,5]]
search_game_at_store(data)
```

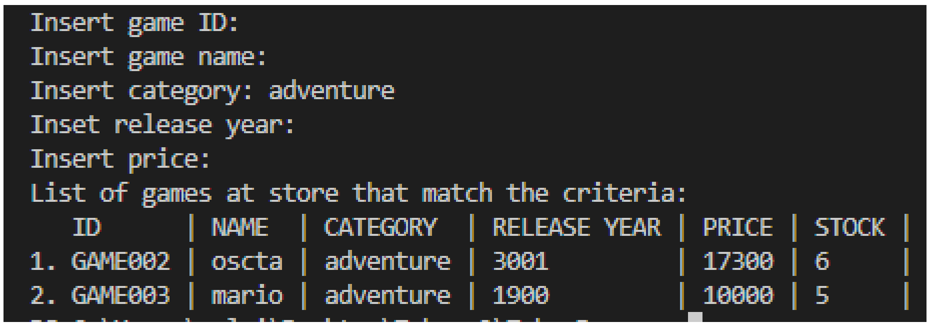
Gambar 9.11.1 Cara memanggil modul F11



```
Insert game ID:
Insert game name:
Insert category:
Inset release year:
Insert price:
```

Gambar 9.11.2 Input untuk modul F11

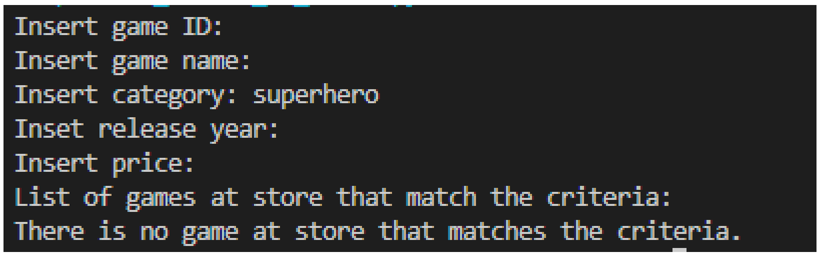
Output



```
Insert game ID:
Insert game name:
Insert category: adventure
Inset release year:
Insert price:
List of games at store that match the criteria:
   ID      | NAME  | CATEGORY  | RELEASE YEAR | PRICE | STOCK |
1. GAME002 | oscta | adventure | 3001         | 17300 | 6     |
2. GAME003 | mario | adventure | 1900         | 10000 | 5     |
```

Gambar 9.11.3 Output untuk modul F11

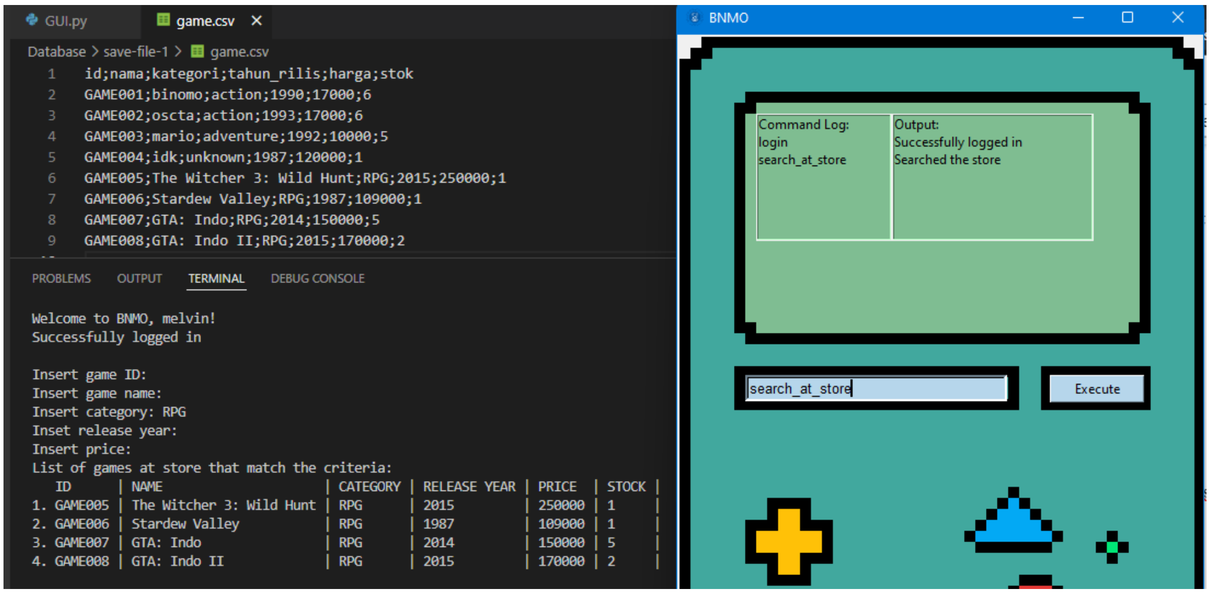
Validasi



```
Insert game ID:
Insert game name:
Insert category: superhero
Inset release year:
Insert price:
List of games at store that match the criteria:
There is no game at store that matches the criteria.
```

Gambar 9.11.4 Validasi untuk modul F11

Implementasi di Main Program



```
Database > save-file-1 > game.csv
id;nama;kategori;tahun_rilis;harga;stok
GAME001;binomo;action;1990;17000;6
GAME002;oscta;action;1993;17000;6
GAME003;mario;adventure;1992;10000;5
GAME004;idk;unknown;1987;120000;1
GAME005;The Witcher 3: Wild Hunt;RPG;2015;250000;1
GAME006;Stardew Valley;RPG;1987;109000;1
GAME007;GTA: Indo;RPG;2014;150000;5
GAME008;GTA: Indo II;RPG;2015;170000;2
```

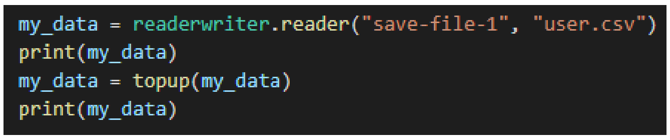
```
Welcome to BNMO, melvin!
Successfully logged in

Insert game ID:
Insert game name:
Insert category: RPG
Inset release year:
Insert price:
List of games at store that match the criteria:
   ID      | NAME                     | CATEGORY | RELEASE YEAR | PRICE  | STOCK |
1. GAME005 | The Witcher 3: Wild Hunt | RPG      | 2015         | 250000 | 1     |
2. GAME006 | Stardew Valley           | RPG      | 1987         | 109000 | 1     |
3. GAME007 | GTA: Indo                | RPG      | 2014         | 150000 | 5     |
4. GAME008 | GTA: Indo II             | RPG      | 2015         | 170000 | 2     |
```

```
Command Log:        Output:
login               Successfully logged in
search_at_store     Searched the store
```

Gambar 9.11.5 Contoh implementasi modul F11 pada main program

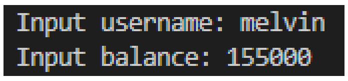
F12 – Top Up Saldo

Input



```python
my_data = readerwriter.reader("save-file-1", "user.csv")
print(my_data)
my_data = topup(my_data)
print(my_data)
```

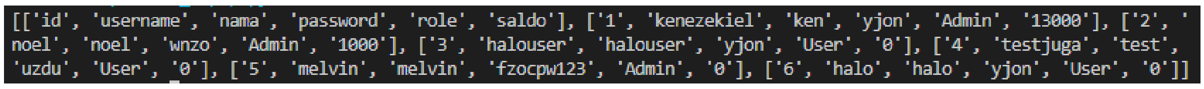
Gambar 9.12.1 Cara memanggil prosedur dalam modul F12



```
Input username: melvin
Input balance: 155000
```

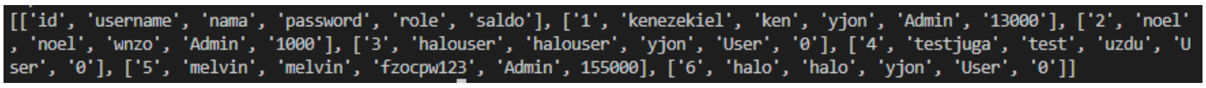
Gambar 9.12.2 Input untuk modul F12

Output



```
[['id', 'username', 'nama', 'password', 'role', 'saldo'], ['1', 'kenezekiel', 'ken', 'yjon', 'Admin', '13000'], ['2', '
noel', 'noel', 'wnzo', 'Admin', '1000'], ['3', 'halouser', 'halouser', 'yjon', 'User', '0'], ['4', 'testjuga', 'test',
'uzdu', 'User', '0'], ['5', 'melvin', 'melvin', 'fzocpw123', 'Admin', '0'], ['6', 'halo', 'halo', 'yjon', 'User', '0']]
```

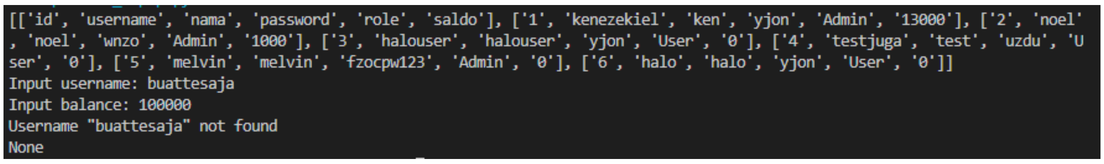
Gambar 9.12.3 Sebelum modifikasi register



```
[['id', 'username', 'nama', 'password', 'role', 'saldo'], ['1', 'kenezekiel', 'ken', 'yjon', 'Admin', '13000'], ['2', 'noel'
, 'noel', 'wnzo', 'Admin', '1000'], ['3', 'halouser', 'halouser', 'yjon', 'User', '0'], ['4', 'testjuga', 'test', 'uzdu', 'U
ser', '0'], ['5', 'melvin', 'melvin', 'fzocpw123', 'Admin', 155000], ['6', 'halo', 'halo', 'yjon', 'User', '0']]
```

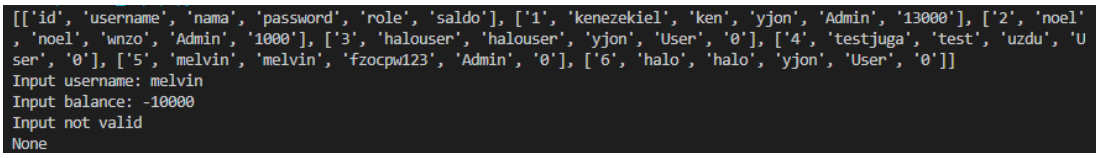
Gambar 9.12.4 Output untuk modul F12

Validasi



```
[['id', 'username', 'nama', 'password', 'role', 'saldo'], ['1', 'kenezekiel', 'ken', 'yjon', 'Admin', '13000'], ['2', 'noel'
, 'noel', 'wnzo', 'Admin', '1000'], ['3', 'halouser', 'halouser', 'yjon', 'User', '0'], ['4', 'testjuga', 'test', 'uzdu', 'U
ser', '0'], ['5', 'melvin', 'melvin', 'fzocpw123', 'Admin', '0'], ['6', 'halo', 'halo', 'yjon', 'User', '0']]
Input username: buattesaja
Input balance: 100000
Username "buattesaja" not found
None
```

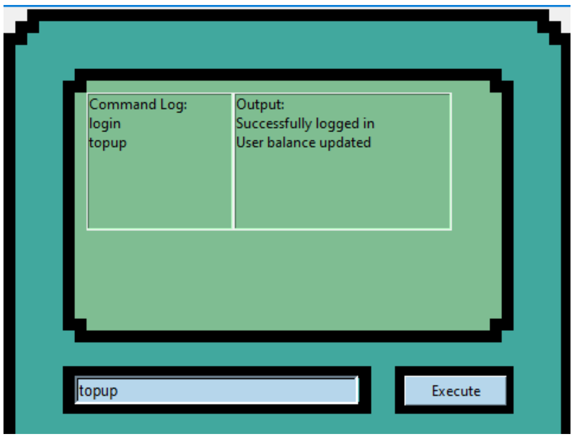
Gambar 9.12.5 Testing validasi username untuk input modul F12



```
[['id', 'username', 'nama', 'password', 'role', 'saldo'], ['1', 'kenezekiel', 'ken', 'yjon', 'Admin', '13000'], ['2', 'noel'
, 'noel', 'wnzo', 'Admin', '1000'], ['3', 'halouser', 'halouser', 'yjon', 'User', '0'], ['4', 'testjuga', 'test', 'uzdu', 'U
ser', '0'], ['5', 'melvin', 'melvin', 'fzocpw123', 'Admin', '0'], ['6', 'halo', 'halo', 'yjon', 'User', '0']]
Input username: melvin
Input balance: -10000
Input not valid
None
```

Gambar 9.12.6 Testing validasi saldo untuk input modul F12

Implementasi di Main Program

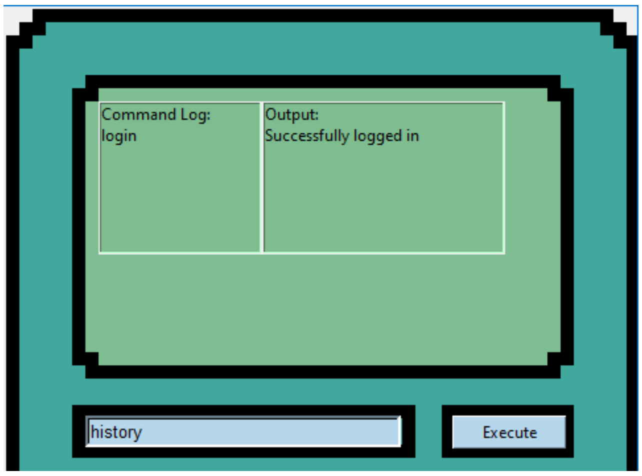


```
Command Log:
login
topup

Output:
Successfully logged in
User balance updated
```

Gambar 9.12.7 Contoh implementasi modul F12 pada main program

F13 – Melihat Riwayat Pembelian

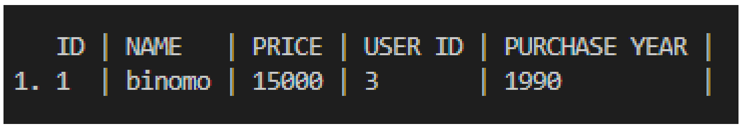
Input



```
Command Log:
login

Output:
Successfully logged in
```

Gambar 9.13.1 Cara memanggil modul F13

Output



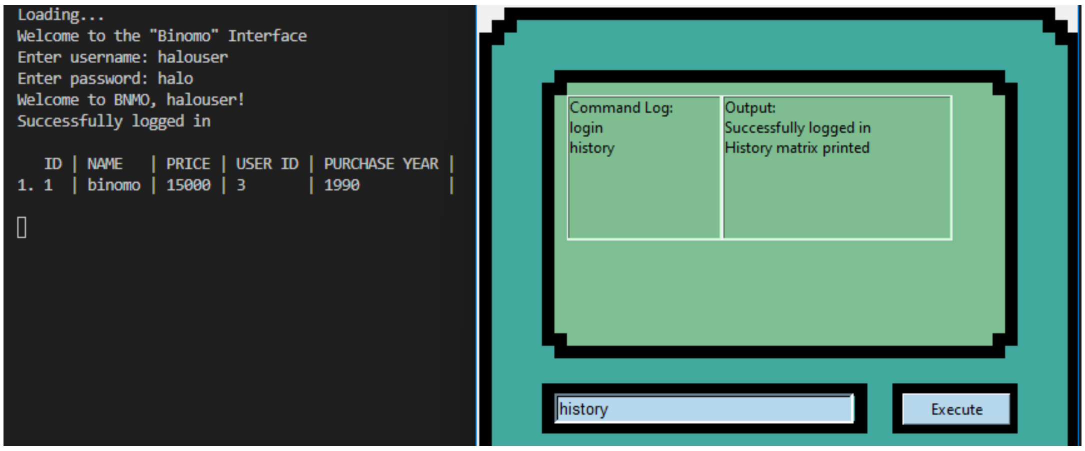
```
   ID | NAME   | PRICE | USER ID | PURCHASE YEAR |
1. 1  | binomo | 15000 | 3       | 1990          |
```

Gambar 9.13.2 Output untuk modul F13

Validasi

Gambar 9.13.3 Validasi untuk modul F13

Implementasi di Main Program



```
Loading...
Welcome to the "Binomo" Interface
Enter username: halouser
Enter password: halo
Welcome to BNMO, halouser!
Successfully logged in

   ID | NAME    | PRICE | USER ID | PURCHASE YEAR |
1. 1  | binomo  | 15000 | 3       | 1990          |
```

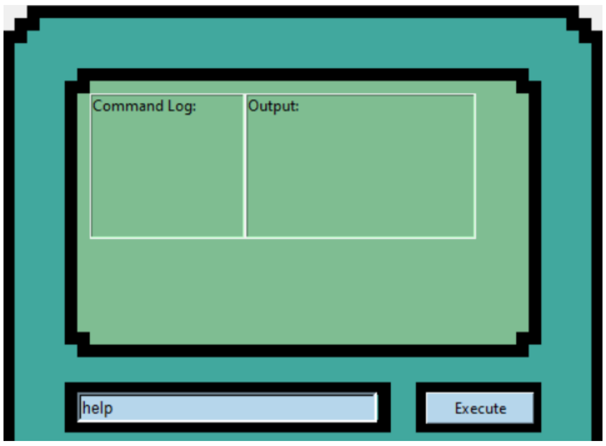
```
Command Log:
login
history
```

```
Output:
Successfully logged in
History matrix printed
```

Gambar 9.13.4 Contoh implementasi modul F13 pada main program

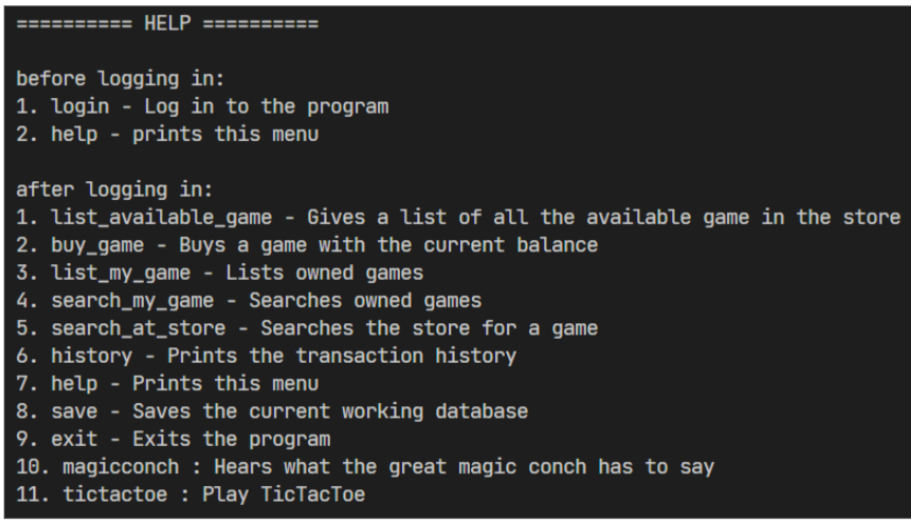
F14 – Help

Input

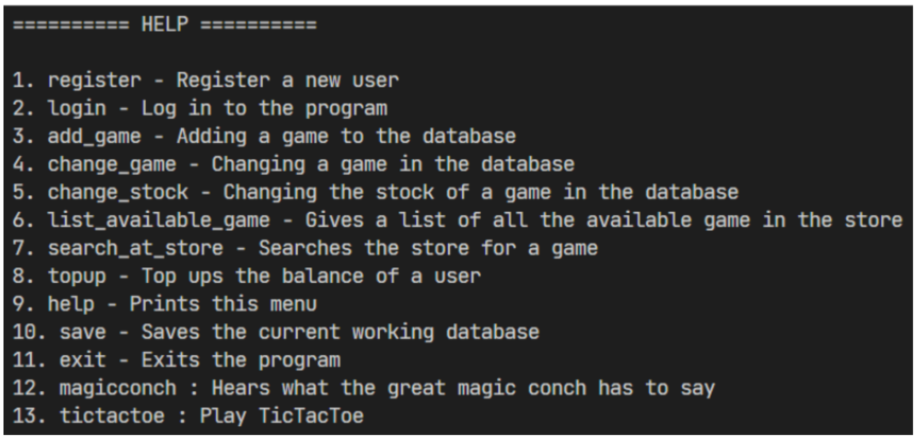


Gambar 9.14.1 Cara memanggil modul F14

Output



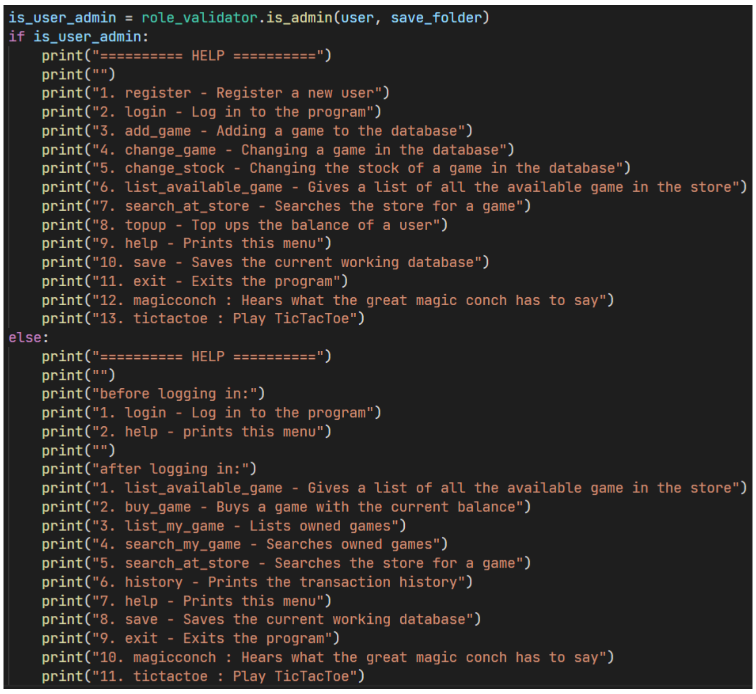
```
========== HELP ==========

before logging in:
1. login - Log in to the program
2. help - prints this menu

after logging in:
1. list_available_game - Gives a list of all the available game in the store
2. buy_game - Buys a game with the current balance
3. list_my_game - Lists owned games
4. search_my_game - Searches owned games
5. search_at_store - Searches the store for a game
6. history - Prints the transaction history
7. help - Prints this menu
8. save - Saves the current working database
9. exit - Exits the program
10. magicconch : Hears what the great magic conch has to say
11. tictactoe : Play TicTacToe
```



```
========== HELP ==========

1. register - Register a new user
2. login - Log in to the program
3. add_game - Adding a game to the database
4. change_game - Changing a game in the database
5. change_stock - Changing the stock of a game in the database
6. list_available_game - Gives a list of all the available game in the store
7. search_at_store - Searches the store for a game
8. topup - Top ups the balance of a user
9. help - Prints this menu
10. save - Saves the current working database
11. exit - Exits the program
12. magicconch : Hears what the great magic conch has to say
13. tictactoe : Play TicTacToe
```

Gambar 9.14.2 Output untuk modul F14

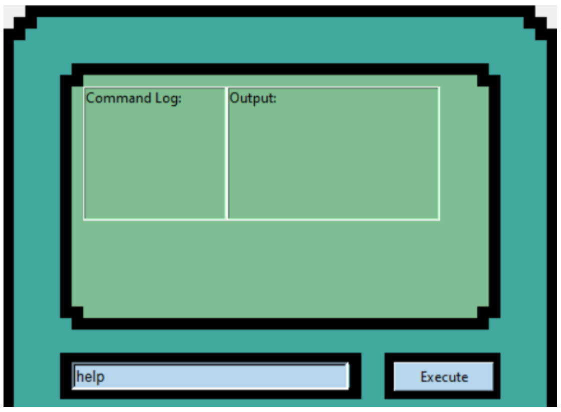
Validasi



```python
is_user_admin = role_validator.is_admin(user, save_folder)
if is_user_admin:
    print("========== HELP ==========")
    print("")
    print("1. register - Register a new user")
    print("2. login - Log in to the program")
    print("3. add_game - Adding a game to the database")
    print("4. change_game - Changing a game in the database")
    print("5. change_stock - Changing the stock of a game in the database")
    print("6. list_available_game - Gives a list of all the available game in the store")
    print("7. search_at_store - Searches the store for a game")
    print("8. topup - Top ups the balance of a user")
    print("9. help - Prints this menu")
    print("10. save - Saves the current working database")
    print("11. exit - Exits the program")
    print("12. magicconch : Hears what the great magic conch has to say")
    print("13. tictactoe : Play TicTacToe")
else:
    print("========== HELP ==========")
    print("")
    print("before logging in:")
    print("1. login - Log in to the program")
    print("2. help - prints this menu")
    print("")
    print("after logging in:")
    print("1. list_available_game - Gives a list of all the available game in the store")
    print("2. buy_game - Buys a game with the current balance")
    print("3. list_my_game - Lists owned games")
    print("4. search_my_game - Searches owned games")
    print("5. search_at_store - Searches the store for a game")
    print("6. history - Prints the transaction history")
    print("7. help - Prints this menu")
    print("8. save - Saves the current working database")
    print("9. exit - Exits the program")
    print("10. magicconch : Hears what the great magic conch has to say")
    print("11. tictactoe : Play TicTacToe")
```

Gambar 9.14.3 Validasi untuk modul F14

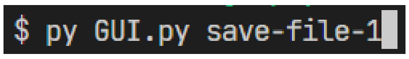
Implementasi di Main Program



Gambar 9.14.4 Contoh implementasi modul F14 pada main program

F15 – Load

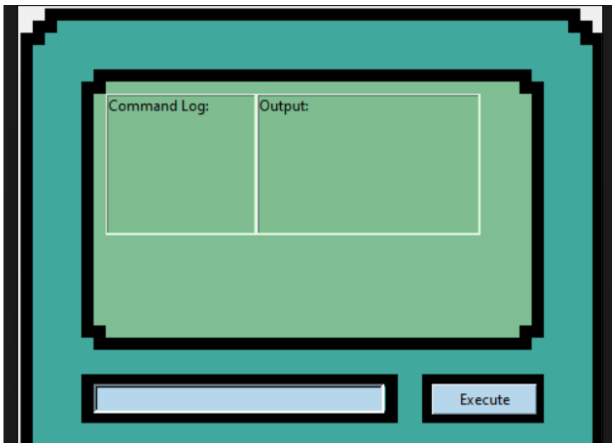
Input



```
$ py GUI.py save-file-1
```

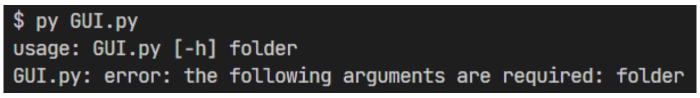
Gambar 9.15.1 Cara memanggil modul F15

Output



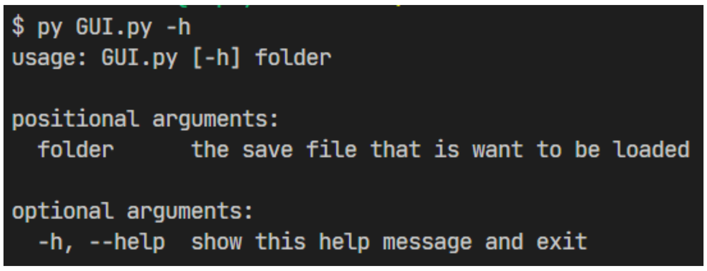
Gambar 9.15.2 Output untuk modul F15

Validasi



```
$ py GUI.py
usage: GUI.py [-h] folder
GUI.py: error: the following arguments are required: folder
```

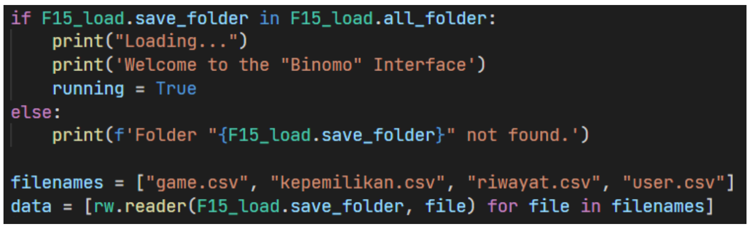
Gambar 9.15.3 Validasi 1 untuk modul F15



```
$ py GUI.py -h
usage: GUI.py [-h] folder

positional arguments:
  folder      the save file that is want to be loaded

optional arguments:
  -h, --help  show this help message and exit
```

Gambar 9.15.4 Validasi 2 untuk modul F15

Implementasi di Main Program

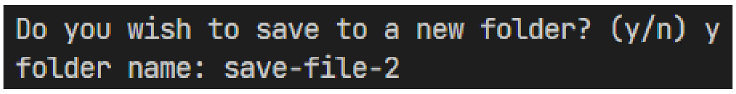


```
if F15_load.save_folder in F15_load.all_folder:
    print("Loading...")
    print('Welcome to the "Binomo" Interface')
    running = True
else:
    print(f'Folder "{F15_load.save_folder}" not found.')

filenames = ["game.csv", "kepemilikan.csv", "riwayat.csv", "user.csv"]
data = [rw.reader(F15_load.save_folder, file) for file in filenames]
```

Gambar 9.15.5 Contoh implementasi modul F15 pada main program

F16 – Save

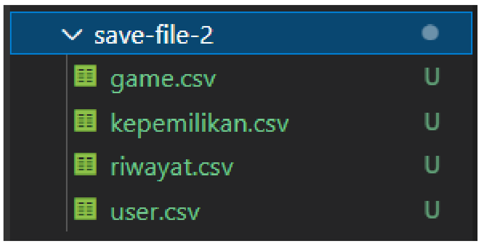
Input



```
Do you wish to save to a new folder? (y/n) y
folder name: save-file-2
```

Gambar 9.16.1 Input untuk modul F16

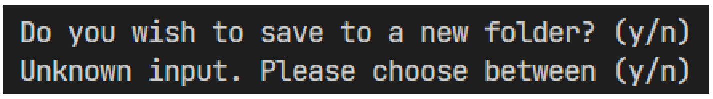
Output



```
∨ save-file-2
    game.csv         U
    kepemilikan.csv  U
    riwayat.csv      U
    user.csv         U
```

Gambar 9.16.2 Output untuk modul F16 berupa folder baru

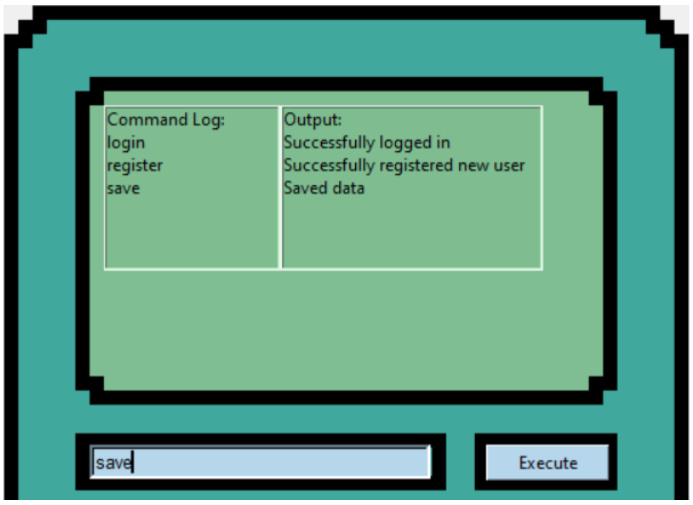
Validasi



```
Do you wish to save to a new folder? (y/n)
Unknown input. Please choose between (y/n)
```

Gambar 9.16.3 Validasi untuk modul F16

Implementasi di Main Program

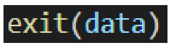


```
Command Log:        Output:
login               Successfully logged in
register            Successfully registered new user
save                Saved data
```

Gambar 9.12.4 Contoh implementasi modul F16 pada main program

F17 – Exit

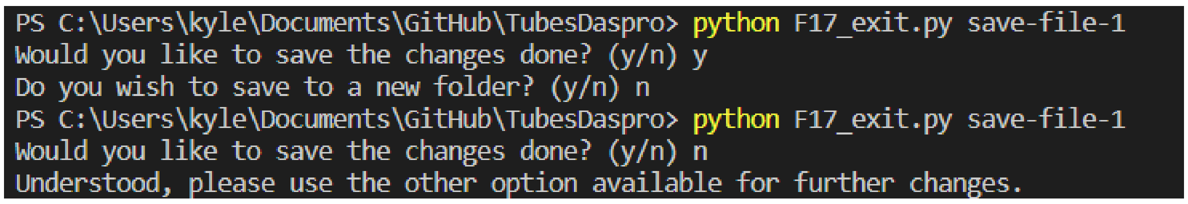
Input



```
exit(data)
```

Gambar 9.17.1 Input untuk modul F17

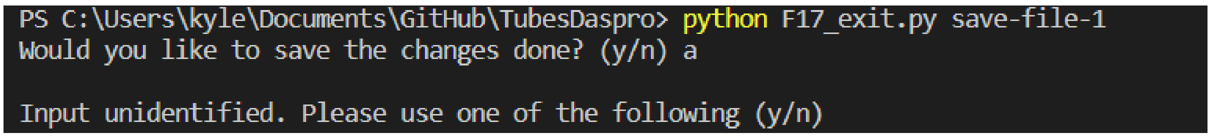
Output



```
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python F17_exit.py save-file-1
Would you like to save the changes done? (y/n) y
Do you wish to save to a new folder? (y/n) n
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python F17_exit.py save-file-1
Would you like to save the changes done? (y/n) n
Understood, please use the other option available for further changes.
```

Gambar 9.17.2 Output untuk modul F17

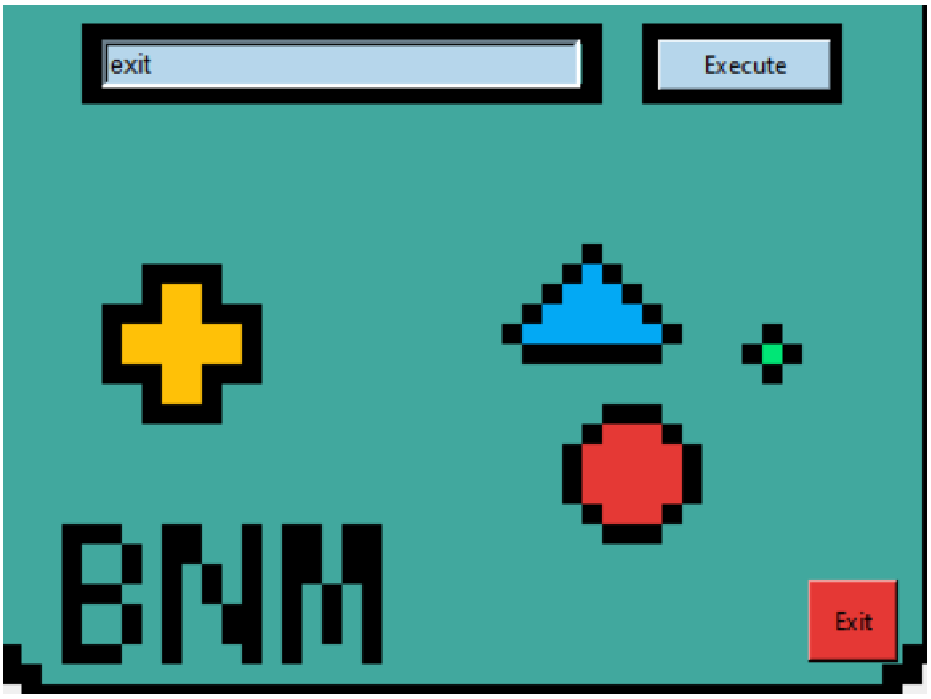
Validasi



```
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python F17_exit.py save-file-1
Would you like to save the changes done? (y/n) a

Input unidentified. Please use one of the following (y/n)
```

Gambar 9.17.3 Validasi untuk modul F17

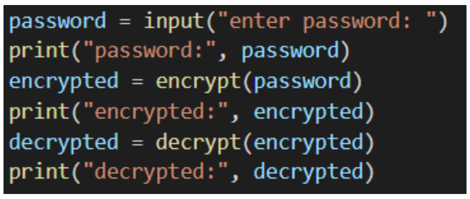
Implementasi di Main Program



Gambar 9.17.4. Implementasi untuk modul F17

B01 – Cipher

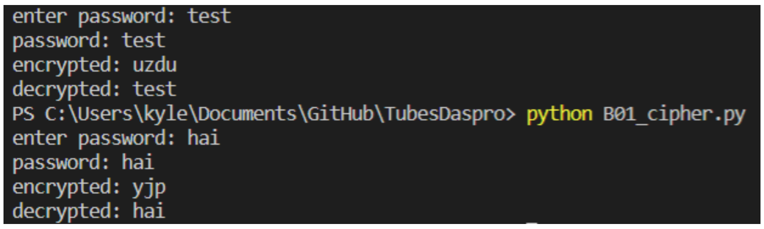
Input



```
password = input("enter password: ")
print("password:", password)
encrypted = encrypt(password)
print("encrypted:", encrypted)
decrypted = decrypt(encrypted)
print("decrypted:", decrypted)
```

Gambar 9.18.1 Cara memanggil fungsi dalam modul B01

Output



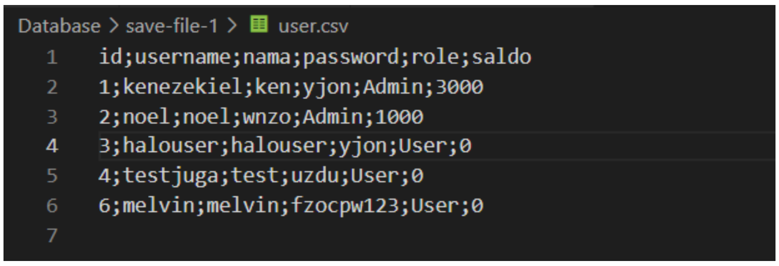
```
enter password: test
password: test
encrypted: uzdu
decrypted: test
PS C:\Users\kyle\Documents\GitHub\TubesDaspro> python B01_cipher.py
enter password: hai
password: hai
encrypted: yjp
decrypted: hai
```

Gambar 9.18.2 Output untuk modul B01

Validasi

Tidak terdapat validasi karena diasumsikan semua password yang masuk telah tervalidasi oleh module F02 - Register

Implementasi di Main Program

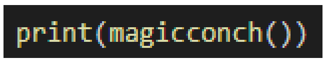


```
Database > save-file-1 > user.csv
id;username;nama;password;role;saldo
1;kenezekiel;ken;yjon;Admin;3000
2;noel;noel;wnzo;Admin;1000
3;halouser;halouser;yjon;User;0
4;testjuga;test;uzdu;User;0
6;melvin;melvin;fzocpw123;User;0
```

Gambar 9.18.3 Hasil implementasi modul B01 pada database main program

B02 – Magic Conch

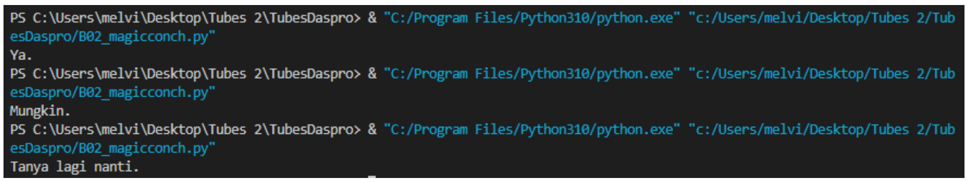
Input



```
print(magicconch())
```

Gambar 9.19.1 Cara memanggil fungsi dalam modul B02

Output



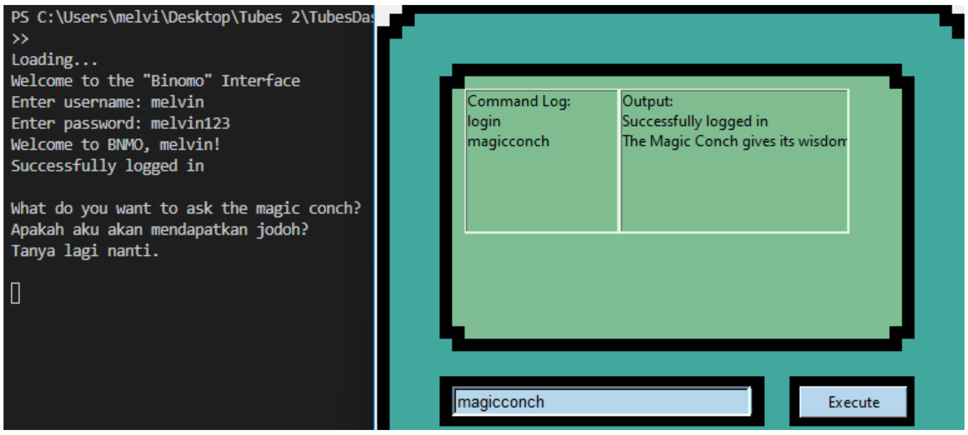
```
PS C:\Users\melvi\Desktop\Tubes 2\TubesDaspro> & "C:/Program Files/Python310/python.exe" "c:/Users/melvi/Desktop/Tubes 2/Tub
esDaspro/B02_magicconch.py"
Ya.
PS C:\Users\melvi\Desktop\Tubes 2\TubesDaspro> & "C:/Program Files/Python310/python.exe" "c:/Users/melvi/Desktop/Tubes 2/Tub
esDaspro/B02_magicconch.py"
Mungkin.
PS C:\Users\melvi\Desktop\Tubes 2\TubesDaspro> & "C:/Program Files/Python310/python.exe" "c:/Users/melvi/Desktop/Tubes 2/Tub
esDaspro/B02_magicconch.py"
Tanya lagi nanti.
```

Gambar 9.19.2 Output untuk modul B02

Validasi

Tidak terdapat validasi karena function tidak menerima input apa pun.

Implementasi di Main Program



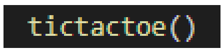
```
PS C:\Users\melvi\Desktop\Tubes 2\TubesDa
>>
Loading...
Welcome to the "Binomo" Interface
Enter username: melvin
Enter password: melvin123
Welcome to BNMO, melvin!
Successfully logged in

What do you want to ask the magic conch?
Apakah aku akan mendapatkan jodoh?
Tanya lagi nanti.
```

```
Command Log:
login
magicconch

Output:
Successfully logged in
The Magic Conch gives its wisdom
```

Gambar 9.19.3 Hasil implementasi modul B02 pada database main program

B03 – Game Tic-Tac-Toe

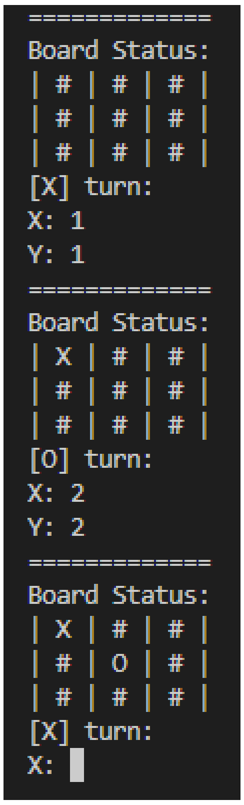
Input



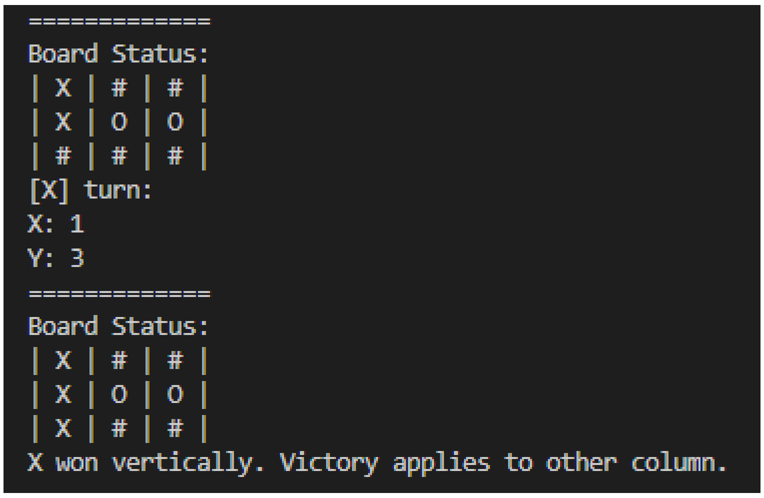
```
tictactoe()
```

Gambar 9.20.1 Cara memanggil fungsi dalam modul B03

Output



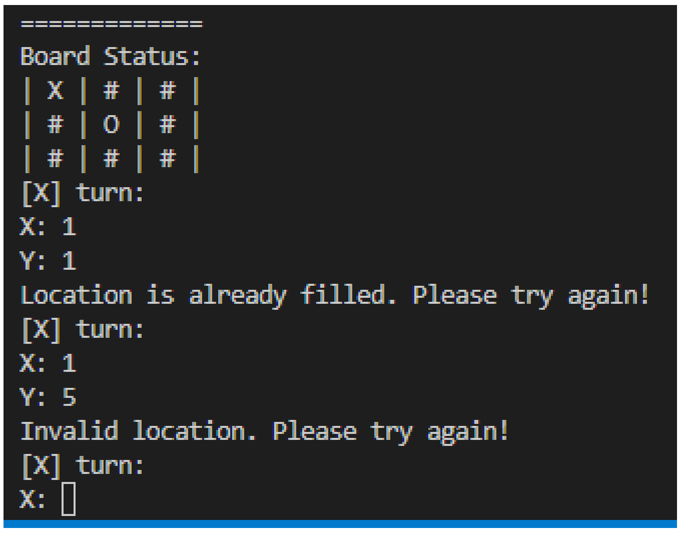
```
=============
Board Status:
| # | # | # |
| # | # | # |
| # | # | # |
[X] turn:
X: 1
Y: 1
=============
Board Status:
| X | # | # |
| # | # | # |
| # | # | # |
[O] turn:
X: 2
Y: 2
=============
Board Status:
| X | # | # |
| # | O | # |
| # | # | # |
[X] turn:
X:
```



```
=============
Board Status:
| X | # | # |
| X | O | O |
| # | # | # |
[X] turn:
X: 1
Y: 3
=============
Board Status:
| X | # | # |
| X | O | O |
| X | # | # |
X won vertically. Victory applies to other column.
```

Gambar 9.20.2 Output modul B03

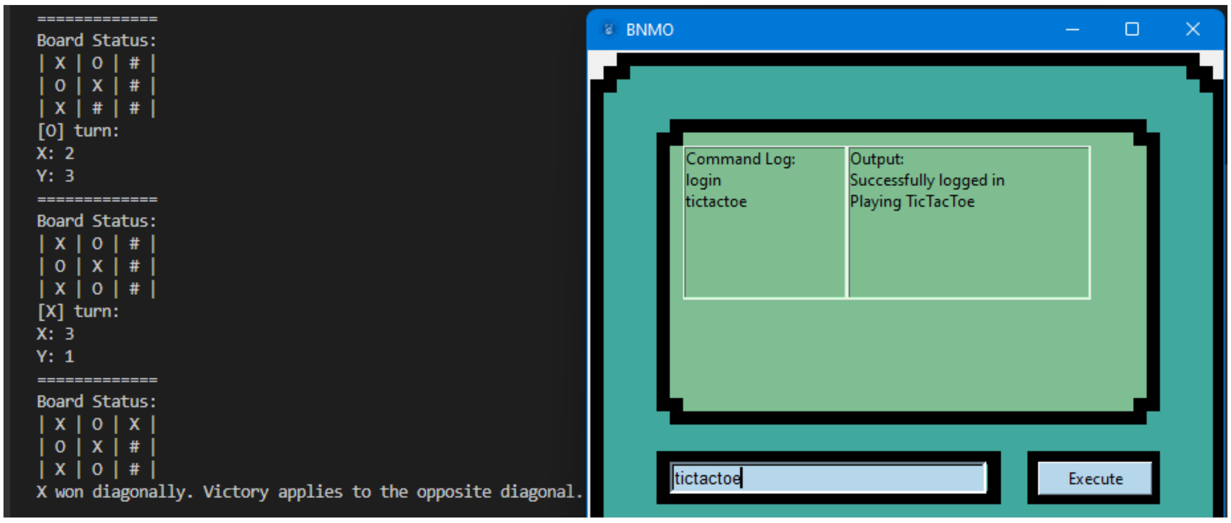
Validasi



```
=============
Board Status:
| X | # | # |
| # | O | # |
| # | # | # |
[X] turn:
X: 1
Y: 1
Location is already filled. Please try again!
[X] turn:
X: 1
Y: 5
Invalid location. Please try again!
[X] turn:
X: 
```

Gambar 9.20.3 Validasi untuk modul B03

Implementasi di Main Program



```
=============
Board Status:
| X | O | # |
| O | X | # |
| X | # | # |
[O] turn:
X: 2
Y: 3
=============
Board Status:
| X | O | # |
| O | X | # |
| X | O | # |
[X] turn:
X: 3
Y: 1
=============
Board Status:
| X | O | X |
| O | X | # |
| X | O | # |
X won diagonally. Victory applies to the opposite diagonal.
```

BNMO

Command Log:
login
tictactoe

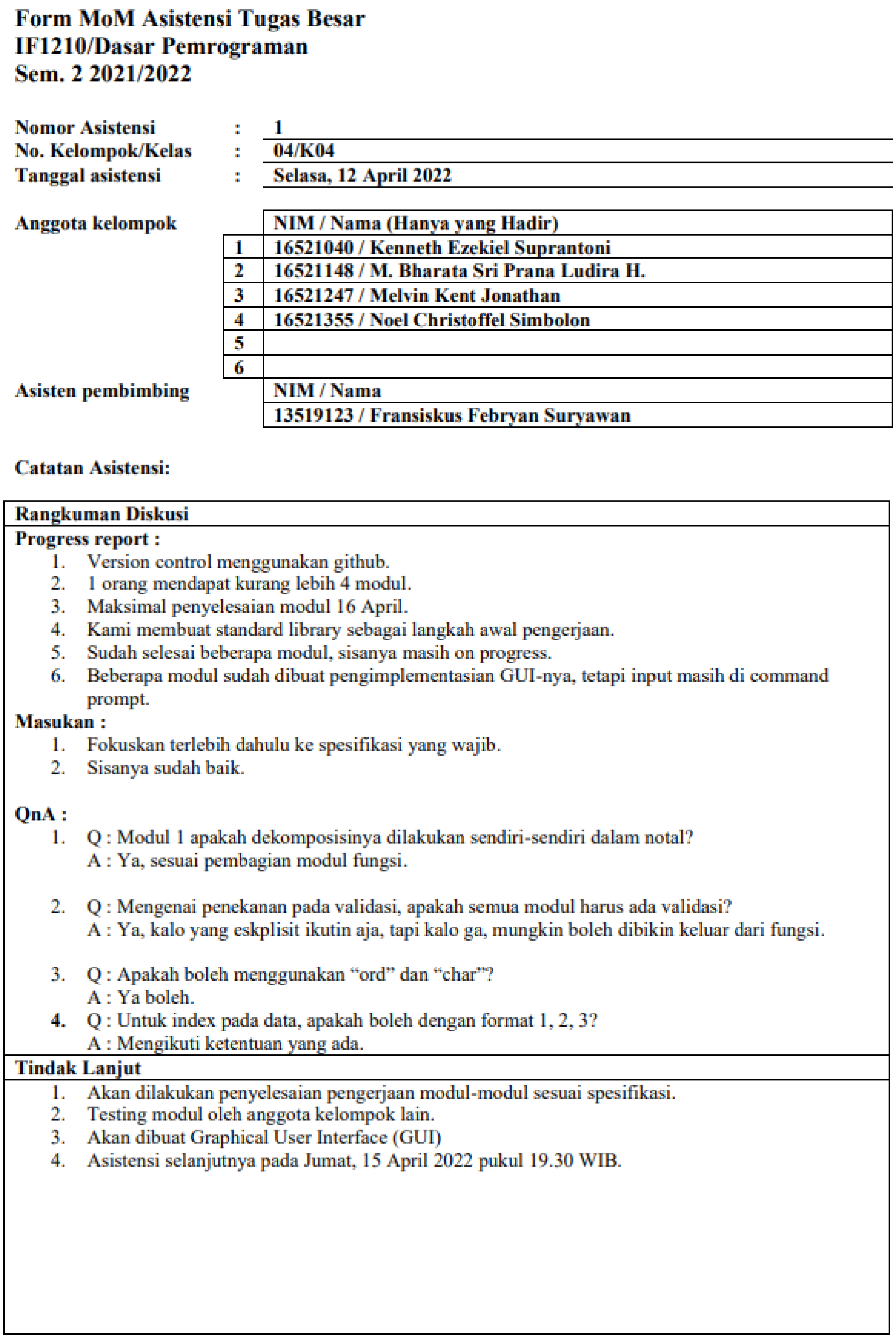
Output:
Successfully logged in
Playing TicTacToe

tictactoe

Gambar 9.20.4 Hasil implementasi modul B03 pada database main program

Lampiran : Hasil Scan Forms Asistensi

Form Asistensi 1



Form MoM Asistensi Tugas Besar
IF1210/Dasar Pemrograman
Sem. 2 2021/2022

Nomor Asistensi	:	**1**
No. Kelompok/Kelas	:	**04/K04**
Tanggal asistensi	:	**Selasa, 12 April 2022**

Anggota kelompok		**NIM / Nama (Hanya yang Hadir)**
	1	**16521040 / Kenneth Ezekiel Suprantoni**
	2	**16521148 / M. Bharata Sri Prana Ludira H.**
	3	**16521247 / Melvin Kent Jonathan**
	4	**16521355 / Noel Christoffel Simbolon**
	5	
	6	
Asisten pembimbing		**NIM / Nama**
		13519123 / Fransiskus Febryan Suryawan

Catatan Asistensi:

Rangkuman Diskusi

Progress report :
1. Version control menggunakan github.
2. 1 orang mendapat kurang lebih 4 modul.
3. Maksimal penyelesaian modul 16 April.
4. Kami membuat standard library sebagai langkah awal pengerjaan.
5. Sudah selesai beberapa modul, sisanya masih on progress.
6. Beberapa modul sudah dibuat pengimplementasian GUI-nya, tetapi input masih di command prompt.

Masukan :
1. Fokuskan terlebih dahulu ke spesifikasi yang wajib.
2. Sisanya sudah baik.

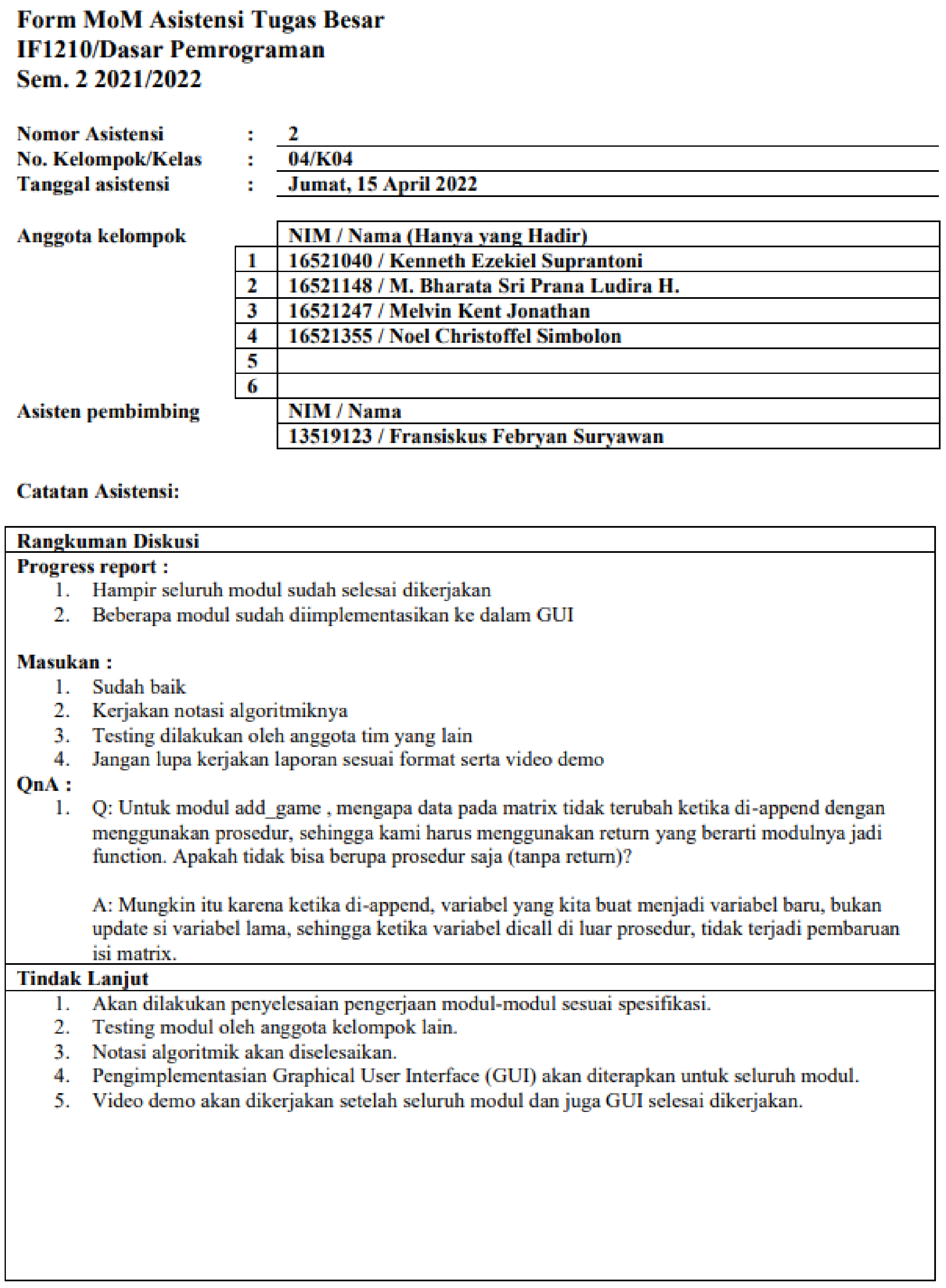
QnA :
1. Q : Modul 1 apakah dekomposisinya dilakukan sendiri-sendiri dalam notal?
   A : Ya, sesuai pembagian modul fungsi.

2. Q : Mengenai penekanan pada validasi, apakah semua modul harus ada validasi?
   A : Ya, kalo yang eskplisit ikutin aja, tapi kalo ga, mungkin boleh dibikin keluar dari fungsi.

3. Q : Apakah boleh menggunakan "ord" dan "char"?
   A : Ya boleh.
4. Q : Untuk index pada data, apakah boleh dengan format 1, 2, 3?
   A : Mengikuti ketentuan yang ada.

Tindak Lanjut
1. Akan dilakukan penyelesaian pengerjaan modul-modul sesuai spesifikasi.
2. Testing modul oleh anggota kelompok lain.
3. Akan dibuat Graphical User Interface (GUI)
4. Asistensi selanjutnya pada Jumat, 15 April 2022 pukul 19.30 WIB.

Form Asistensi 2



Form MoM Asistensi Tugas Besar
IF1210/Dasar Pemrograman
Sem. 2 2021/2022

Nomor Asistensi	:	**2**
No. Kelompok/Kelas	:	**04/K04**
Tanggal asistensi	:	**Jumat, 15 April 2022**

Anggota kelompok		**NIM / Nama (Hanya yang Hadir)**
	1	**16521040 / Kenneth Ezekiel Suprantoni**
	2	**16521148 / M. Bharata Sri Prana Ludira H.**
	3	**16521247 / Melvin Kent Jonathan**
	4	**16521355 / Noel Christoffel Simbolon**
	5	
	6	
Asisten pembimbing		**NIM / Nama**
		13519123 / Fransiskus Febryan Suryawan

Catatan Asistensi:

Rangkuman Diskusi

Progress report :

1. Hampir seluruh modul sudah selesai dikerjakan
2. Beberapa modul sudah diimplementasikan ke dalam GUI

Masukan :

1. Sudah baik
2. Kerjakan notasi algoritmiknya
3. Testing dilakukan oleh anggota tim yang lain
4. Jangan lupa kerjakan laporan sesuai format serta video demo

QnA :

1. Q: Untuk modul add_game , mengapa data pada matrix tidak terubah ketika di-append dengan menggunakan prosedur, sehingga kami harus menggunakan return yang berarti modulnya jadi function. Apakah tidak bisa berupa prosedur saja (tanpa return)?

   A: Mungkin itu karena ketika di-append, variabel yang kita buat menjadi variabel baru, bukan update si variabel lama, sehingga ketika variabel dicall di luar prosedur, tidak terjadi pembaruan isi matrix.

Tindak Lanjut

1. Akan dilakukan penyelesaian pengerjaan modul-modul sesuai spesifikasi.
2. Testing modul oleh anggota kelompok lain.
3. Notasi algoritmik akan diselesaikan.
4. Pengimplementasian Graphical User Interface (GUI) akan diterapkan untuk seluruh modul.
5. Video demo akan dikerjakan setelah seluruh modul dan juga GUI selesai dikerjakan.