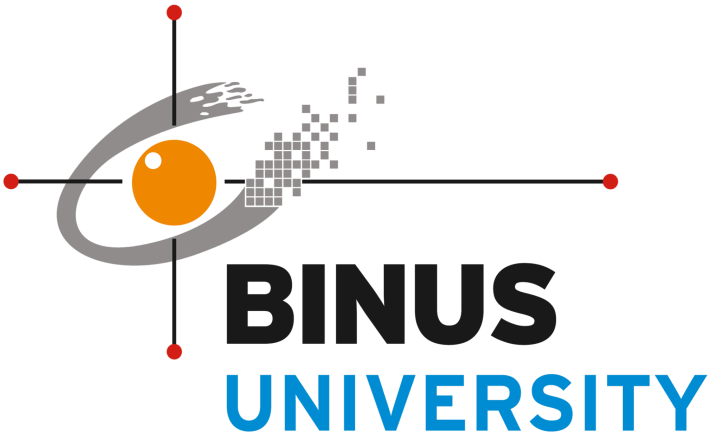
PEMBUATAN APLIKASI KASIR DAN MANAJEMEN WAREHOUSE DENGAN METODE HASHING

LAPORAN PROYEK AKHIR

MATA KULIAH COMP63622004 – DATA STRUCTURES

KELAS LB-20



Oleh :

2702355302 - Kevin Joseph Handoyo
2702350106 - Muhammad Ibraahiim Putra Wahana

Semester Genap 2023/2024

LEMBAR PERSETUJUAN PROYEK AKHIR

PEMBUATAN APLIKASI KASIR DAN MANAJEMEN WAREHOUSE DENGAN METODE HASHING

MATA KULIAH COMP6362004 – DATA STRUCTURES

KELAS LB-20

Semester Genap 2023/2024

Laporan akhir proyek ini adalah benar karya kami :

Kevin Joseph Handoyo	**Muhammad Ibraahiim Putra Wahana**
2702355302	**2702350106**

Malang ,. -…… ….

Chasandra Puspitasari, S.Kom., M.Cs.

DAFTAR ISI

BAB I

A. Background

Pada project kali ini, kami sepakat untuk menggunakan Metode **Hashing** untuk pembuatan dan pengembangan program aplikasi **Kasir dan Manajemen Warehouse**. Alasan kami memilih metode hashing dikarenakan hashing memiliki kelebihan utama yakni kecepatannya dalam mengakses sebuah element, oleh karena itu proses seperti searching dapat dilakukan dengan cepat. Metode ini juga sudah mencakup dan memadai fitur – fitur yang dibutuhkan untuk mengembangkan program sistem kasir ini. Ditambah lagi dengan stock barang yang terkadang jumlah nya ratusan dengan kode pada setiap produk berbeda-beda, metode hashing ini dapat mengatasi collision dengan menggunakana collision single linked-list.

Program terbagi menjadi 2 sistem utama yaitu fungsi sebagai **Kasir & Manajemen** dan setiap fungsi utama terbagi menjadi beberapa fungsi :

1. **Kasir**
    A. Purchase Product
    B. Cashflow
    C. Hash-Table View
    D. Check Expiration
2. **Manajemen**
    A. Add Item
    B. Add Stock
    C. Delete Stock
    D. Hash-Table View
    E. Check Expiration
    F. Update Data
        a. Update Price
        b. Update Expiry Date

Program ini juga menggunakan sistem **File Processing** untuk membaca daftar barang yang tersimpan dalam file database txt, serta melakukan update data pada file database.txt dan cashflow.txt jika user melakukan interaksi pada menu purchase, cashflow, hash-Table view, dan check expiration pada fungsi kasir serta menu add item, add stock, delete stock, hash-table view , check expiration dan update data pada fungsi manajemen warehouse. Pada menu pertama dalam fungsi kasir yaitu Purchase, menu ini

akan mengurangi barang atau dapat menghapus stock pada database sesuai dengan input yang diberikan oleh user. Kedua, menu Cashflow, berfungsi untuk melacak hasil transaksi yang telah dilakukan user pada menu purchase. Ketiga, menu hash-table view yang berfungsi menampilkan daftar produk yang terdapat dalam bentuk Hash-table. Terakhir pada fungsi kasir yaitu terdapat menu check expiration, menu ini sesuai namanya berfungsi untuk mengecek apakah barang yang ingin dicek sudah melewati masa expired atau belum. Selanjutnya yang pertama untuk fungsi kasir yaitu menu Add item, menu ini berfungsi apabila user ingin menambahkan produk baru dimana nantinya produk ini akan menghasilkan ID baru serta hash value untuk dimasukkan ke dalam Hash-table. Kedua, menu Add stock, menu ini berfungsi untuk menambahkan stock barang yang tersimpan pada database. Ketiga, menu Delete stock, sebagaimana namanya menu ini berfungsi untuk menghapus stock atau bahkan bisa menghapus barang yang terdapat pada file database. Menu keempat dan kelima memiliki fungsi yang sama sebagaimana fungsi menu Hash-table view dan Check expiration yang terdapat pada fungsi kasir. Terakhir yaitu terdapat menu Update data dimana user dapat melakukan update pada harga dan expiry date bada barang yang terdapat pada database.

BAB II

B. Literature Review

- **Theory of Related Data Structure**

Hashing adalah teknik penting dalam ilmu komputer yang digunakan untuk menyimpan dan mengambil data secara efisien. Teknik ini bekerja dengan mengubah data dari berbagai ukuran menjadi nilai tetap yang disebut kode hash, menggunakan fungsi hash. Penelitian telah mengeksplorasi berbagai aspek dari hashing, termasuk cara-cara melakukan hashing pada data terstruktur, hashing yang peka terhadap lokasi, hashing data dalam klasifikasi objek visual, dan hashing dalam pencarian gambar berskala besar.

Hash *function* adalah rumus matematika yang diterapkan pada string untuk menghasilkan kunci atau nilai yang dapat digunakan sebagai index dalam penempatan elemen ke hash table. Metode-metode seperti *mid-square, division, dan folding* adalah beberapa metode yang dapat digunakan untuk menjalankan *hash function*. Dengan menggunakan operator modulus, *division* membagi string dengan menggunakan operator modulus. Setiap jumlah ASCII dari setiap karakter akan dijumlahkan dan di modulo dengan ukuran table(*table size*)/

String yang dimasukkan ke dalam hash function dapat menghasilkan kunci yang sama. Dalam hal ini, elemen-elemen yang memiliki kunci yang sama akan bertabrakan. Istilah untuk kejadian ini adalah benturan. Baik Chaining maupun Linear Probing adalah beberapa metode untuk menangani collision. Penanganan Linear Probing melibatkan menemukan slot kosong, atau slot tambahan, dan kemudian menempatkan elemen pada slot tersebut. Meskipun demikian, selama chaining, setiap index dalam daftar Hash berbentuk daftar terkait. Akibatnya, apabila terjadi *collision*, elemen yang sudah berbentuk node akan *dipush* menjadi tail ke daftar terkait dari index yang sesuai dengan *key element*.

- **General**

Program sistem kasir dan manajemen warehouse yang kami buat akan menggunakan metode hashing, dimana program kami akan membaca daftar barang yang terdapat pada file database.txt lalu dimasukkan ke dalam hash-table. Dalam proses memasukkan setiap barang ke hash-table, kode dari barang tersebut akan dimasukkan ke dalam hash function untuk memperoleh key. Pada program ini, hash function yang kami gunakan adalah metode division. Dimana setiap kode dari barang berbentuk string setiap huruf dari ID (ASCII) akan dijumlahkan lalu akan dimodulo dengan table size. ID yang kami dapatkan bisa didapatkan dapat didapatkan dengan 2 rumus yaitu rumus pertama dengan mengambil 2 huruf pertama

dan rumus yang kedua yaitu dengan mengambil 1 kata dari kata-kata yang ada serta menggabungkannya dengan 2 angka pada bagian terakhir untuk kedua rumus tadi. Jika barang yang dimasukkan terdapat 2 kata, maka akan diambil huruf pertama pada setiap kata. Apabila barang hanya memiliki 1 kata maka akan diambil 2 huruf pertama pada kata tersebut. Lalu untuk mendapatkan 2 angka terakhir bisa didapatkan dengan menghitung panjang nama produk lalu ditambahkan dengan 20.

Apabila terjadi collision pada program ini, metode chaining akan digunakan sebagai metode penanganannya. Chaining adalah teknik yang menempatkan semua elemen yang memiliki hash key yang sama ke dalam struktur data terpisah yang disebut linked list. Setelah key didapatkan dari hash function, program akan memeriksa apakah terdapat node di index tersebut dalam hash table. Jika tidak ada, node baru yang berisikan produk akan diletakkan pada linked list sesuai index key-nya. Node baru ini akan diset menjadi head sekaligus tail dari linked list tersebut, mirip seperti metode push pada single linked list. Namun, jika sudah ada node di index tersebut, artinya terjadi collision. Dalam kasus ini, pointer next dari node terakhir pada linked list tersebut akan menunjuk ke new node (node baru) yang berisikan produk baru tersebut. Kemudian, new node ini akan diset menjadi tail baru dari linked list. Dengan demikian, semua produk yang memiliki hash key yang sama akan diorganisir dalam linked list.

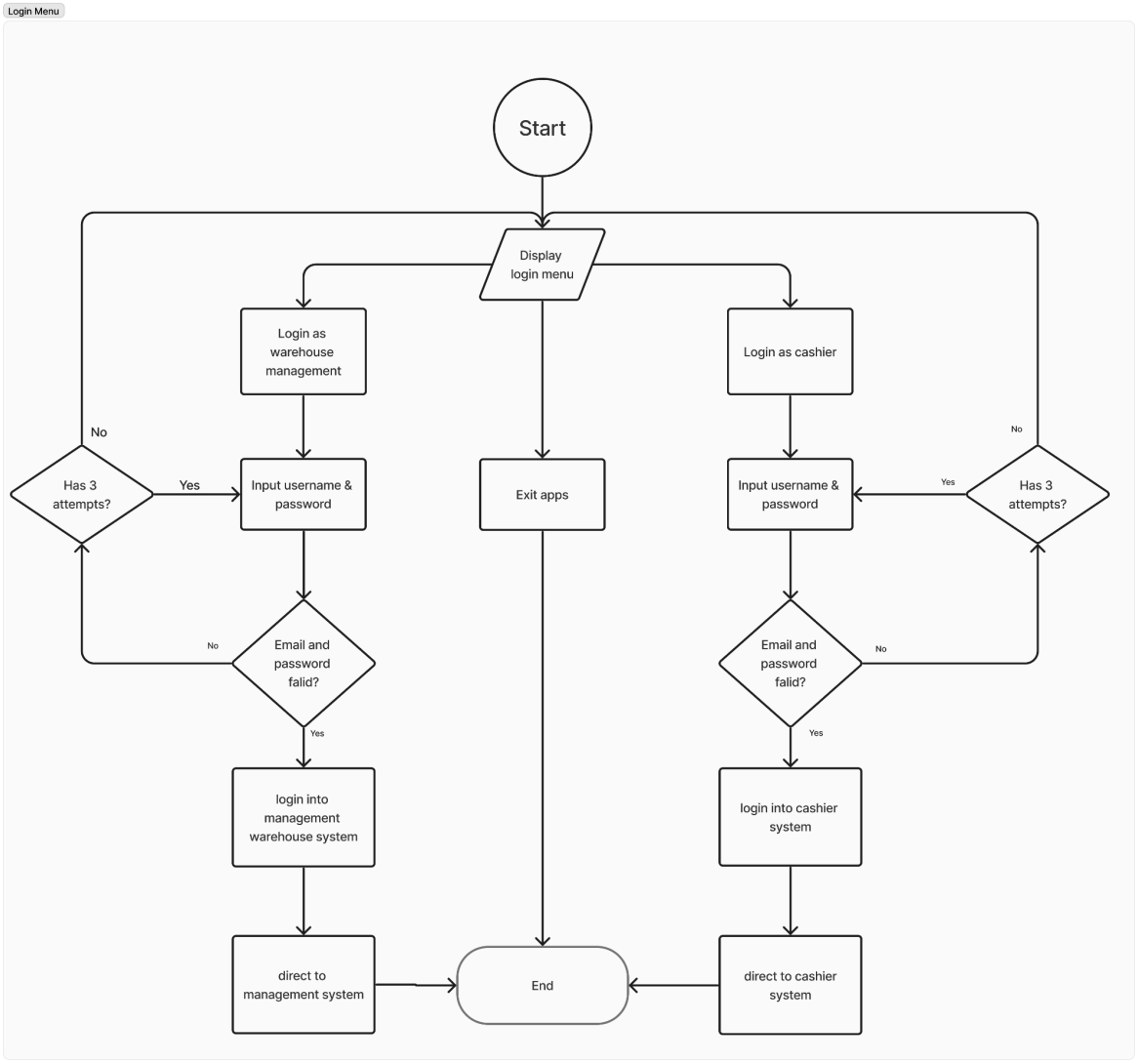
Sesuai dengan teori yang menjelaskan keunggulan metode hashing alasan kami menggunakan metode ini yaitu akses yang lebih cepat dalam mengakses suatu elemen. Proses pencarian dalam program yang menggunakan hashing dapat dikatakan cepat, karena langsung menunjuk ke indeks yang ingin dicari. Dalam program ini tidak perlu memeriksa indeks satu per satu, tetapi langsung mengakses indeks yang ingin dituju dan akan menjalankan proses traversal pada linked list.

BAB III

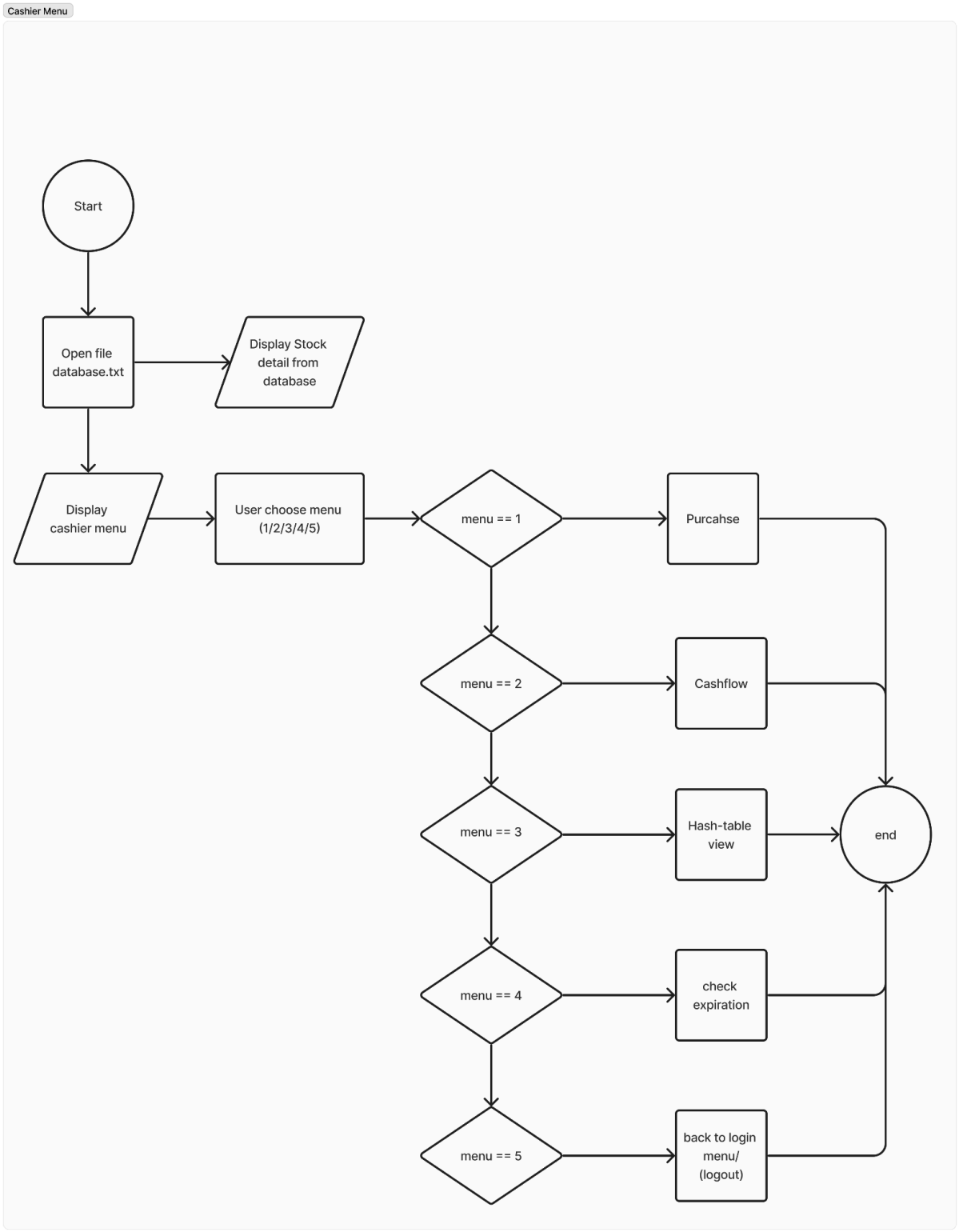
3. Program Definition

1.Flowchart

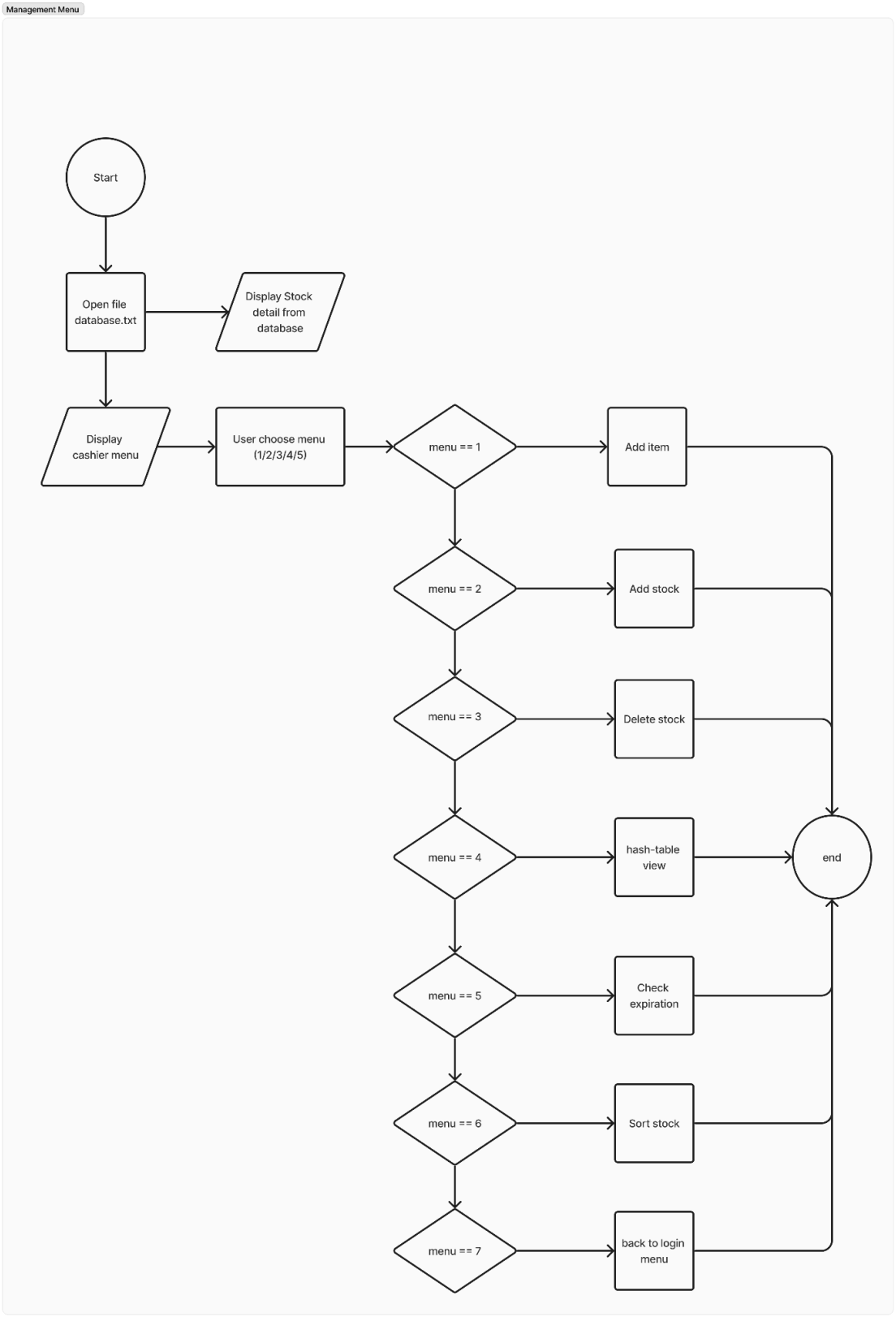
- Login menu



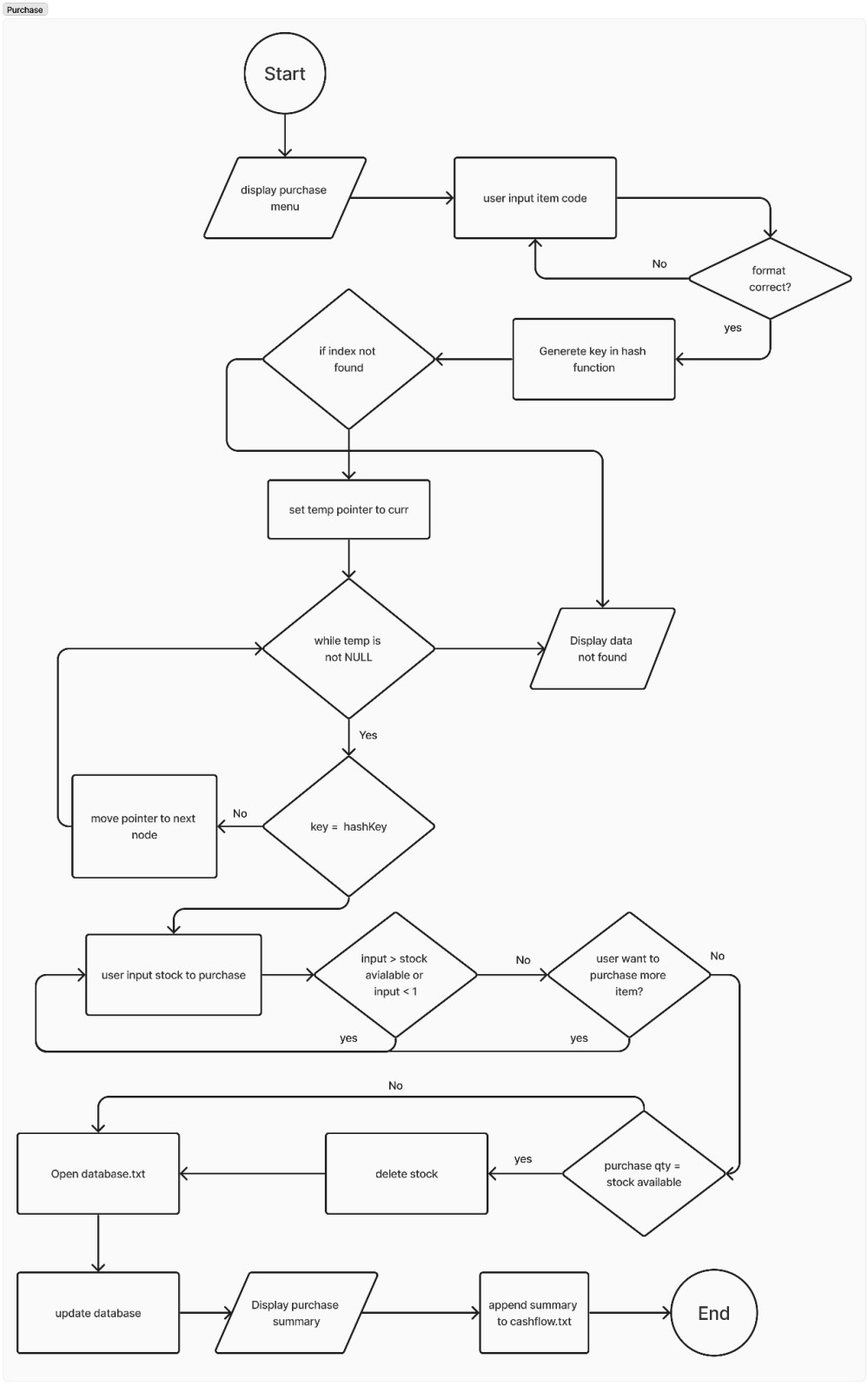
- Cashier menu



- Management menu

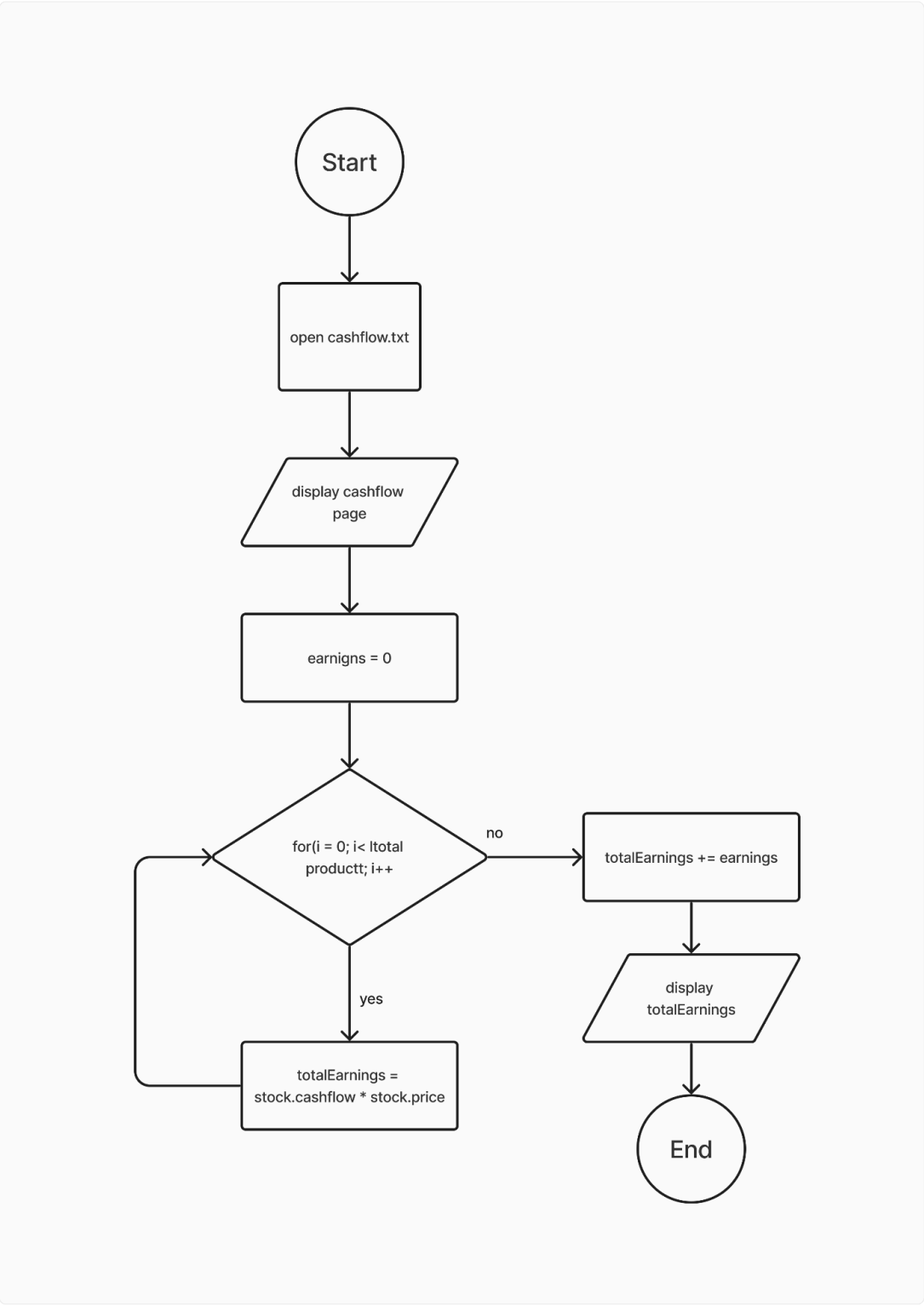


- Purchase

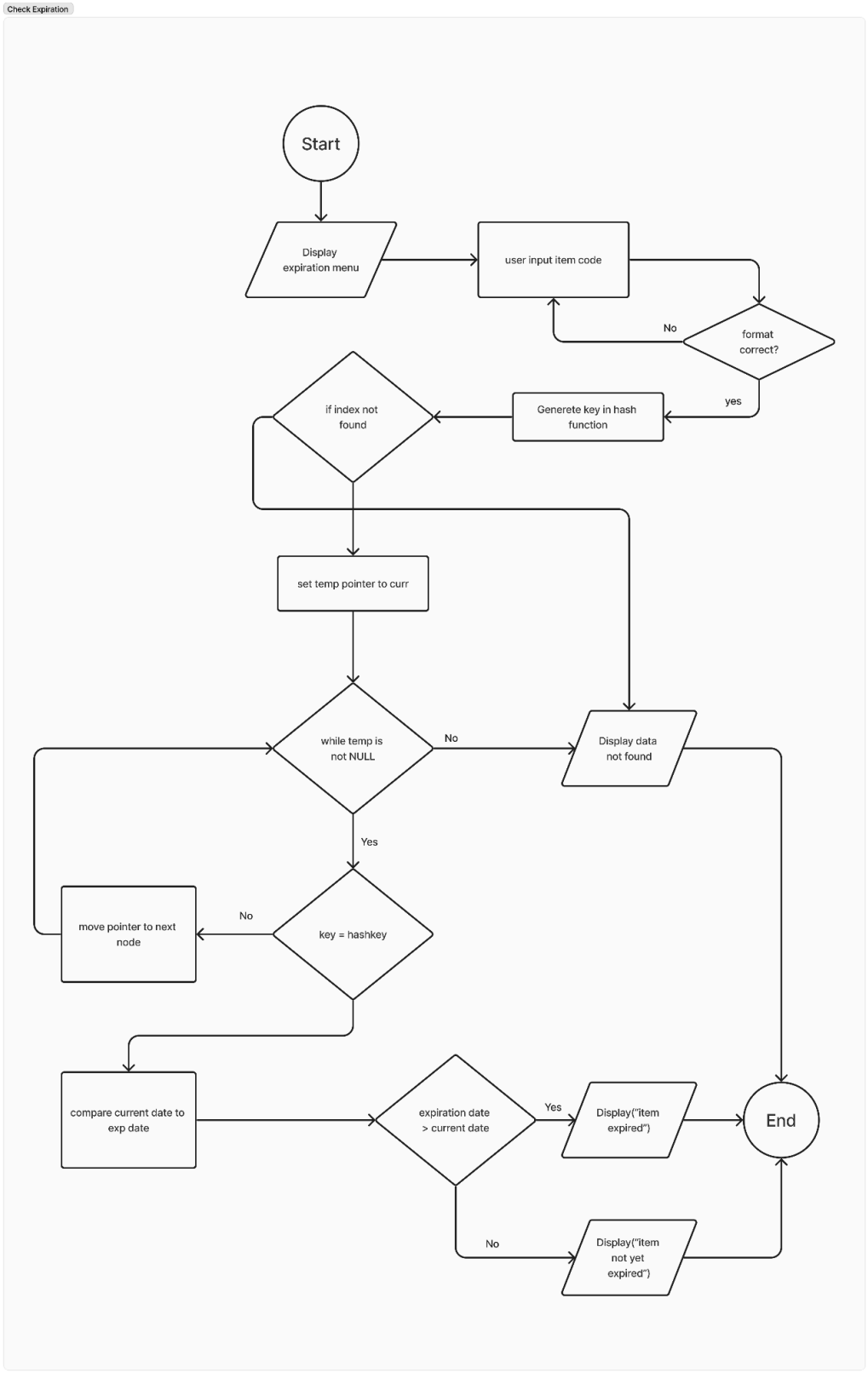

Purchase
Start
display purchase menu
user input item code
format correct?
No
yes
Generete key in hash function
if index not found
set temp pointer to curr
while temp is not NULL
Display data not found
Yes
key = hashKey
No
move pointer to next node
user input stock to purchase
input > stock avialable or input < 1
No
user want to purchase more item?
No
yes
yes
No
purchase qty = stock available
yes
delete stock
Open database.txt
update database
Display purchase summary
append summary to cashflow.txt
End

- Cashflow

cashflow

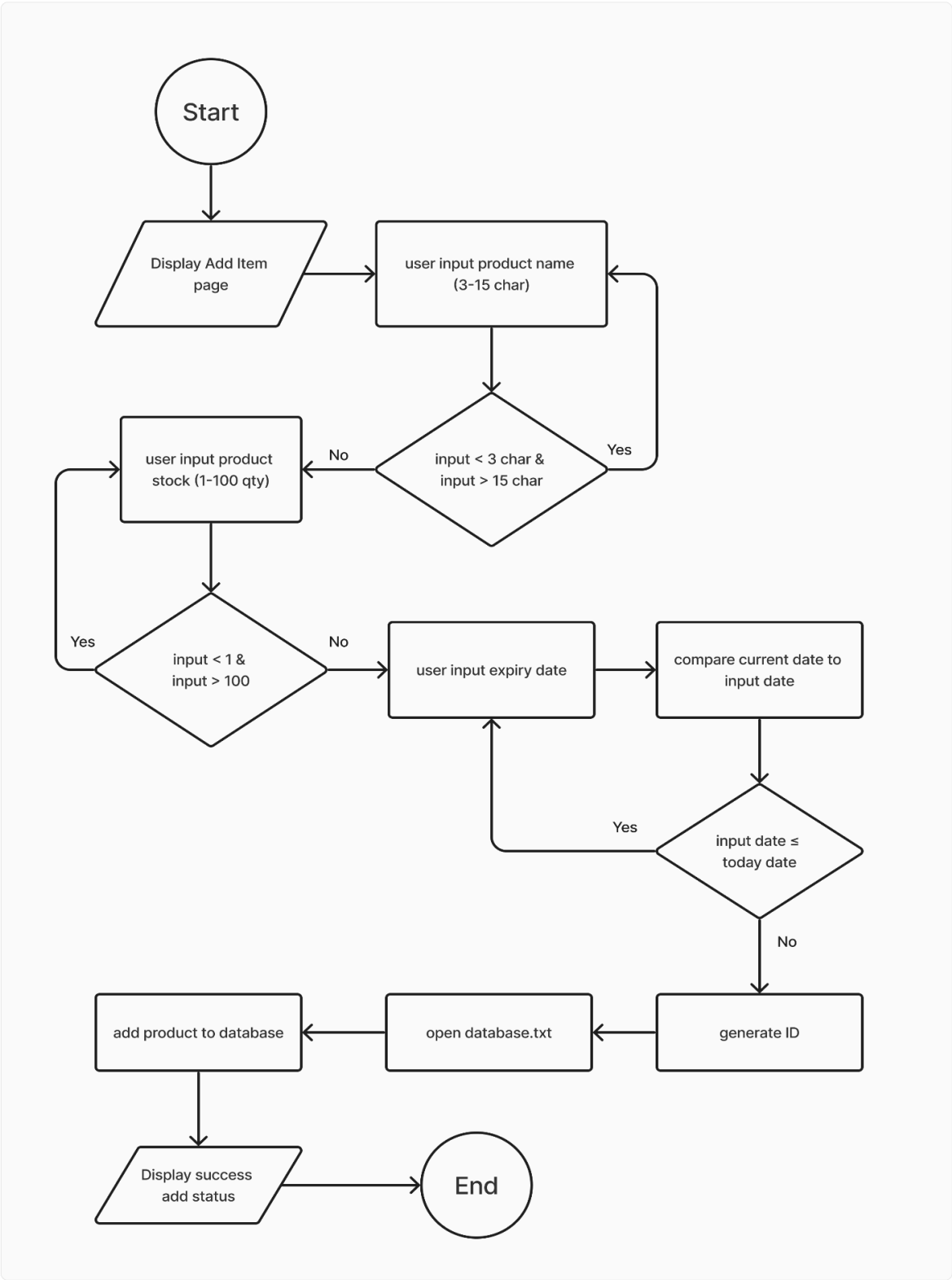


- Check Expiration



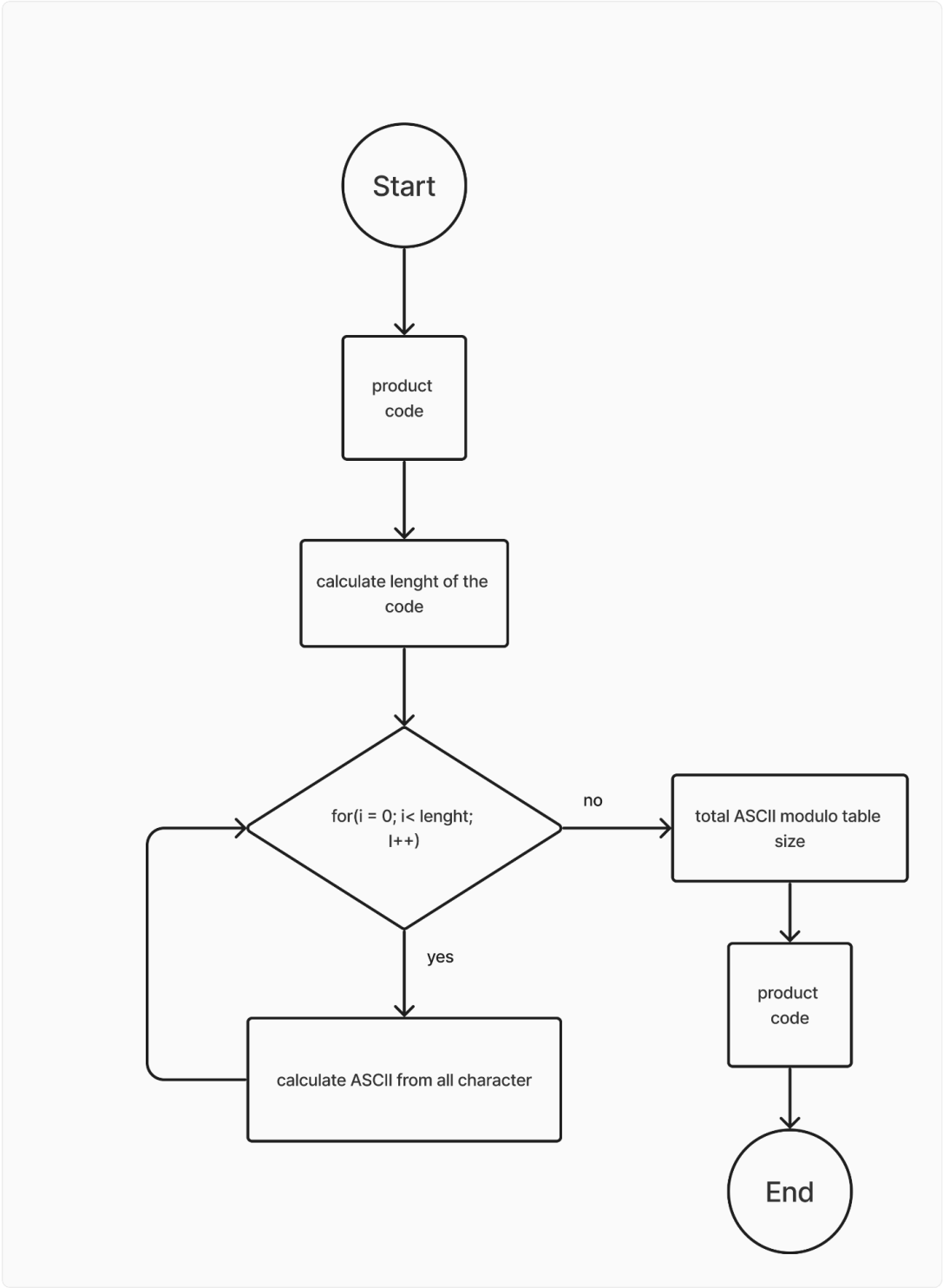
- Add item

add item

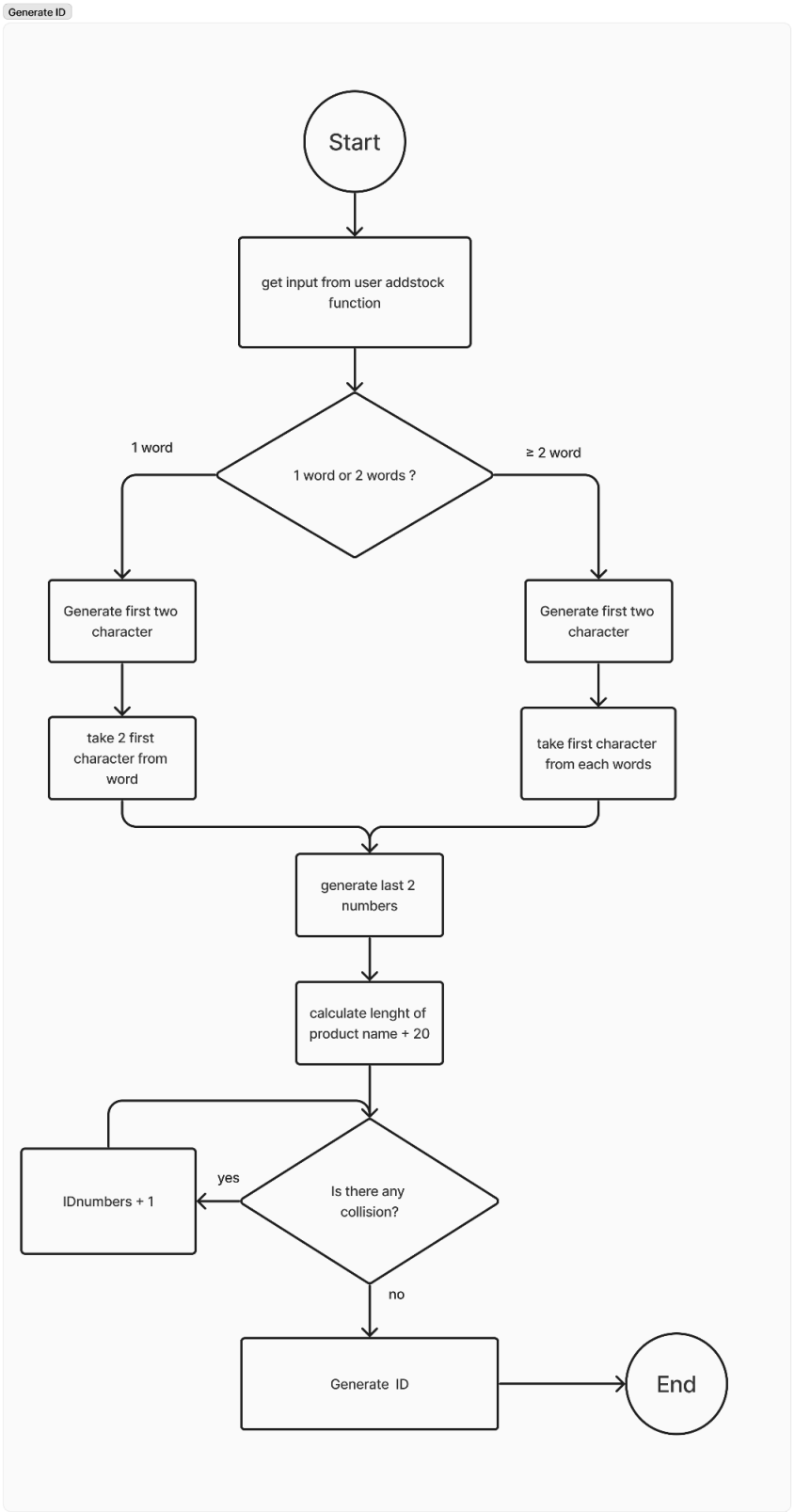


- Hash function

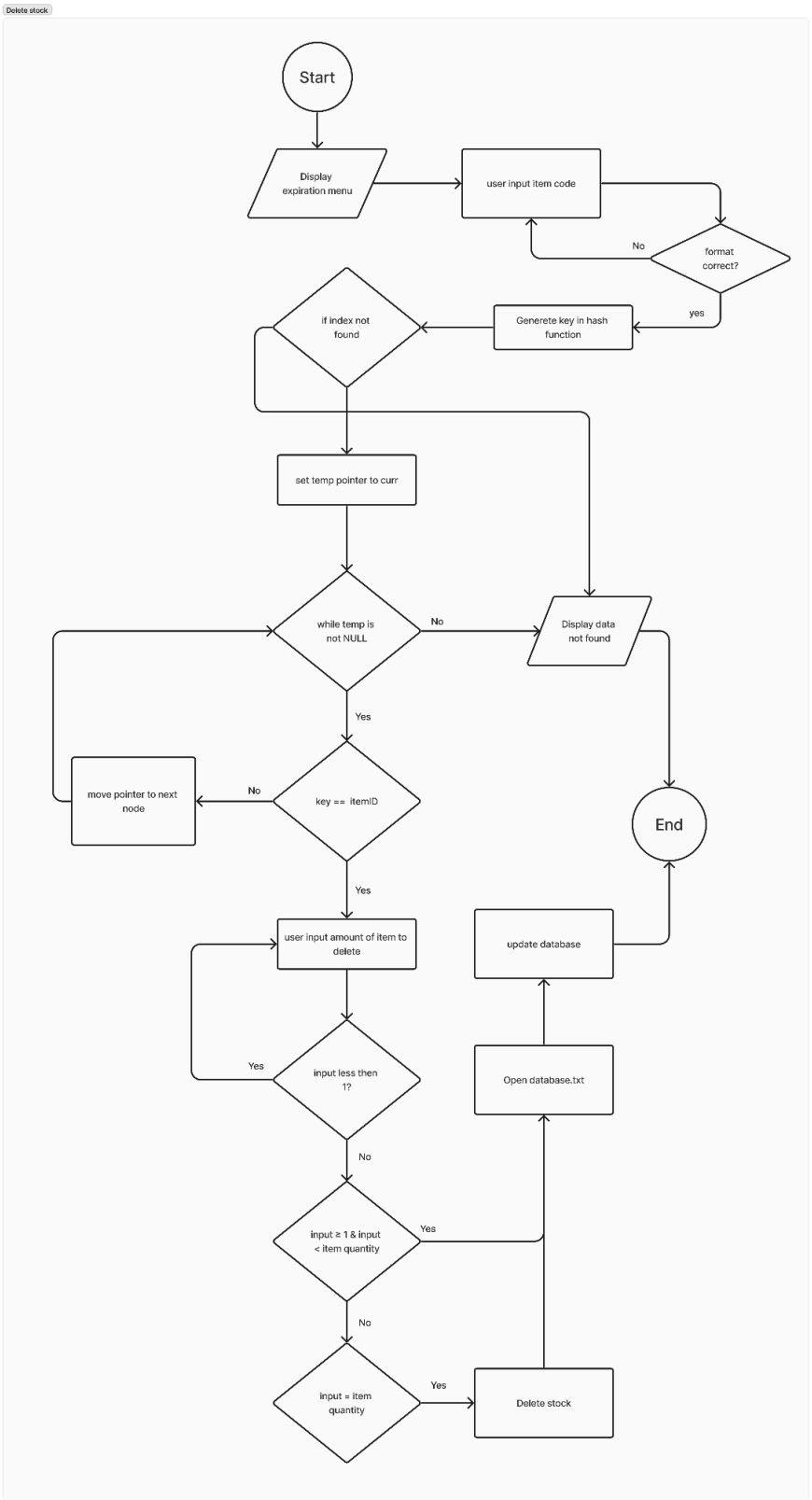
Hash function



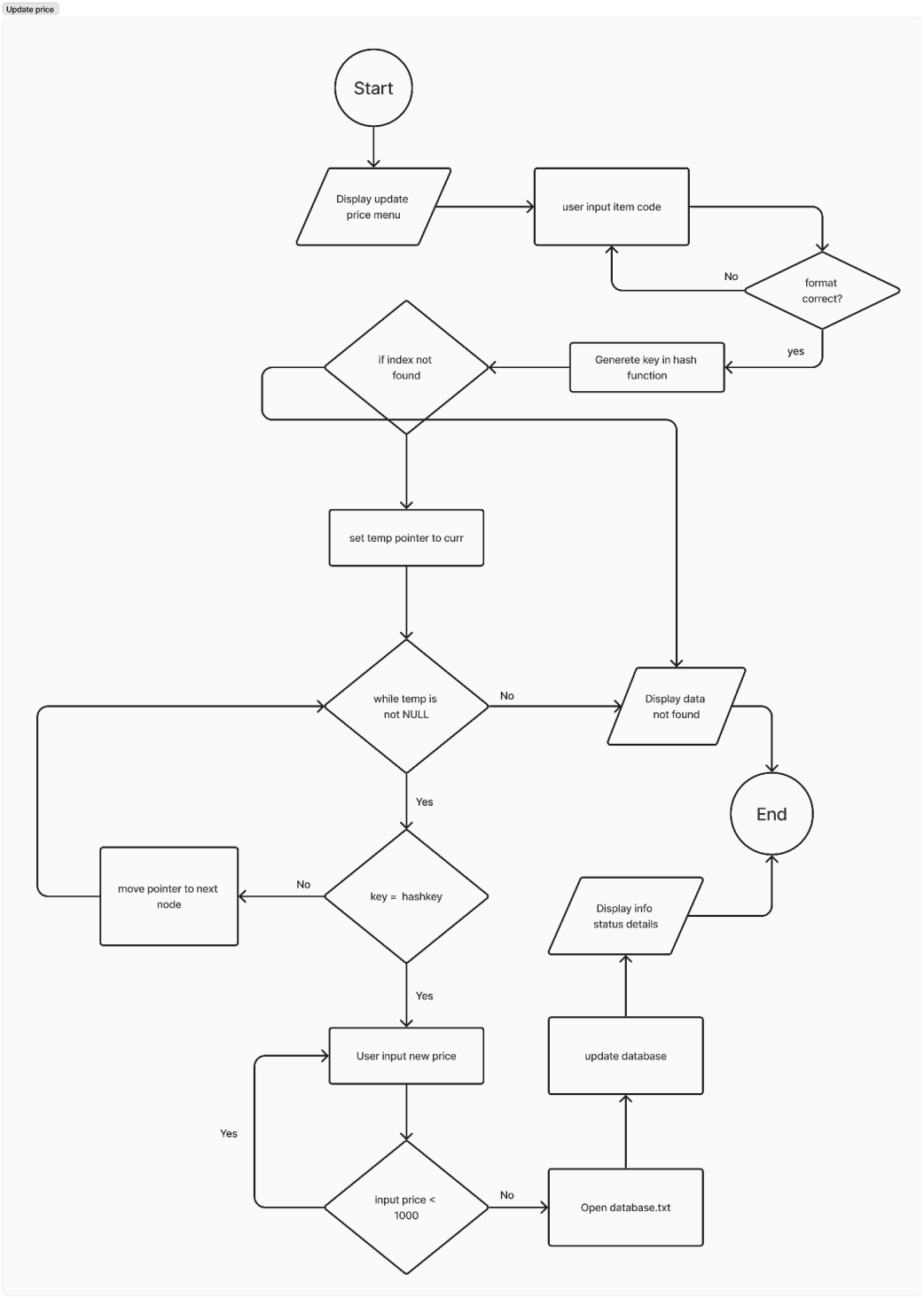
- Generate ID


Generate ID
Start
get input from user addstock function
1 word or 2 words ?
1 word
≥ 2 word
Generate first two character
take 2 first character from word
Generate first two character
take first character from each words
generate last 2 numbers
calculate lenght of product name + 20
Is there any collision?
yes
IDnumbers + 1
no
Generate ID
End

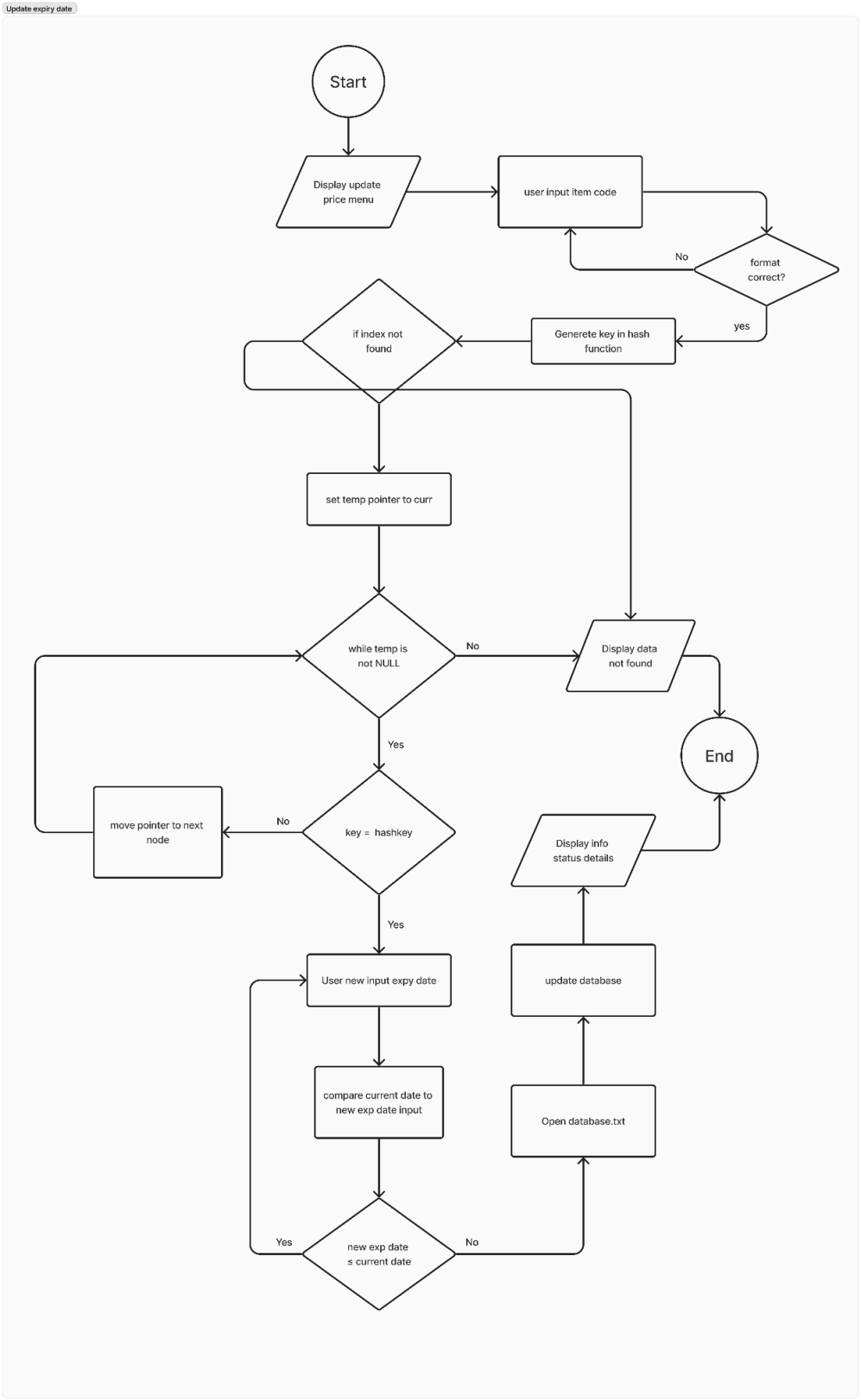
- Delete stock



- Update price



- Update expiry date



1. **Program overview**

- **Program Description**

Program yang kami buat merupakan program sistem Kasir dan Manajemen Warehouse yang dimana terbagi menjadi 2 menu utama yaitu menu kasir dan menu manajemen warehouse. Menu kasir sendiri terdiri dari menu purchase, cashflow, hash-table view, dan check expiration. Untuk menu manajemen warehouse terbagi menjadi menu add item, add stock, delete stock, hash-table view, dan update data. Metode yang kami gunakan pada program ini adalah metode Hashing, dimana hash function-nya menggunakan metode division dan collision menggunakan metode chaining sebagai handling-nya. Program juga menggunakan file processing untuk membaca file data barang yaitu database.txt serta melacak transaksi keuangan menggunakan flowchart.txt. Pada setiap barang yang terdapat pada database akan dibaca oleh program dan dimasukkan ke dalam hash-table. Setiap barang memiliki data yang terdiri dari ID, nama product, jumlah stock, harga, dan tanggal expired.

Dalam proses memasukkan setiap barang ke dalam hash-table, ID barang tersebut akan dimasukkan ke dalam hash function dengan metode division untuk memperoleh hash-value. ID sendiri terdiri dari format 2 huruf 2 angka dan bisa didapatkan melalui 2 cara, yaitu dengan mengambil setiap huruf pertama pada setiap kata apabila terdapat 2 kata dan mengambil 2 huruf pertama pada kata apabila hanya terdapat 1 kata. Untuk 2 angka terakhir dapat diperoleh dengan menghitung panjang nama produk lalu ditambahkan dengan 20. Hash function pada program ini akan menjumlahkan nilai ASCII dari ID barang yang kemudian akan dimoduli oleh table size. Setiap data barang nantinya akan dimasukkan ke dalam bentuk node mengingat collision handling yang digunakan berupa chaining, dimana setiap index pada hash-table bentuk single linked list.

- **Layout Design**

Desain dari program ini menyediakan tampilan yang sederhana dan menarik dimata pengguna dengan tujuan pengguna dapat melihat daftar barang serta mengakses fitur-fitur yang telah disediakan. Menu utama akan menampilkan display program “LengkapMart” disertai dengan dengan 3 menu utama yaitu :

1. Login as Cashier
2. Login as Warehouse Staff
3. Exit

User disini akan memilih pilihan menu yang diinginkan dengan menggunakan interaksi arrows keys sebagai navigasi dan memencet tombol enter untuk memilih opsi. Apabila user disini memilih Login

as Cashier sebagai opsinya, maka user akan diarahkan ke login menu dimana nantinya user akan diarahkan ke login menu untuk bisa mengakses fitur-fitur yang ada pada menu kasir. Pada menu login, user akan memasukkan input dan password yang telah tersedia dan hanya mempunyai 3x kesempatan untuk mengisi username dan password dengan benar.

Pada menu kasir program akan menampilkan tabel yang berisikan daftar barang. Tabel tersebut memuat ID, nama produk, stock harga, serta tanggal expired dari setiap barang. Tabel hanya akan menampilkan 10 daftar barang setiap pagenya dan user dapat berpindah page dengan menggunakan navigasi arrows keys. Dibawah tabel daftar barang, akan tersedia pilihan fitur yang dapat digunakan oleh user. Terdapat 5 fitur menu yang dapat dipilih oleh user yaitu :

1. Purchase
2. Cashflow
3. Hash-table View
4. Check Expiration
5. Logout

User akan menggunakan arrows keys sebagai navigasi dan tombol enter sebagai konfirmasi akan menu yang ingin dipilih. Pada menu Purchase & Check Expiration program akan menampilkan input apa saja yang harus dimasukkan oleh user. Program nantinya akan menampilkan feedback sesuai dengan apa yang telah user input, apakah input tersebut sudah benar ataupun belum, barang yang dicari apakah tersedia atau tidak, dan menampilkan proses pesan berhasil apabila proses dari fitur tersebut berjalan dengan lancar. Untuk menu Cashflow program akan menampilkan riwayat transaksi apa saja yang telah dilakukan oleh user pada menu Purchase dan akan menampilkan jumlah pendapatan yang diterima dari setiap produk serta total pendapatan yang diterima. Pada menu Hash-table View, program akan menampilkan bentuk hash-table yang sudah terisi oleh data-data dari daftar barang. Hash-table sendiri memiliki bentuk berupa tabel dengan dua kolom, dimana pada kolom pertama akan berisikan index seperti array dan pada kolom kedua berisikan data-data yang setiap index berbentuk linked list. Dimana data dalam bentuk node sudah diletakkan sesuai index berdasarkan hash-value yang diperoleh dari hash function. Apabila user ingin keluar dari menu kasir, user dapat memilih menu Logout untuk kembali ke menu utama.

Apabila user disini memilih Login As Warehouse Staff sebagai opsinya, maka user akan diarahkan ke login menu dimana nantinya user akan diarahkan ke login menu untuk bisa mengakses fitur-fitur yang ada pada menu Management Warehouse. Pada menu login, user akan memasukkan input

dan password yang telah tersedia dan hanya mempunyai 3x kesempatan untuk mengisi username dan password dengan benar sebagaimana proses yang sama seperti login sebagai kasir.

Pada menu Manajemen Warehouse program akan menampilkan tabel yang berisikan daftar barang. Tabel tersebut memuat ID, nama produk, stock harga, serta tanggal expired dari setiap barang. Tabel hanya akan menampilkan 10 daftar barang setiap pagenya dan user dapat berpindah page dengan menggunakan navigasi arrows keys. Dibawah tabel daftar barang, akan tersedia pilihan fitur yang dapat digunakan oleh user. Terdapat 7 fitur menu yang dapat dipilih oleh user yaitu :

1. Add Item
2. Add Stock
3. Delete Stock
4. Hash-table View
5. Check Expiration
6. Update Data
7. Logout

User akan menggunakan arrows keys sebagai navigasi dan tombol enter sebagai konfirmasi akan menu yang ingin dipilih. Pada menu Add item, Add stock, Delete Stock, & Update data program akan menampilkan input apa saja yang harus dimasukkan oleh user. Program nantinya akan menampilkan feedback sesuai dengan apa yang telah user input, apakah input tersebut sudah benar ataupun belum, barang yang dicari apakah tersedia atau tidak, dan menampilkan proses pesan berhasil apabila proses dari fitur tersebut berjalan dengan lancar. Pada menu Hash-table view, program akan menampilkan bentuk hash-table yang sudah terisi oleh data-data dari daftar barang. Hash-table sendiri memiliki bentuk berupa tabel dengan dua kolom, dimana pada kolom pertama akan berisikan index seperti array dan pada kolom kedua berisikan data-data yang setiap index berbentuk linked list. Dimana data dalam bentuk node sudah diletakkan sesuai index berdasarkan hash-value yang diperoleh dari hash function. Apabila user ingin keluar dari menu manajemen warehouse, user dapat memilih menu Logout untuk kembali ke menu utama.

Apabila user ingin keluar dari program, maka user dapat memilih menu Exit dan program akan memberikan konfirmasi apakah user ingin benar-benar keluar atau ingin lanjut menggunakan program. Ketika user memiilih untuk keluar dari program, program akan menampilkan pesan terimakasih karena telah menggunakan program.

- **Program Features**

Dimulai dari menu utama yaitu **Login menu** menu ini berfungsi untuk sebagai opsi untuk mengakses fitur-fitur yang tersedia pada menu kasir ataupun manajemen menu. Ketika user memilih opsi ini maka user akan diarahkan untuk login sebagai kasir melalui menu login. Pada menu login ini user akan diberikan kesempatan untuk memasukkan username dan password sebanyak tiga kali kesempatan, apabila user gagal dalam melakukan memasukkan username dan password yang sesuai, program akan mengembalikan user ke menu utama. Setelah user berhasil login, program akan menampilkan Hash-table yang berisikan barang-barang yang tersedia serta menampilkan fitur-fitur menu dari menu kasir.

Pertama dari menu kasir yaitu menu **Purchase**. Sesuai dengan namanya, menu ini berfungsi untuk melakukan proses pembelian barang. Pertama pengguna akan diminta melakukan input ID barang berupa “Input item ID [A-Z][A-Z][1-9][1-9] : “ yang mengharuskan mengisi sesuai dengan format yang benar. Apabila user memasukkan format dengan salah maka program akan melakukan looping hingga jawaban yang diberikan oleh user sesuai. Akan tetapi, jika format yang telah dimasukkan user telah sesuai tetapi barang tidak ditemukkan, program akan menampilkan “Status: ID %s not found in the database. Please try again.”. Ketika user berhasil melakukan input barang program akan menampilkan “Status: ID %s found in the database at index %d.” Setelah proses input ID barang berhasil user akan diarahkan untuk melakukan input total stock dari barang tersebut yang ingin dibeli “ Input total stock to purchase [1-%d] : “ dengan ketentuan angka 1-%d barang tersebut tersedia. Sedikit tambahan, jika user data yang diinput tidak sesuai dengan ketentuan maka program akan melakukan looping hingga user menginput dengan format yang sesuai. Setelah proses pembelian stok barang berhasil program akan menampilkan purchase info, berapa stock yang barang yang tersisa serta tanggal barang tersebut dibeli. Selanjutnya program akan menawarkan user, apakah user ingin membeli barang lagi atau tidak. Apabila user ingin melakukan pemberian barang lagi maka user akan diarahkan untuk melakukan input seperti yang sudah dijelaskan pada awal tadi. Sebaliknya, jika user menolak tawaran untuk melakukan belanja lagi, maka user akan diarahkan ke purchase summary dimana struct detail barang yang dibeli akan ditampilkan oleh program.

Kedua, terdapat menu **Cashflow.** Pada menu ini user dapat melihat riwayat transaksi apa saja yang telah dilakukan user dari aktivitas yang dilakukan pada menu **Purchase.** Disini program akan menampilkan seluruh riwayat transaksi user dalam bentuk tabel dan menghitung pendapatan dari setiap barang serta program akan menjumlah earnings dari setiap barang untuk dihitung totalnya.

Ketiga, menu **Hash-table view**. Menu ini berfungsi untuk menampilkan hasil akhir dari hash table. Hasil akhir dari hash-table sendiri didapatkan melalui proses memasukkan setiap ID barang ke dalam hash function yang menggunakan metode **Division**. Dimana akan diperoleh Hash-value dari hasil

hash function tersebut, untuk menempatkan data barang ke dalam index ke sekian berdasarkan Hash-value. Nantinya, setiap index masing-masing akan berbentuk single linked list dikarenakan collision handling yang digunakan adalah **Chaining**. Jadi setiap data barang akan berbentuk node, dan apabila ada collision akan dilakukan push tail single linked list pada umumnya.

Keempat, menu **Check Expiration**. Seperti halnya pada menu **Purchase** user akan diminta untuk melakukan ID barang “Input item ID [A-Z][A-Z][1-9][1-9] : “ yang mengharuskan mengisi sesuai dengan format yang benar. Apabila user memasukkan format dengan salah maka program akan melakukan looping hingga jawaban yang diberikan oleh user sesuai. Akan tetapi, jika format yang telah dimasukkan user telah sesuai tetapi barang tidak ditemukkan, program akan menampilkan “Status: ID %s not found in the database. Please try again.”. Ketika user berhasil melakukan input barang program akan menampilkan “Status: ID %s found in the database at index %d.” Program juga akan menampilkan apakah barang tersebut sudah melewati batas expired atau tidak. Ketika barang belum melewati batas expired program akan menampilkan pesan “Expiration Info: %s is not yet expired ” sebaliknya jika barang sudah melewati batas expired maka program akan menampilkan pesan “Expiration Info: %s is yet expired”.

Kelima, menu yang terakhir untuk menu kasir yaitu **Logout.** Apabila user memilih menu ini program akan melakukan proses exit dan user akan diarahkan kembali ke menu utama.

Pertama untuk menu manajemen warehouse yaitu menu **Add Item**. sebagaimana namanya pada menu ini user akan diminta untuk menambahkan barang baru ke dalam database. Pertama user akan diminta untuk memasukkan nama barang baru “Input new product name (3 - 15 Characters) : ”, apabila user memasukkan nama barang yang sama maka user akan diminta untuk melakukan input lagi”Status: Product already existed!”, serta sebagaimana menu input lainnya apabila user melakukan kesalahan pada input format, user akan diminta untuk melakukan input ulang dan apabila ID barang tidak ditemukan, program akan mengambilkan user ke menu manajemen warehouse. Selanjutya user akan diarahkan untuk melakukan input stock barang dari barang tersebut yang ingin ditambahkan”Input starting stocks [1-100] : ” dengan ketentuan jumlah barang yang diinput [1-100]. Jika user memasukkan data yang diinput tidak sesuai dengan ketentuan maka program akan melakukan looping hingga user menginput dengan format yang sesuai. Setelah proses memasukkan jumlah stock, user akan diminta untuk memasukkan harga dari barang yang dimasukkan “Input price (Min. Value: 1000) : “ dengan ketentuan format barang yang dimasukkan harus minimal mempunyai harga 1000, seperti halnya sebelumnya apabila user melakukan input kurang dari format yang ditentukan program akan memvalidasi inputan user. Step yang terakhir yaitu user akan diminta untuk melakukan input expiry date “Input expiry date (After today date) [Format: DD/MM/YYYY] : “. Format yang dipakai mengharuskan user untuk input sesuai format yang telah ditentukan dan melakukan input program setelah tanggal pada hari itu user menggunakan program, jika

tidak program akan melakukan validasi pada input user “Date must be after (current date)!”. Setelah semua proses pada program jika berjalan dengan lancar, program akan menampilkan ID yang telah dibuat oleh sistem serta menampilkan status berhasil “Status: Product added to database! “.

kedua dari menu manajemen warehouse yaitu menu **Add Stock** . Sesuai dengan namanya, menu ini berfungsi untuk menambahkan stock barang yang telah ada pada database. Pertama user akan diminta melakukan input ID barang berupa “Input item ID [A-Z][A-Z][1-9][1-9] : “ yang mengharuskan mengisi sesuai dengan format yang benar. Apabila user memasukkan format dengan salah maka program akan melakukan looping hingga jawaban yang diberikan oleh user sesuai. Akan tetapi, jika format yang telah dimasukkan user telah sesuai tetapi barang tidak ditemukkan, program akan menampilkan “Status: ID %s not found in the database. Please try again.”. Ketika user berhasil melakukan input baran, program akan menampilkan “Status: ID %s found in the database at index %d.” Setelah proses input ID barang berhasil user akan diarahkan untuk melakukan input jumlah stock yang ingin ditambahkan pada barang tersebut “Input total stock to add [1-100] : “ dengan ketentuan jumlah barang yang diinput [1-100]. Jika user memasukkan data yang diinput tidak sesuai dengan ketentuan maka program akan melakukan looping hingga user menginput dengan format yang sesuai. Apabila semua proses tadi dijalankan dengan lancar, maka program akan menampilkan pesan berhasil pada menu **Add Stock** beserta detail-detailnya

Ketiga, menu **Delete Stock**. Pada menu ini user dapat mengurangi jumlah stock barang yang tersedia pada database ataupun menghapus barang tersebut. Seperti halnya pada menu yang lain user akan diminta melakukan input ID barang berupa “Input item ID [A-Z][A-Z][1-9][1-9] : “ yang mengharuskan mengisi sesuai dengan format yang benar. Apabila user memasukkan format dengan salah maka program akan melakukan looping hingga jawaban yang diberikan oleh user sesuai. Akan tetapi, jika format yang telah dimasukkan user telah sesuai tetapi barang tidak ditemukkan, program akan menampilkan “Status: ID %s not found in the database. Please try again.”. Ketika user berhasil melakukan input barang, program akan menampilkan “Status: ID %s found in the database at index %d.” Selanjutnya user akan diminta untuk melakukan input jumlah stock yang ingin dihapus ataupun dikurangi “Input total stock to delete [1-%d] : “ dengan ketentuan angka 1-%d barang tersebut tersedia. Apabila user disini memasukkan input angka maksimum dari barang tersebut, program akan otomatis menghapus barang tersebut dari database”Stock Info: %s deleted from database successfully on (current date) !.” Akan tetapi, jika user melakukan input di antara jumlah stock yang tersedia, program hanya akan mengurangi jumlah stock dari ketersediaan barang tersebut.

Menu keempat dan kelima pada menu manajemen warehouse yaitu **Hash-table View** & **Check Expiration** berfungsi sebagaimana halnya sama dengan menu **Hash-table View & Check Expiration** pada menu kasir.

Keenam, menu **Update Data.** Pada menu ini terbagi menjadi 2 menu bagian utama yaitu menu **Update Price** & **Update Expiry Date.** Jika user disini memilih untuk melakukan menu **Update by Price.** User akan diminta untuk memasukkan item ID "Input item ID [A-Z][A-Z][1-9][1-9] : ", serta jika user melakukan kesalahan input format ataupun ID tidak ditemukan, program akan melakukan validasi seperti halnya pada menu-menu sebelumnya. Apabila user berhasil melakukan process input item ID, user akan diarahkan untuk melakukan input harga barang baru yang ingin dimasukkan "Input new %s price (Min. Value: 1000) : " dengan ketentuan format barang yang dimasukkan harus minimal mempunyai harga 1000, seperti halnya sebelumnya apabila user melakukan input kurang dari format yang ditentukan program akan memvalidasi inputan user"Status: Invalid price format! ". Apabila user berhasil melakukan input dengan ketentuan format yang berlaku program akan menampilkan status berhasil" Status: %s price updated in the database!". Selanjutnya untuk menu **Update Expiry date,** User akan diminta untuk melakukan input item ID seperti halnya menu **Update Price.** Setelah user berhasil melakukan input item ID user akan diarahkan untuk melakukan input expiry date pada barang yang dipilih"Input new item expiry date (After 15/06/2024) [DD/MM/YYYY] : " .Format yang dipakai mengharuskan user untuk input sesuai format yang telah ditentukan dan melakukan input program setelah tanggal pada hari itu user menggunakan program , jika tidak program akan melakukan validasi pada input user.

Terakhir, menu **Logout**, menu ini berfungsi seperti halnya menu **Logout** pada menu kasir. Apabila user memilih menu ini program akan melakukan proses exit dan user akan diarahkan kembali ke menu utama.

Menu yang terakhir pada menu utama yaitu menu **Exit.** Sebelum user benar-benar keluar meninggalkan program, program akan memberi konfimarsi kepada user apakah user ingin benar-benar meninggalkan program "Are you sure you want to quit the application? (yes/no)". Disini apabila user sudah merasa cukup dengan menggunakan program, user dapat memilih jawaban "yes" untuk keluar dari program dan "no" untuk melanjutkan menggunakan program.

- **Program Details**

Pada bagian ini akan dijelaskan detail dari proses yang terjadi di dalam program. Dimulai dari process read file, file database.txt dengan menggunakan file opening type "r" (read). Selama belum mencapai akhir dari file, program akan membaca data barang per baris. Setiap satu baris akan dilakukan

proses insert data ke dalam hash-table. Kemudian dalam proses insert sendiri dimulai dari pembuatan node baru, dimana seluruh data barang seperti ID, nama produk, jumlah stock, harga, dan tanggal expired akan dimasukan ke dalam node tersebut. Pada node baru terdapat satu elemen untuk menyimpan hasil Hash-value yang didapat dari hash function. Program ini juga menggunakan satu file txt lagi yang berfungsi sebagai pengecekan riwayat transaksi yaitu cashflow.txt.

Seperti yang telah dijelaskan, hash function pada program ini menggunakan metode division. ID dari barang yang di insert dimasukkan ke dalam hash function, kemudian nilai ASCII dari setiap karakter code di totalkan. Setelah total nilai ASCII dari ID didapatkan, selanjutnya memasukkan total ASCII tersebut ke dalam rumus division yakni Value = Total ASCII % Table Size. untuk Table Size sendiri mengikuti jumlah index yang telah ditetapkan yaitu 50. Semisal diambil contoh, misalkan sebuah ID produk 'Ayam Bakar' 'AB30', yang masing-masing memiliki nilai ASCII seperti A = 65, B = 66 , 3 = 51, dan 0 = 48 apabila ditotalkan menjadi 230 (65+66+51+48). Kemudian dimasukkan kedalam rumus division, Value = 230 % 50 menghasilkan nilai 30. Karena hasil dari hash function atau valuenya 30, maka AB30 diletakkan pada index 30 di hash table.

Jikalau terjadi collision pada index tersebut, maka akan ditangani dengan metode **Chaining**. Jadi setelah Hash-value didapatkan dari hash function, maka node baru yang berisikan data produk akan diletakkan pada linked list sesuai index keynya. Sama seperti metode push singly linked list, apabila list tersebut belum memiliki node maka node baru akan diset menjadi head sekaligus tail. Dan apabila list tersebut sudah memiliki node, maka pointer next dari tail akan menunjuk ke new node kemudian new node diset menjadi tail.

Beralih pada fitur menu pada program yang menggunakan fitur search seperti **Purchase, Check Expiration, Add Item, Add Stock, Delete Stock, & Update Data. Se**telah seluruh input dan validasi sudah benar, input code barang dari user akan dimasukkan ke dalam hash function untuk mendapatkan index yang dituju. Kemudian program akan melakukan searching dengan langsung menunjuk index yang dituju dan melakukan traversal list pada index tersebut. Selama traversal list, program akan mencari code barang yang cocok. Apabila data sudah ditemukan, program akan memberikan feedback sesuai dengan kegunaan dari fitur-fitur itu sendiri seperti mengurangi stock barang, menghapus barang, menambahkan jumlah barang, mengupdate harga & tanggal expired barang hingga mengecek apakah barang tersebut sudah melewati batas expired atau belum. Untuk fitur fitur yang sudah dijelaskan di kalimat sebelumnya program akan menggunakan file processing "w" untuk mengubah data yang terdapat pada database.txt & cashflow.txt untuk menu **Purchase**.

Selanjutnya adalah menu **Hash-table View**. Proses yang dilakukan pada menu ini cukup sederhana, dimana, program akan melakukan looping dari index 0 dan pada setiap index akan dilakukan traversal list. Selama proses looping tersebut program akan k menampilkan data-data yang tersimpan pada hash table. Hasil akhir dari proses ini adalah bentuk Hash-table yang sudah lengkap beserta data barang dalam bentuk single linked list sesuai indexnya.

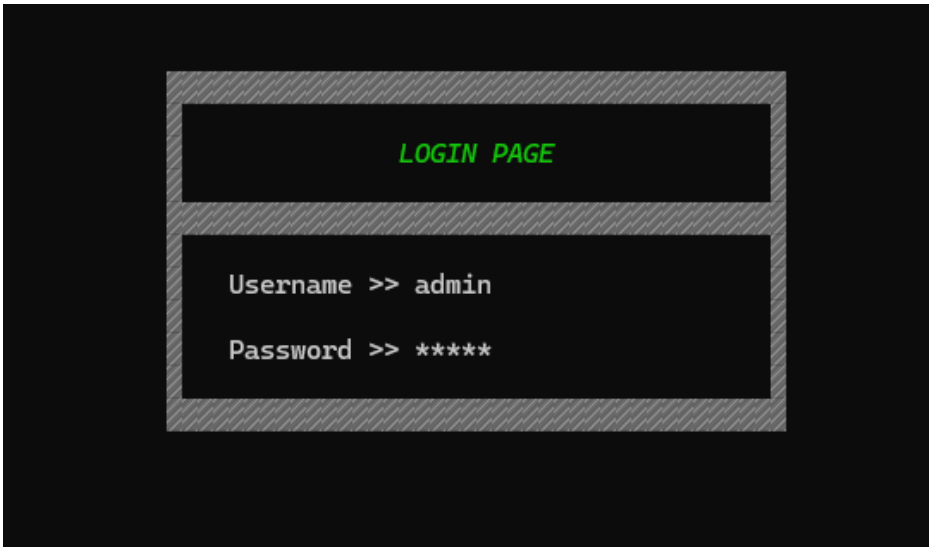
Untuk menu **Cashflow** program akan membaca file cashflow.txt dan akan membaca seluruh data yang terdapat pada cashflow.txt dan menampilkan seluruh data transaksi dengan menggunakan looping. Program juga akan menghitung pendapatan dari setiap barang dengan menggunakan rumus Earnings = item.price * item.stock. Setelah menghitung seluruh pendapatan pada setiap barang, program akan menampilkan seluruh pendapatan dengan menjumlahkan setiap pendapatan yang didapatkan dari barang setiap barang.

BAB IV

4. Result

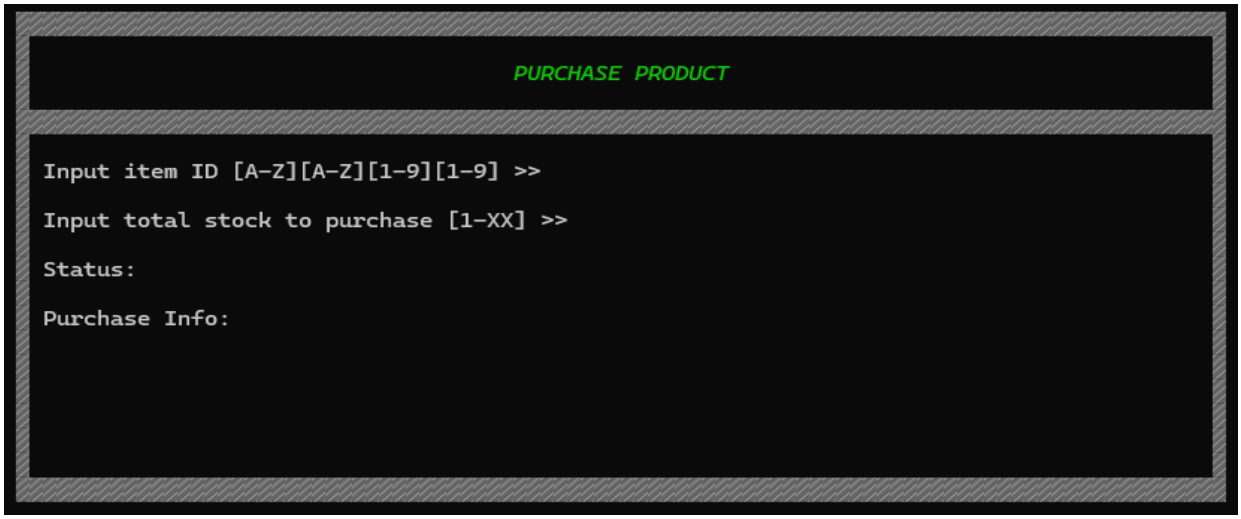
- **Program Screenshot**

1.0 Tampilan Menu Awal Program

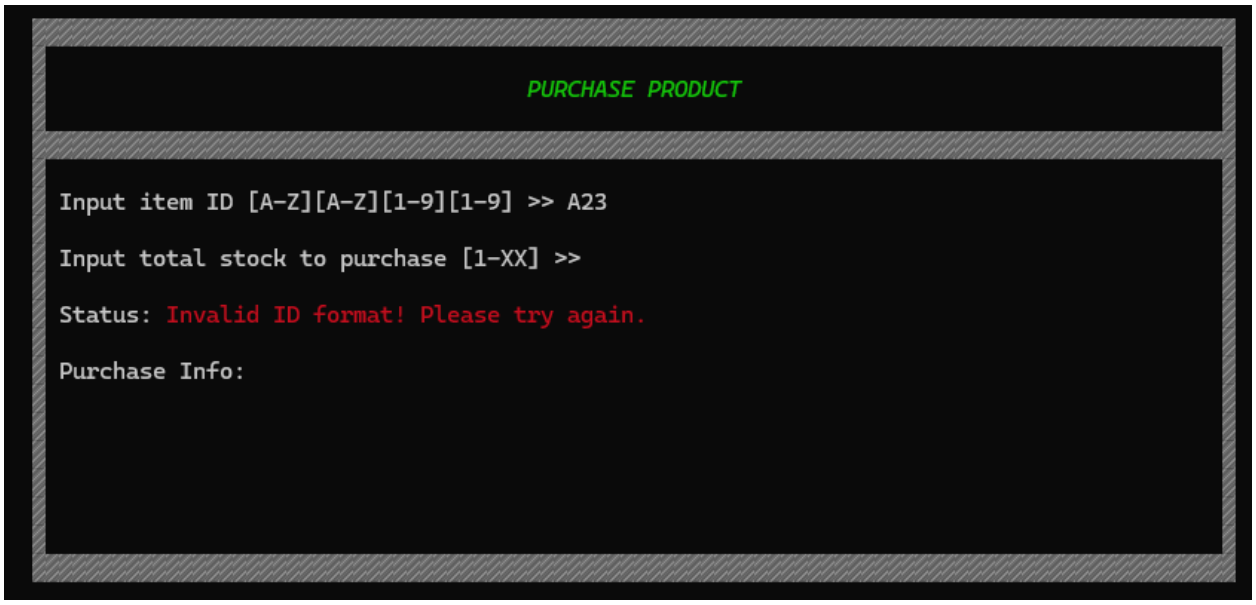


2.0 Tampilan Login Page

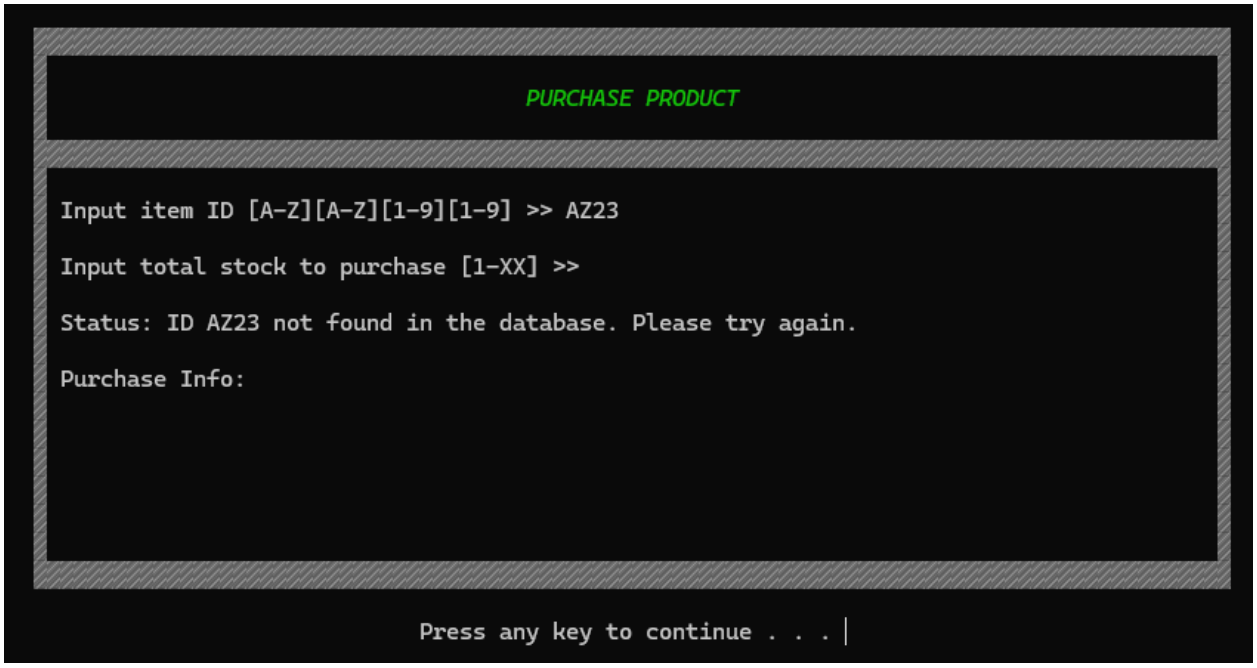
3.0 Tampilan Menu Awal Pada Menu Kasir



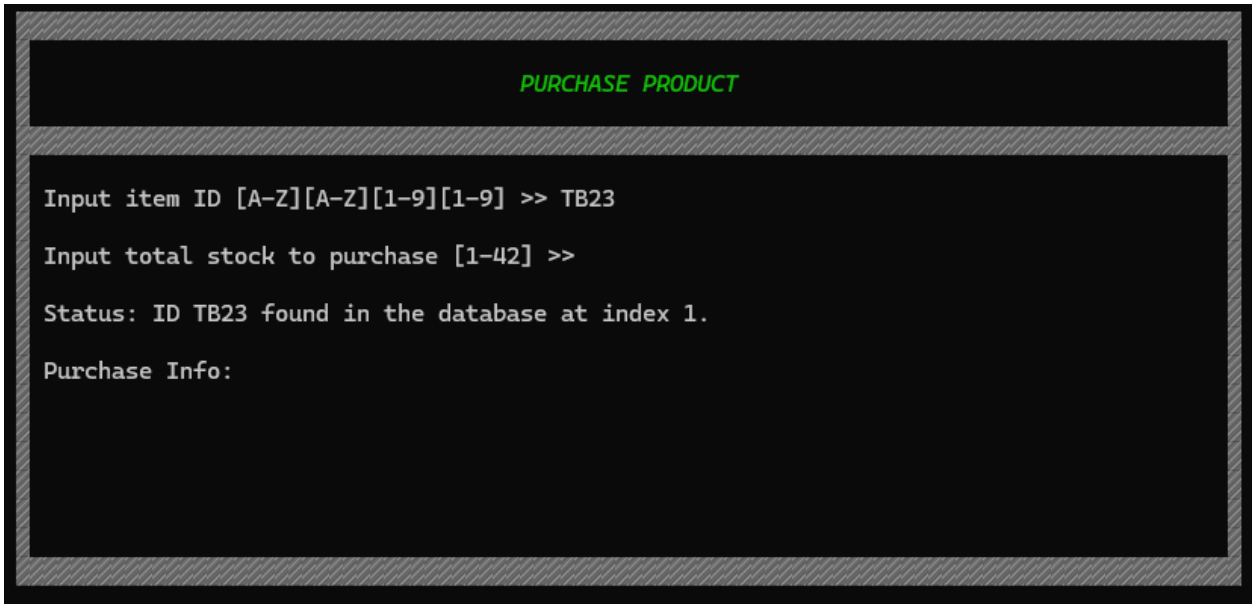
4.0 Tampilan Awal Purchase Product



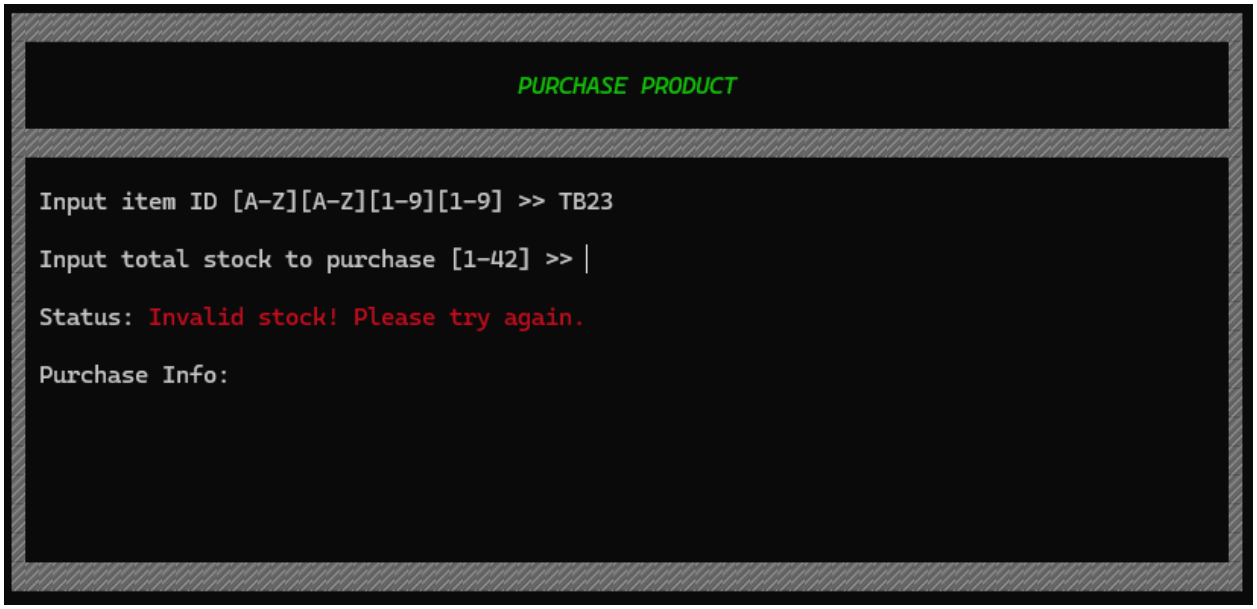
4.1 Tampilan Purchase Product Jika Format Tidak Sesuai



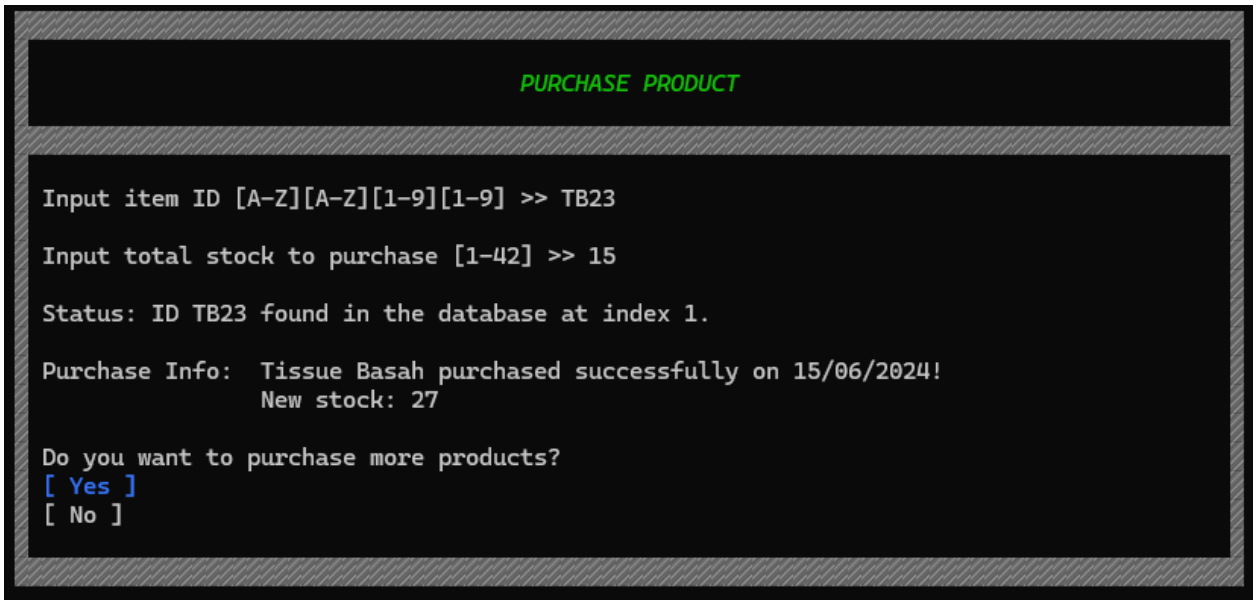
4.2 Tampilan Purchase Product Jika ID tidak Ditemukan Dalam Database



4.3 Tampilan Purchase Product Jika ID Berhasil Ditemukan



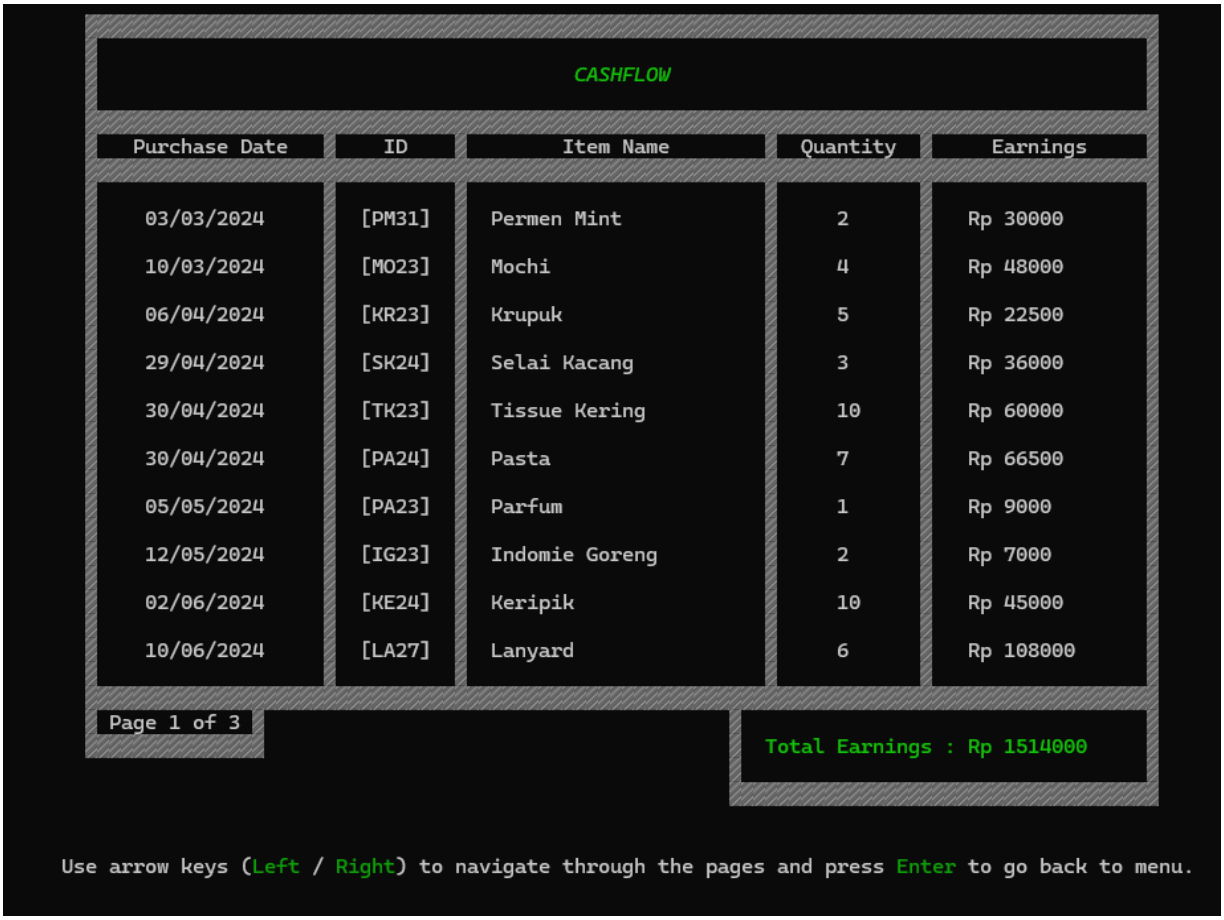
4.4 Tampilan Purchase Product Jika Stock Input Tidak Sesuai



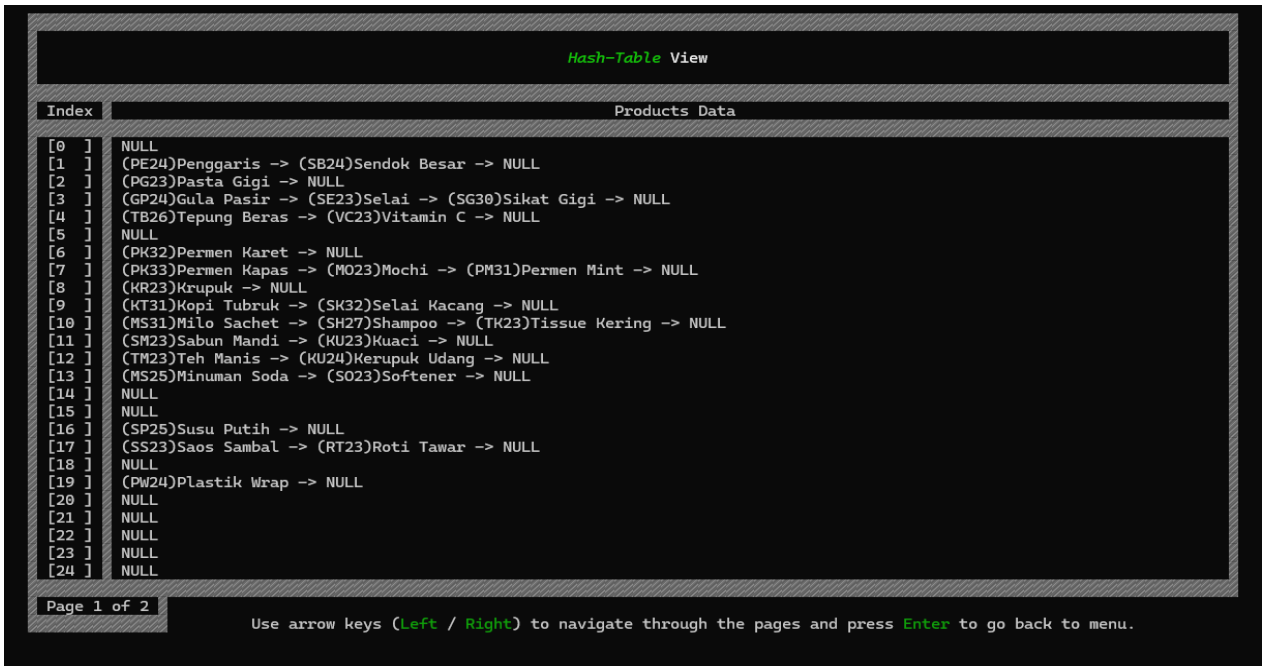
4.5 Tampilan Purchase Product Jika Berhasil Membeli Stock Barang

4.6 Tampilan Purchase Product Jika Membeli Barang Dengan Input Batas Maksimum

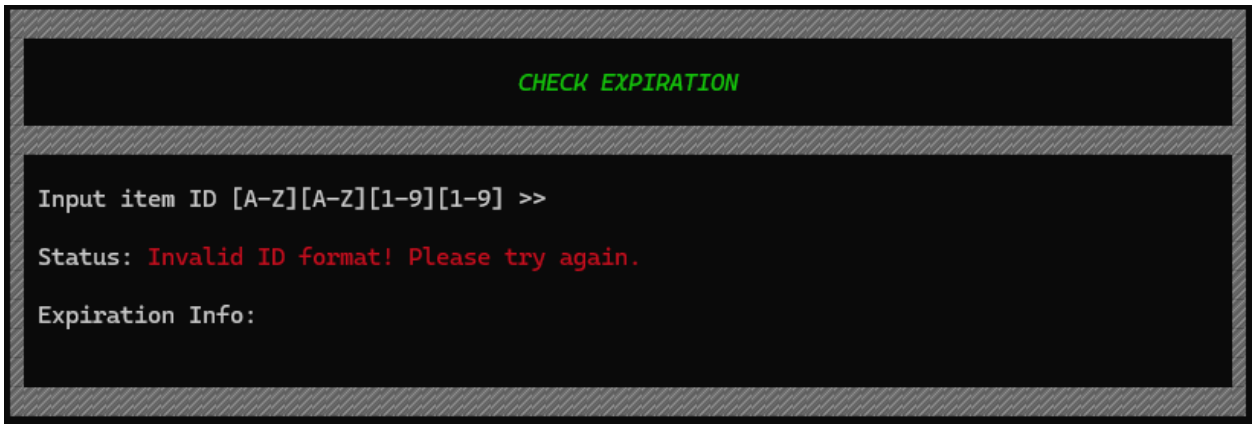
4.7 Tampilan Purchase Summary



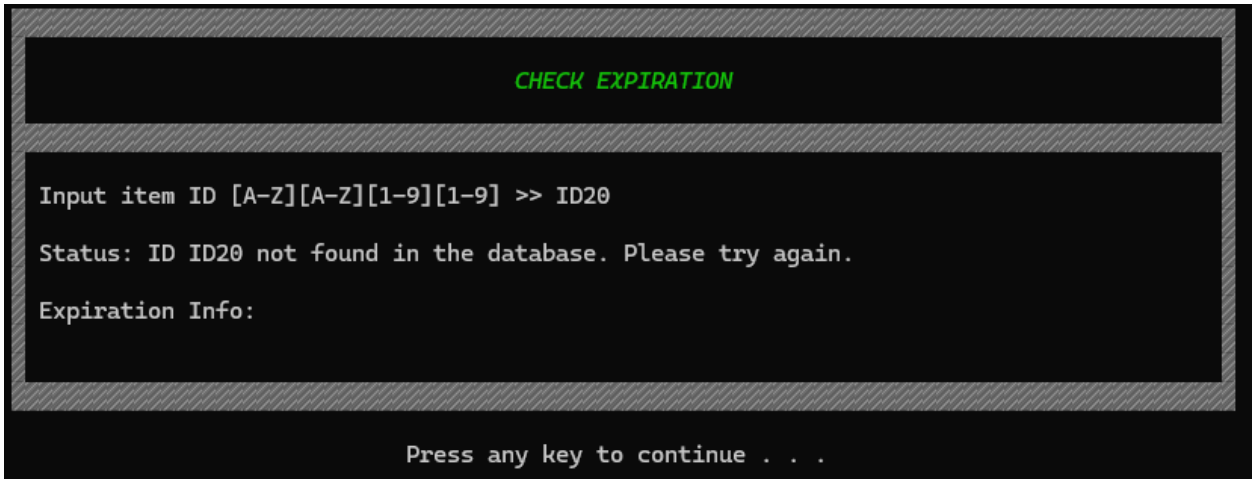
5.0 Tampilan Menu Cashflow



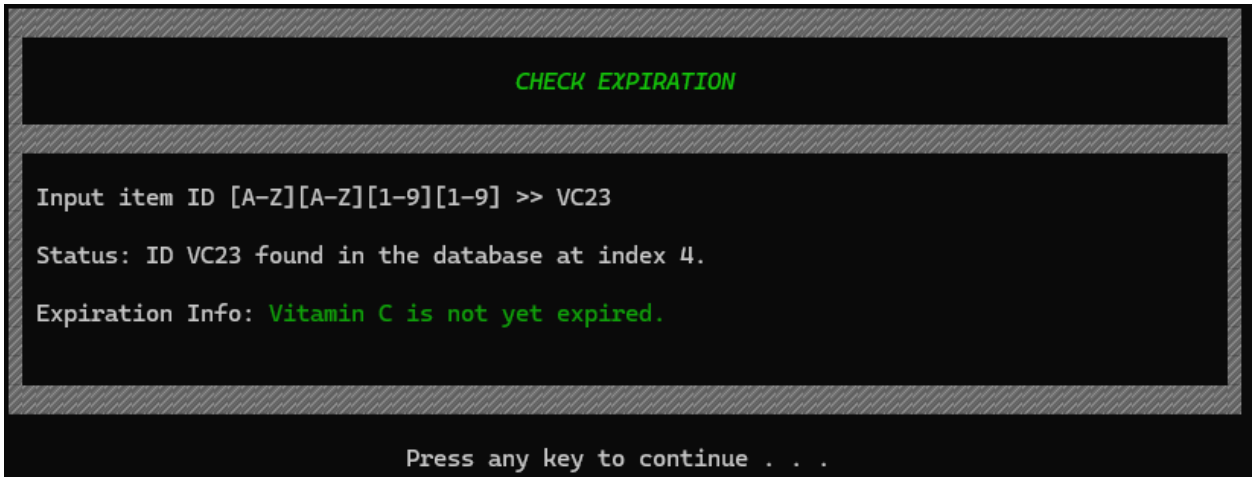
6.0 Tampilan Menu Hash-table View



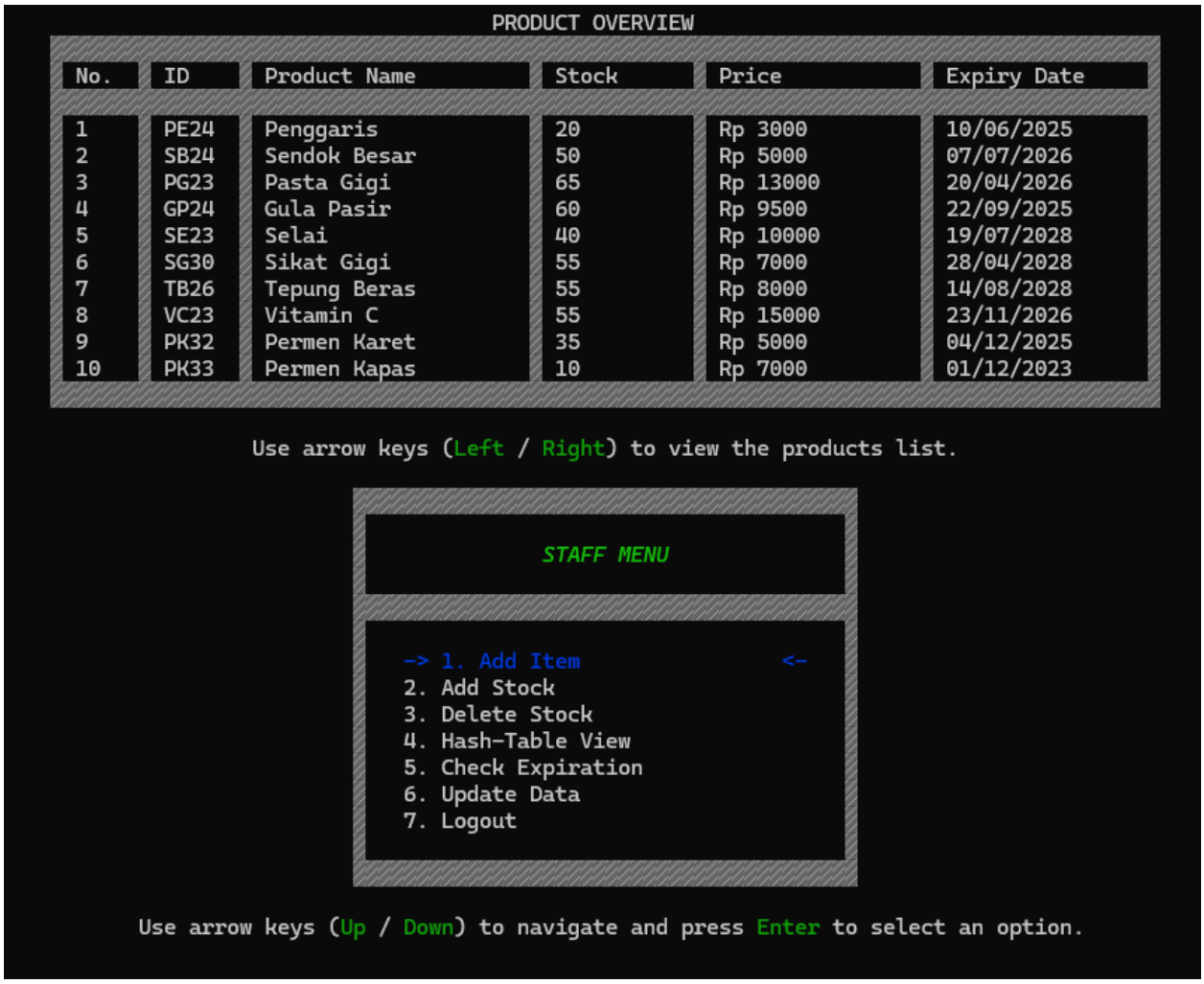
7.0 Tampilan Menu Check Expration Jika Salah Input Format



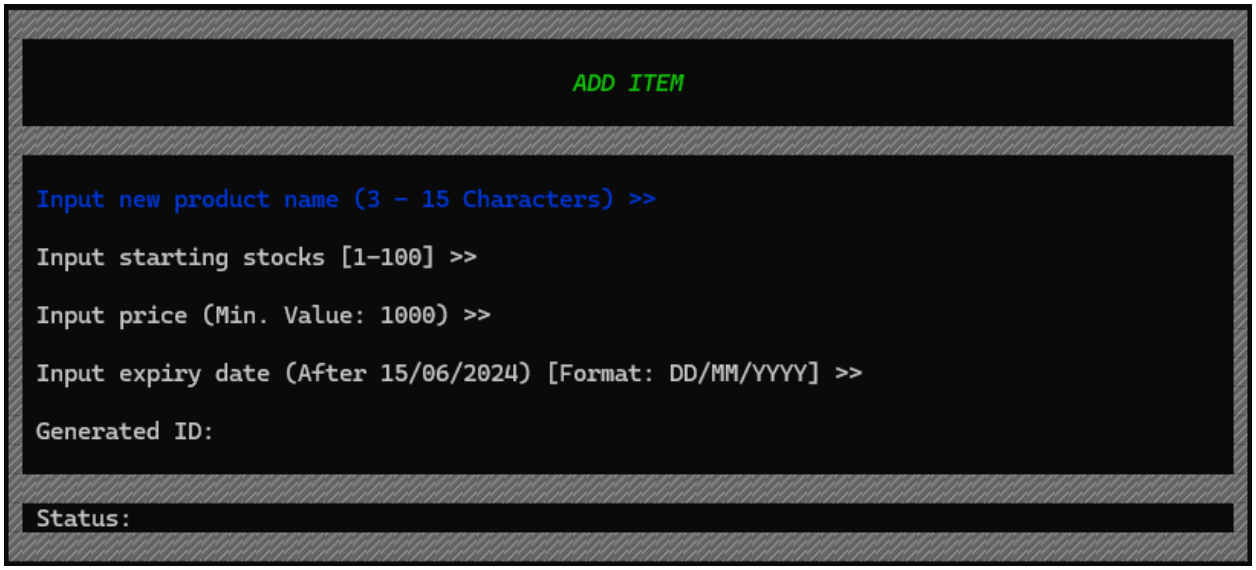
7.1 Tampilan Menu Check Expiration Jika ID Tidak Ditemukan



7.2 Tampilan Menu Check Expiration Jika ID Ditemukan



8.0 Tampilan Menu Utama Pada Product Overview

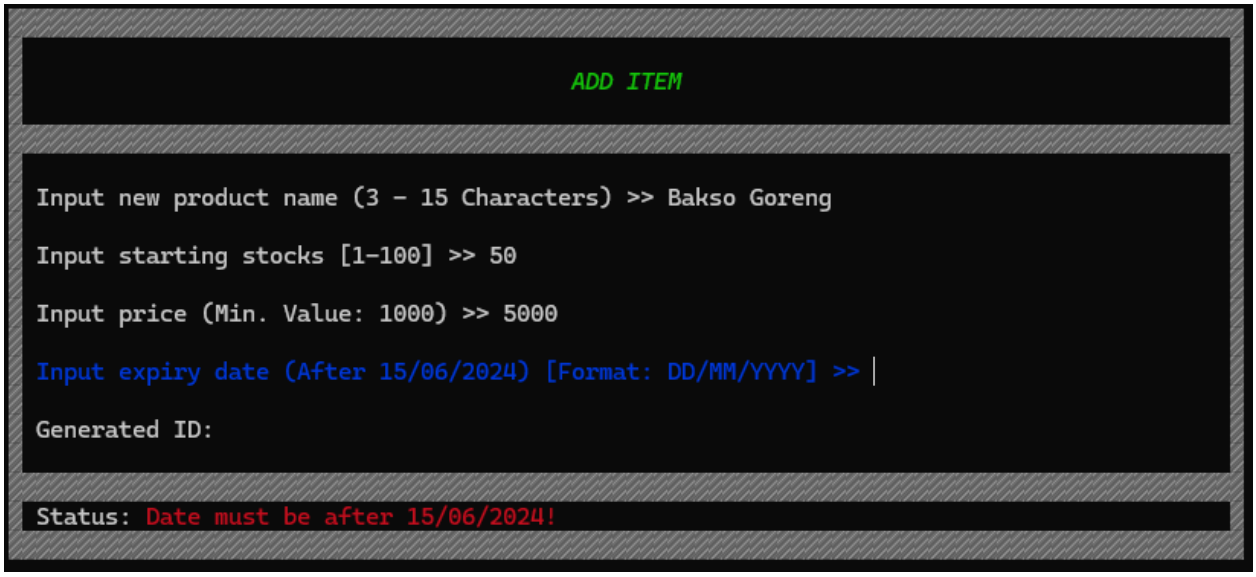


9.0 Tampilan Menu Awal Add Item

9.1 Tampilan Menu Add Item Jika Salah Input Nama Produk

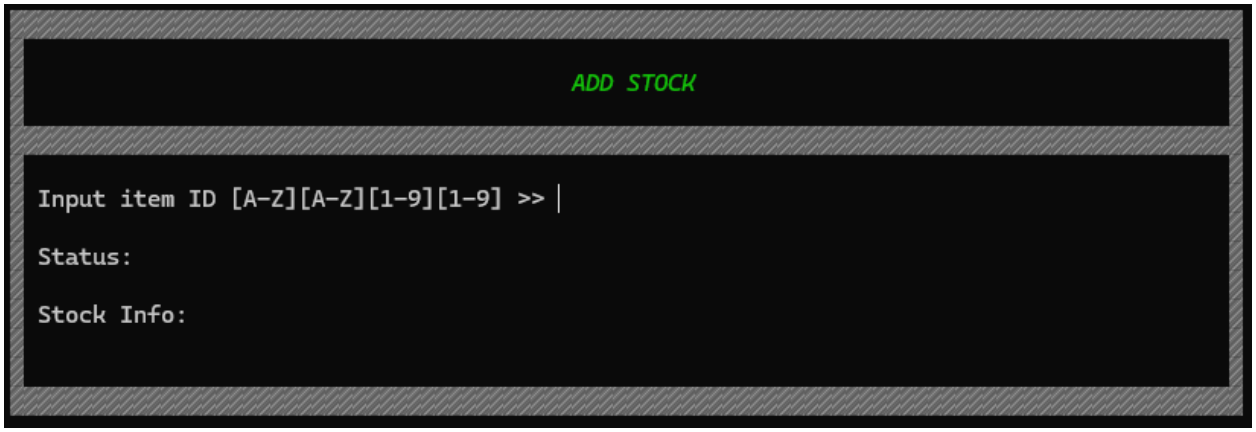
9.2 Tampilan Menu Add Item Jika Salah Memasukkan Input Stock

9.3 Tampilan Menu Add Item Jika Salah Memasukkan Harga

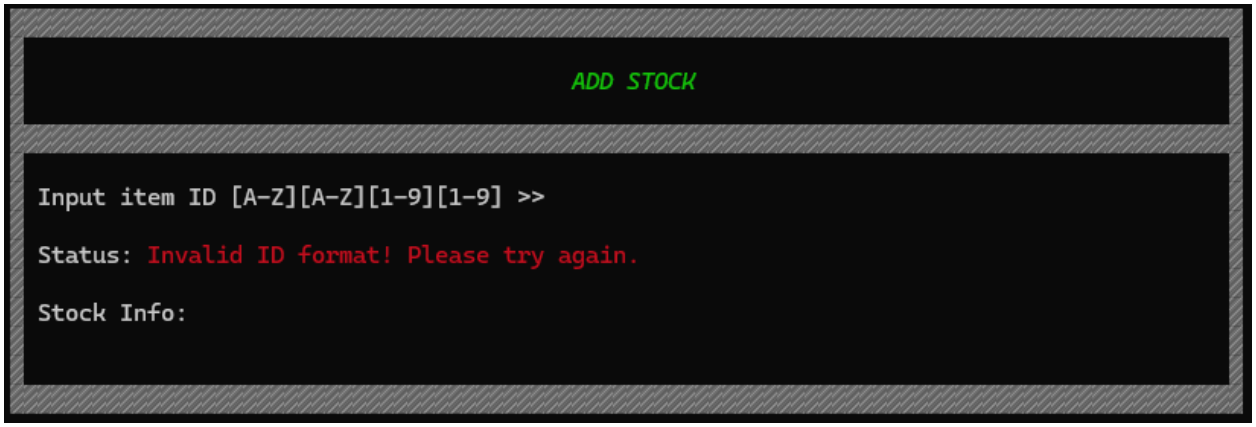


9.4 Tampilan Menu Add Item Jika Salah Memasukkan Format Tanggal

9.5 Tampilan Menu Add Item Jika Berhasil Menambahkan Barang



10.0 Tampilan Menu Awal Add Stock



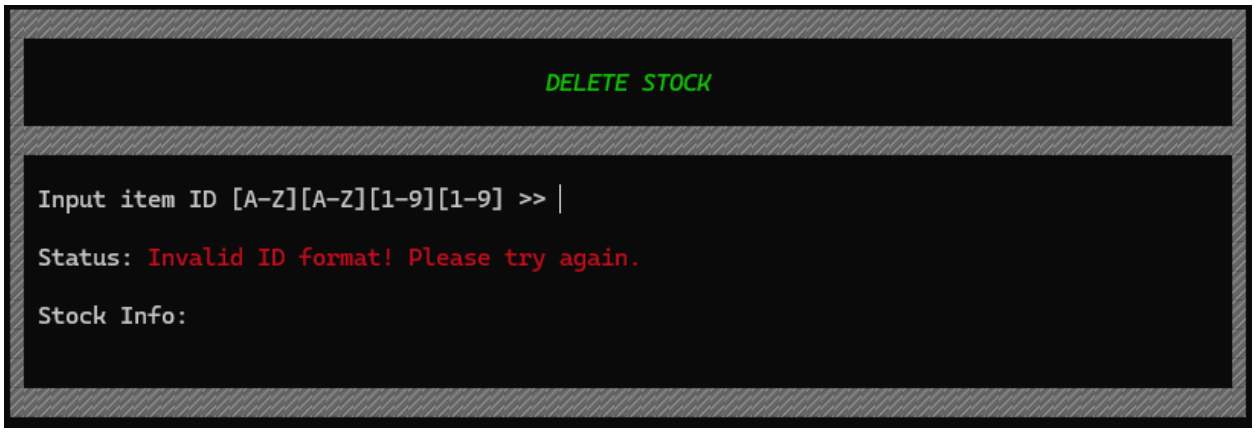
10.1 Tampilan Add Stock Jika Salah Memasukkan Format ID

10.2 Tampilan Add Stock Jika Tidak Menemukan ID

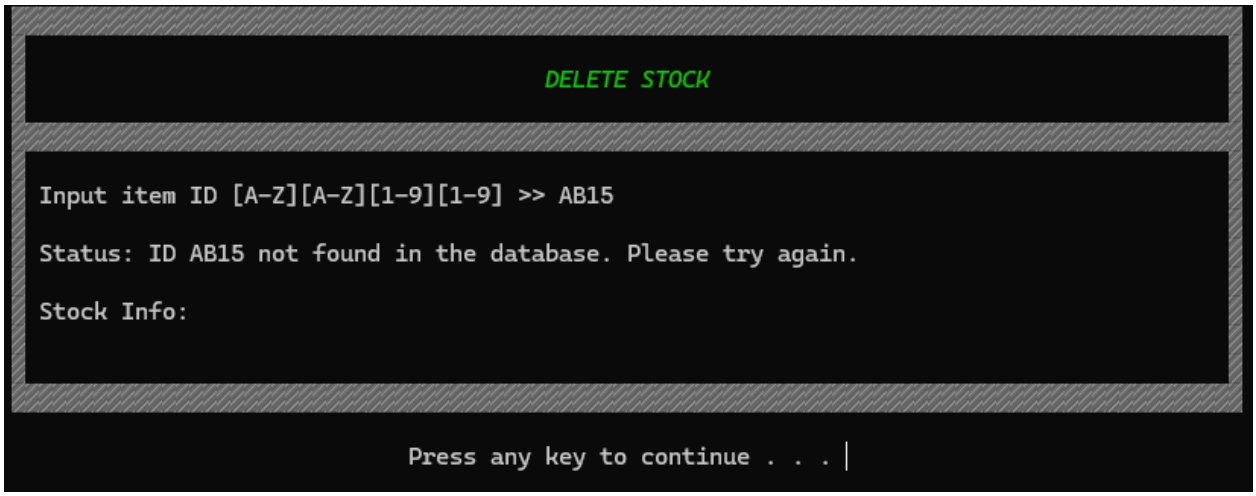
10.3 Tampilan Add Stock Jika Salah Memasukkan Angka Stock

10.4 Tampilan Add Stock Jika Berhasil Menambahkan Jumlah Stock

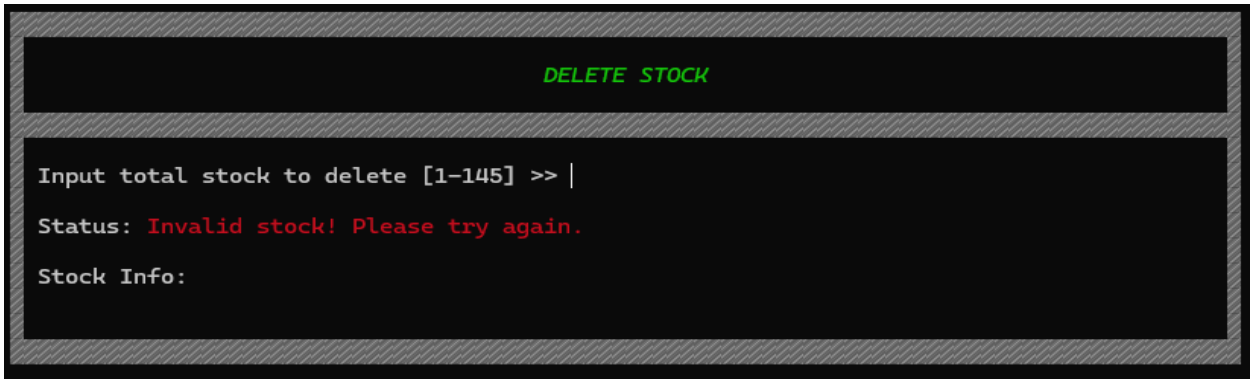
11.0 Tampilan Menu Awal Delete Stock



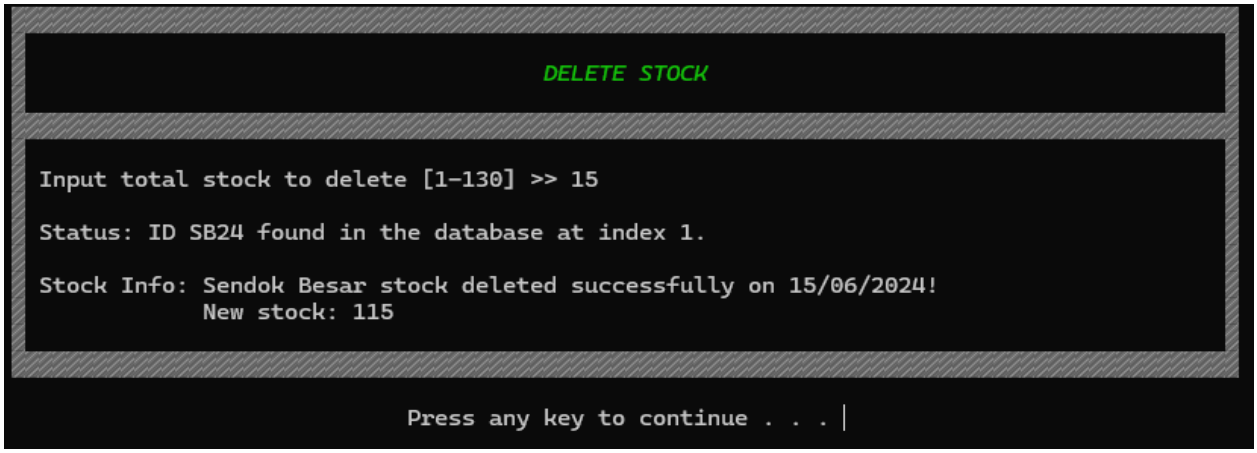
11.1 Tampilan Menu Delete Stock Jika Salah Memasukkan Format



11.2 Tampilan Menu Delete Stock Jika TIdak Menemukan ID

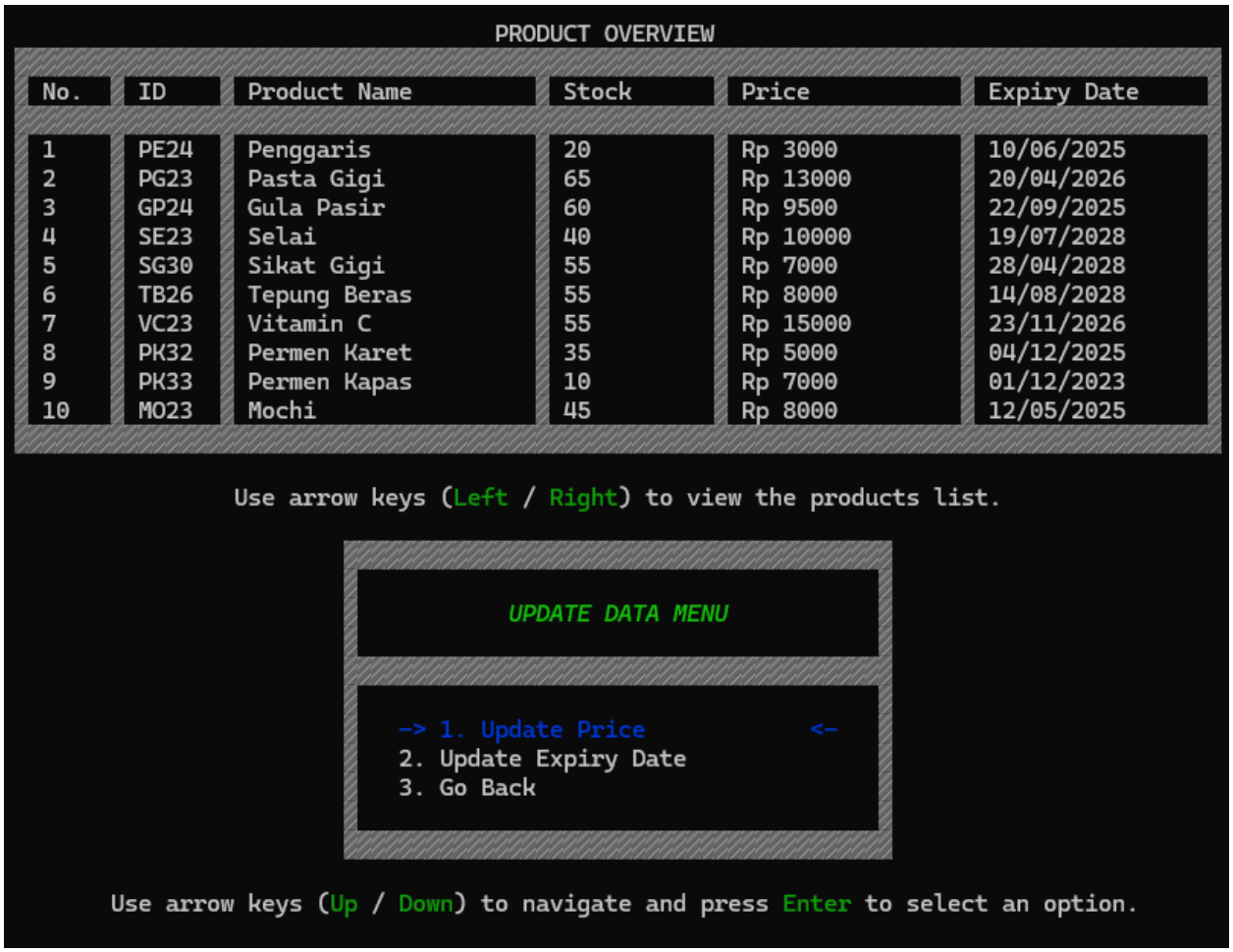


11.3 Tampilan Menu Delete Stock Jika Salah Memasukkan Angka



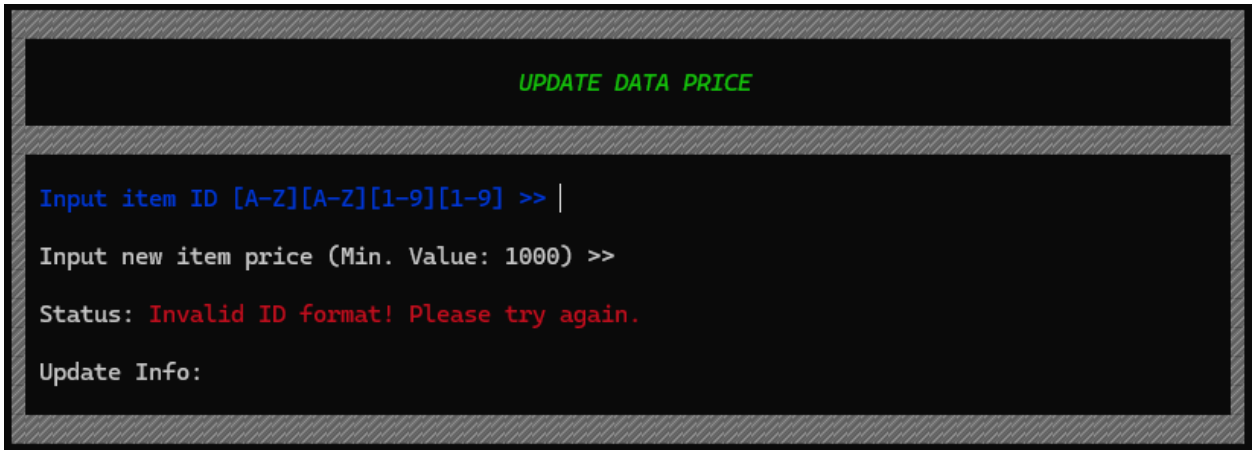
11.4 Tampilan Menu Delete Stock Jika Berhasil Mengurangi Stock Barang

11.5 Tampilan Menu Delete Stock Jika Berhasil Menghapus Barang

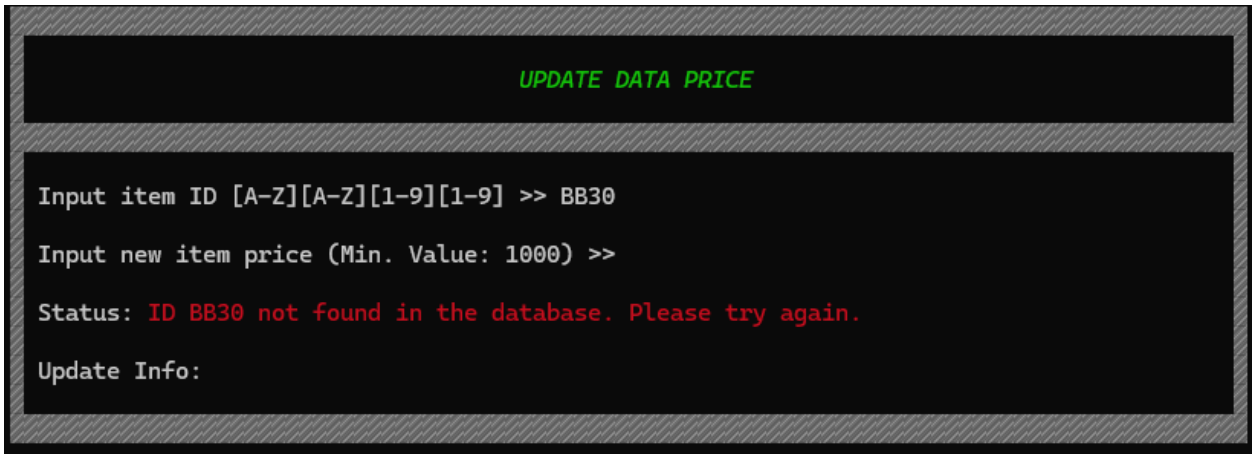


12.0 Tampilan Menu Awal Update Data Menu

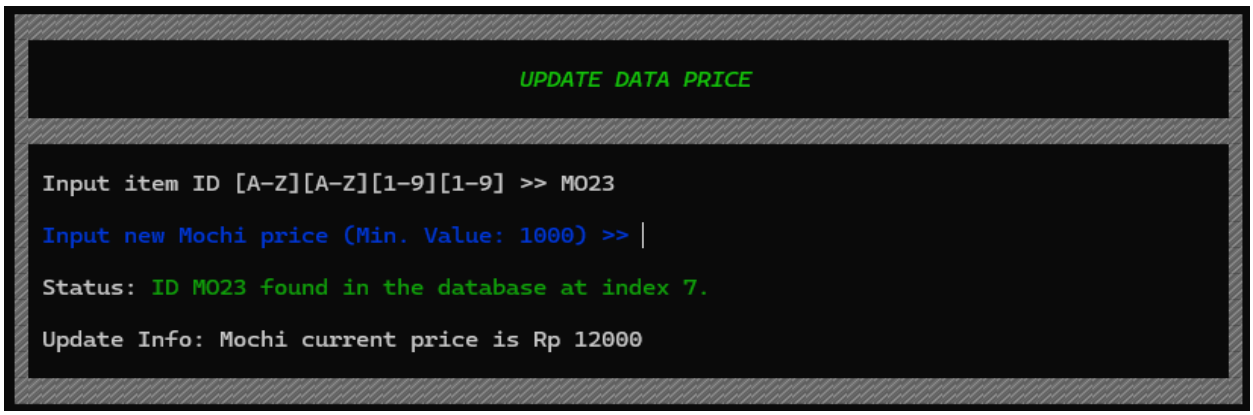
13.0 Tampilan Menu Awal Update Data Price



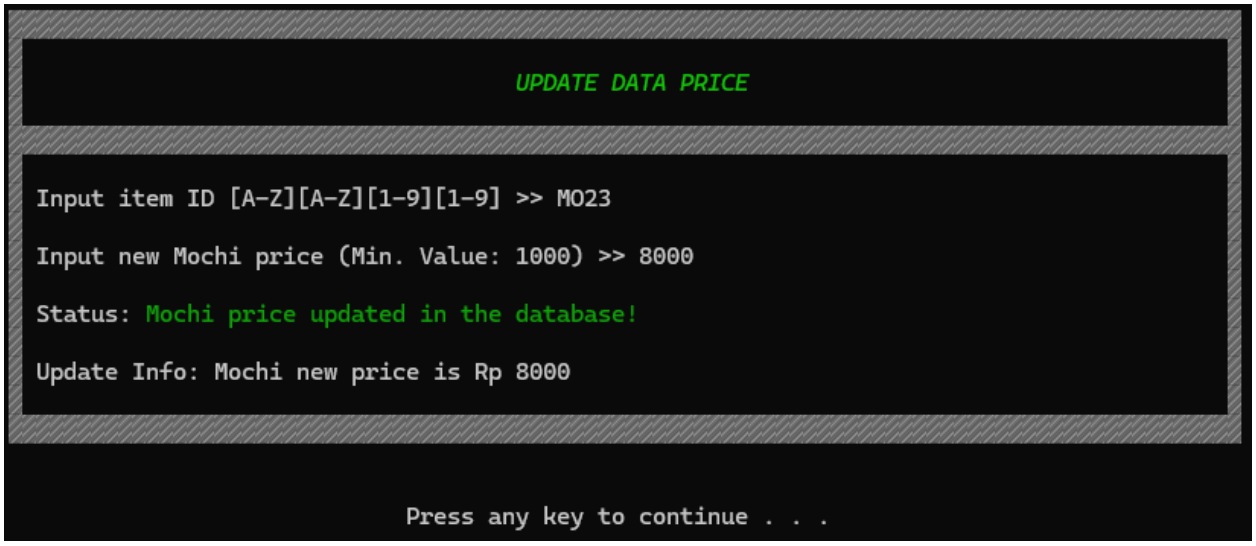
13.1 Tampilan Update Data Price Jika Salah Memasukkan Format



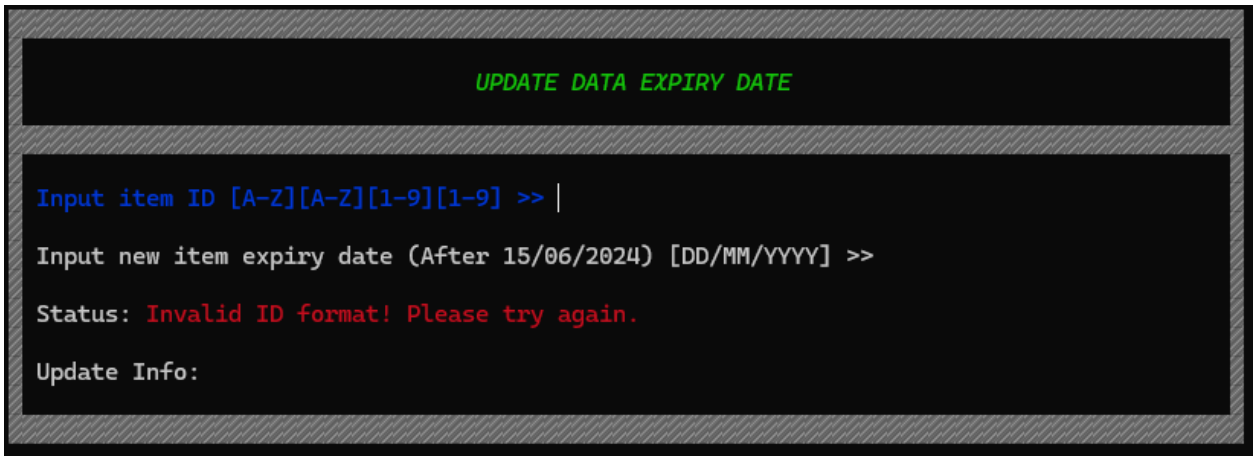
13.2 Tampilan Update Data Price Jika Tidak Menemukan ID



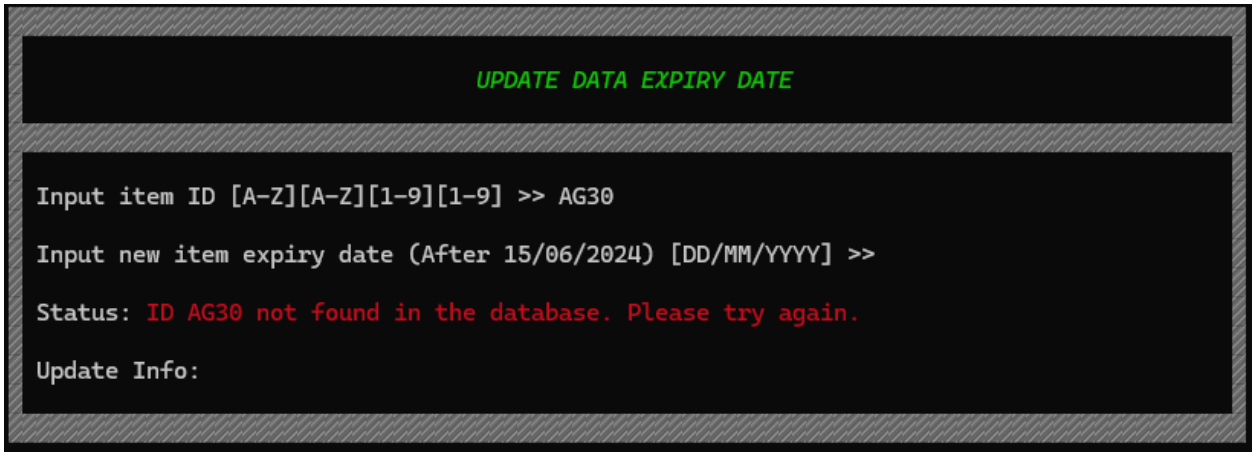
13.3 Tampilan Update Data Price Jika Berhasil Menemukan ID



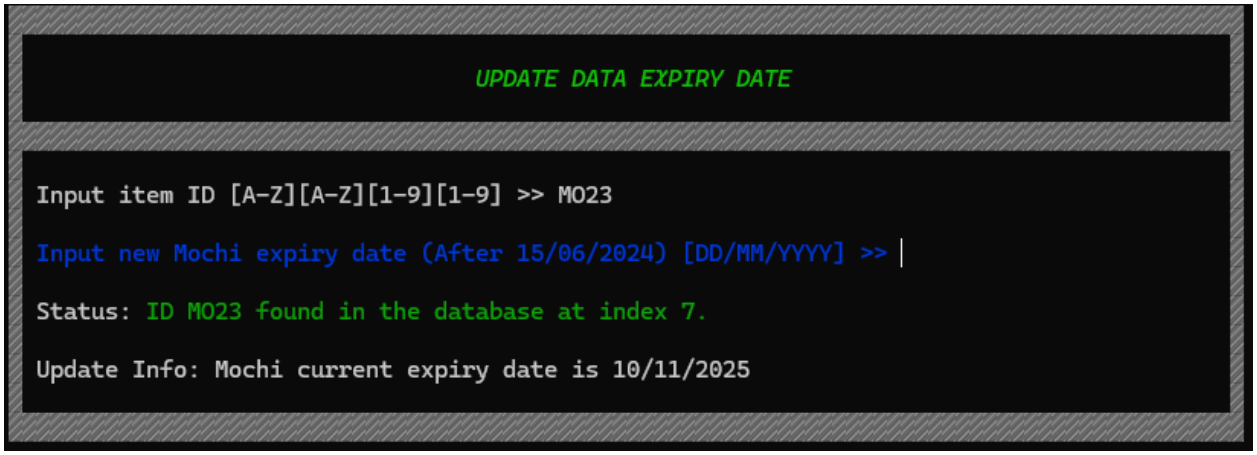
13.4 Tampilan Update Data Price Jika Berhasil Mengubah Harga



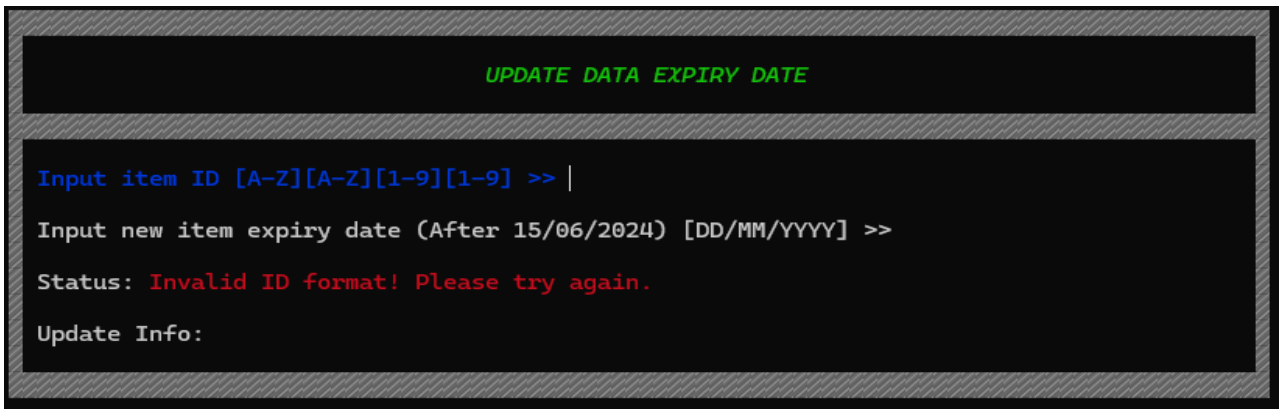
14.1 Tampilan Awal Update Data Expiry Date Jika Salah Memasukkan Format



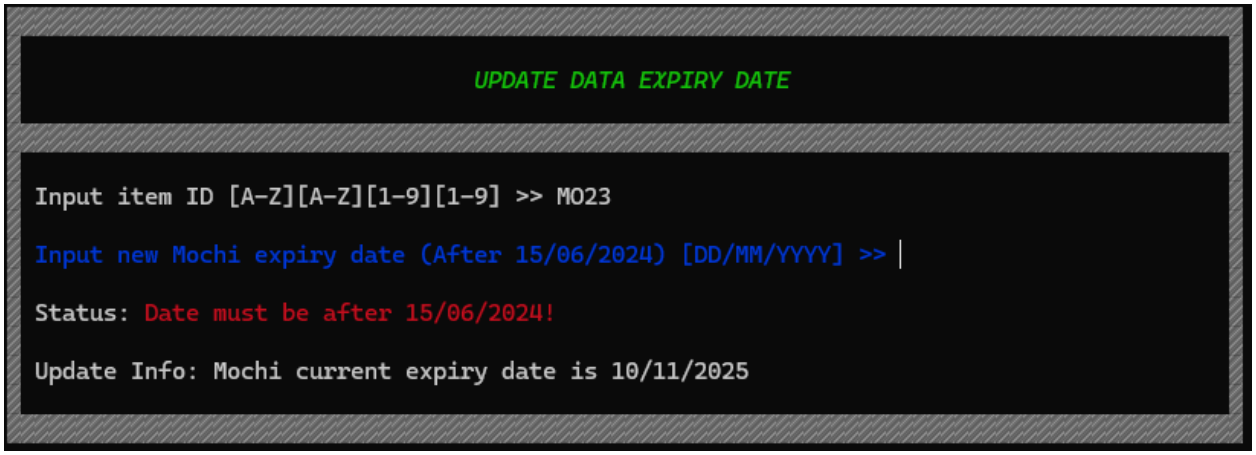
14.2 Tampilan Update Data Expiry Date Jika ID Tidak Ditemukan



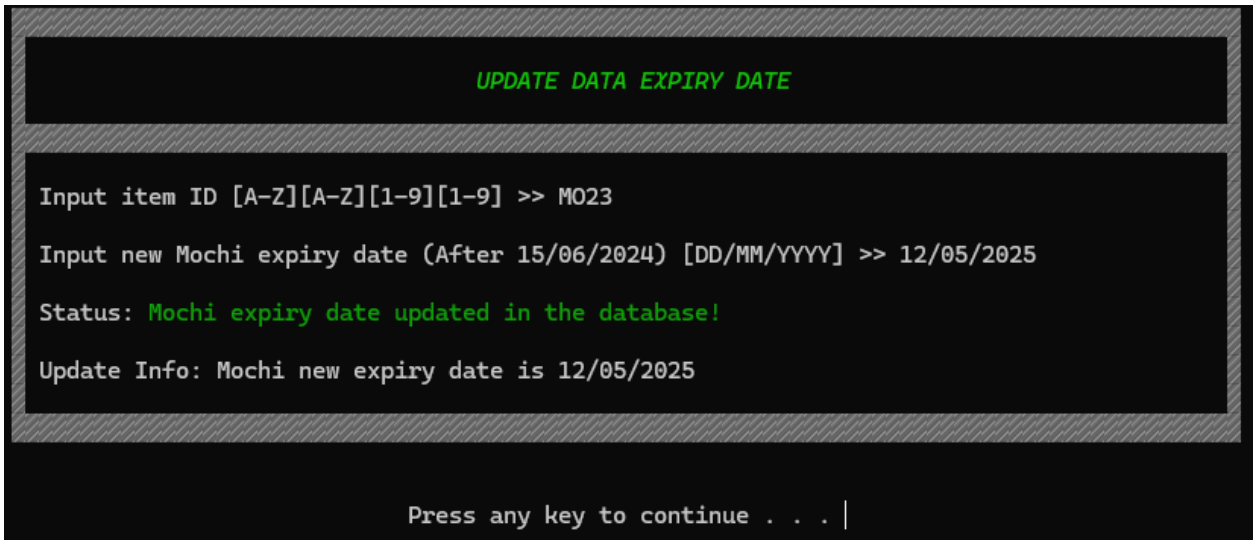
14.3 Tampilan Update Expiry Date Jika Berhasil Menemukan ID



14.4 Tampilan Update Expiry Date Jika Salah Memasukkan Format



14.4 Tampilan Update Expiry Date Jika Salah Memasukkan Tanggal



14.5 Tampilan Expiry Date Jika Berhasil Mengupdate Tanggal Expired

15.0 Exit Menu

- **Program Code**

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <string.h>
#include <time.h>
#include <conio.h>
#include <math.h>

#define RESET "\x1b[0m"
#define RED "\x1b[31m"
#define GREEN "\x1b[32m"
#define BLUE "\x1b[34m"
#define BOLD "\x1b[1m"
#define ITALIC "\x1b[3m"
#define CLEAR_LINE "\x1b[2K"
#define TABLESIZE 50

// Struct untuk menyimpan data hashing
struct hashTable {
    char itemID[5];
    char itemName[15];
    int stock;
    long int price;
    char expiryDate[15];
    struct hashTable *next;
};

struct data {
    char itemID[5];
    char itemName[15];
```

```
    int stock;
    long int price;
    char expiryDate[15];
};

struct date {
    int day;
    int month;
    int year;
};

struct hashTable *table[TABLESIZE];

// Metode hashing division
// Rumus => Setiap huruf dari id (ASCII) dijumlah lalu dibagi table size
int hashFunction(char id[5]) {
    int index = 0;
    for(int i = 0; i < strlen(id); i++) {
        index += id[i];
    }
    return index % TABLESIZE;
}

struct hashTable *createNode(char id[5], char name[15], int productStock, long
int productPrice, char expiry[15]) {
    struct hashTable *newNode = (struct hashTable*)malloc(sizeof(struct
hashTable));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    strcpy(newNode->itemID, id);
    strcpy(newNode->itemName, name);
    newNode->stock = productStock;
    newNode->price = productPrice;
    strcpy(newNode->expiryDate, expiry);
    newNode->next = NULL;
    return newNode;
}

void insert(char id[5], char name[15], int productStock, long int productPrice,
char expiry[15]) {
    int index = hashFunction(id);
    struct hashTable *newNode = createNode(id, name, productStock, productPrice,
expiry);

    newNode->next = table[index];
    table[index] = newNode;
}

void extractFile(int *totalProduct) {
    FILE *file = fopen("Database/database.txt", "r");
    if (file == NULL) {
        printf("Failed to open file\n");
        exit(1);
    }

    char line[150];
    while (fgets(line, sizeof(line), file)) {
        char id[5];
        char name[15];
        int stock;
        long int price;
```

```
        char expiry[15];

        if (sscanf(line, "%[^,],%[^,],%d,%ld,%[^\n]\n", id, name, &stock, &price,
expiry) == 5) {
            insert(id, name, stock, price, expiry);
            *totalProduct = *totalProduct + 1;
        } else {
            printf("Failed to parse line.\n");
        }
    }

    fclose(file);
}

void saveToFile() {
    FILE *file = fopen("Database/database.txt", "w");
    if (file) {
        for (int i = 0; i < TABLESIZE; i++) {
            struct hashTable *node = table[i];
            while (node) {
                fprintf(file, "%s,%s,%d,%ld,%s\n", node->itemID, node->itemName,
node->stock, node->price, node->expiryDate);
                node = node->next;
            }
        }
        fclose(file);
    }
}

void appendToFile(struct data purchase[20], int totalPurchase) {
    FILE *file = fopen("Database/cashflow.txt", "a");
    if (file) {
        for (int i = 0; i < totalPurchase; i++) {
            fprintf(file, "%s,%s,%d,%ld,%s\n", purchase[i].itemID,
purchase[i].itemName, purchase[i].stock, purchase[i].price,
purchase[i].expiryDate);
        }
        fclose(file);
    }
}

void importCashflow(struct data cashflow[100], int *totalProduct) {
    FILE *file = fopen("Database/cashflow.txt", "r");
    if (file == NULL) {
        printf("Failed to open file\n");
        exit(1);
    }

    char line[150];
    while (fgets(line, sizeof(line), file)) {
        char id[5];
        char name[15];
        int quantity;
        long int price;
        char buyDate[15];

        if (sscanf(line, "%[^,],%[^,],%d,%ld,%[^\n]\n", id, name, &quantity,
&price, buyDate) == 5) {
            strcpy(cashflow[*totalProduct].itemID, id);
            strcpy(cashflow[*totalProduct].itemName, name);
            strcpy(cashflow[*totalProduct].expiryDate, buyDate);
            cashflow[*totalProduct].stock = quantity;
            cashflow[*totalProduct].price = price;
```

```
            *totalProduct = *totalProduct + 1;
        } else {
            printf("Failed to parse line.\n");
        }
    }

    fclose(file);
}

// Fungsi menentukan position kursor
void cursorPositionCenter(int lineNumber, int textWidth) {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(hConsole, &csbi);

    int consoleWidth = csbi.dwSize.X;

    // Calculate the center position
    COORD coord = { (consoleWidth - textWidth) / 2, lineNumber };
    SetConsoleCursorPosition(hConsole, coord);
}

void clearHash() {
    for (int i = 0; i < TABLESIZE; i++) {
        table[i] = NULL;
    }
}

void clearInputBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}

// Fungsi menentukan tanggal saat ini
void today() {
    int day, month, year;
    time_t now;
    time(&now);

    struct tm *local = localtime(&now);

    day = local->tm_mday;
    month = local->tm_mon + 1;
    year = local->tm_year + 1900;

    printf("Today Date : %02d/%02d/%d\n\n", day, month, year);
}

void todayDate(char dateNow[15]) {
    int day, month, year;
    time_t now;
    time(&now);

    struct tm *local = localtime(&now);

    day = local->tm_mday;
    month = local->tm_mon + 1;
    year = local->tm_year + 1900;

    sprintf(dateNow, "%02d/%02d/%04d", day, month, year);
}

struct date parseDate(char *dateStr) {
```

```
    struct date tanggal;
    sscanf(dateStr, "%d/%d/%d", &tanggal.day, &tanggal.month, &tanggal.year);
    return tanggal;
}

int compareDates(struct date date1, struct date date2) {
    if (date1.year != date2.year) {
        return date1.year - date2.year;
    }
    if (date1.month != date2.month) {
        return date1.month - date2.month;
    }
    return date1.day - date2.day;
}

void lowercase(char name[15]) {
    for(int i = 0; i < strlen(name); i++) {
        name[i] =  tolower(name[i]);
    }
}

int isDigits(const char *str) {
    while (*str) {
        if (!isdigit(*str)) {
            return 0;
        }
        str++;
    }
    return 1;
}

int checkAvailability(char id[5]) {
    for (int i = 0; i < TABLESIZE; i++) {
        struct hashTable* node = table[i];
        while (node) {
            if (strcmp(node->itemID, id) == 0) {
                return 1;
            }
            node = node->next;
        }
    }
    return 0;
}

// Rumus membuat ID:
// 2 kata pertama: 1) Jika 2 kata, maka huruf pertama dari setiap kata
//                 2) Jika 1 kata, maka 2 huruf pertama dari kata tersebut
// 2 angka terakhir: Panjang nama produk + 20
// Jika ID sudah terpakai, maka angka + 1 sampai dapat ID yang masih kosong
void generateId(char *input, char *id) {
    id[0] = '\0';

    if (input == NULL || strlen(input) == 0) {
        return;
    }

    char inputCopy[100];
    strncpy(inputCopy, input, 100);
    inputCopy[99] = '\0';

    char *firstWord = strtok(inputCopy, " ");
    char *secondWord = strtok(NULL, " ");
```

```
        if (secondWord != NULL) {
            id[0] = toupper(firstWord[0]);
            id[1] = toupper(secondWord[0]);
        } else {
            id[0] = toupper(firstWord[0]);
            id[1] = toupper(firstWord[1]);
        }

        int length = 0;
        for (int i = 0; i < strlen(input); i++) {
            length++;
        }

        int lastTwoDigits = length + 20;

        sprintf(id + 2, "%02d", lastTwoDigits);
        id[4] = '\0';

        while (checkAvailability(id)) {
            lastTwoDigits++;
            sprintf(id + 2, "%02d", lastTwoDigits);
        }
    }

    int validateID(const char *id) {
        if (strlen(id) != 4) {
            return 0;
        }

        if (!(id[0] >= 'A' && id[0] <= 'Z' && id[1] >= 'A' && id[1] <= 'Z')) {
            return 0;
        }

        if (!(id[2] >= '0' && id[2] <= '9' && id[3] >= '0' && id[3] <= '9')) {
            return 0;
        }

        return 1;
    }

    int validateName(char name[15]) {
        if (strlen(name) < 3 || strlen(name) > 15) {
            cursorPositionCenter(25, 92);
            printf("\xb2
    \xb2\n");
            cursorPositionCenter(25, 88);
            printf("Status: " RED "Invalid name format!" RESET "\n");
            return 0;
        }

        int found = 0;

        for(int i = 0; i < TABLESIZE; i++) {
            struct hashTable *current = table[i];
            while (current) {
                char name1[15], name2[15];
                strcpy(name1, current->itemName);
                strcpy(name2, name);
                lowercase(name1);
                lowercase(name2);

                if (strcmp(name1, name2) == 0) {
                    found = 1;
```

```
            }
            current = current->next;
        }
        if (found == 1) {
            cursorPositionCenter(25, 92);
            printf("\xb2
\xb2\n");
            cursorPositionCenter(25, 88);
            printf("Status: " RED "Product already existed!" RESET "\n");
            return 0;
        }
    }

    return 1;
}

int validateStock(int stock) {
    if(stock < 1 || stock > 100) {
        cursorPositionCenter(25, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(25, 88);
        printf("Status: " RED "Invalid stock format!" RESET "\n");
        return 0;
    }
    return 1;
}

int validatePrice(long int price, int status) {
    if(price < 1000) {
        cursorPositionCenter(status, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(status, 88);
        printf("Status: " RED "Invalid price format!" RESET "\n");
        return 0;
    }
    return 1;
}

int validateExpDate(char expDate[15], int status) {
    if (strlen(expDate) != 10) {
        cursorPositionCenter(status, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(status, 88);
        printf("Status: " RED "Invalid date format!" RESET "\n");
        return 0;
    }

    if (expDate[2] != '/' || expDate[5] != '/') {
        cursorPositionCenter(status, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(status, 88);
        printf("Status: " RED "Invalid date format!" RESET "\n");
        return 0;
    }

    char day[3] = { expDate[0], expDate[1], '\0' };
    char month[3] = { expDate[3], expDate[4], '\0' };
    char year[5] = { expDate[6], expDate[7], expDate[8], expDate[9], '\0' };
```

```
    if (!isDigits(day) || !isDigits(month) || !isDigits(year)) {
        cursorPositionCenter(status, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(status, 88);
        printf("Status: " RED "Invalid date format!" RESET "\n");
        return 0;
    }

    int dayInt = atoi(day);
    int monthInt = atoi(month);
    int yearInt = atoi(year);

    if (dayInt < 1 || dayInt > 31 || monthInt < 1 || monthInt > 12 || yearInt <
1) {
        cursorPositionCenter(status, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(status, 88);
        printf("Status: " RED "Invalid date format!" RESET "\n");
        return 0;
    }

    if ((monthInt == 4 || monthInt == 6 || monthInt == 9 || monthInt == 11) &&
dayInt > 30) {
        cursorPositionCenter(status, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(status, 88);
        printf("Status: " RED "Invalid date format!" RESET"\n");
        return 0;
    }
    if (monthInt == 2) {
        int isLeap = (yearInt % 4 == 0 && (yearInt % 100 != 0 || yearInt % 400 ==
0));
        if (dayInt > 29 || (dayInt == 29 && !isLeap)) {
            cursorPositionCenter(status, 92);
            printf("\xb2
\xb2\n");
            cursorPositionCenter(status, 88);
            printf("Status: " RED "Invalid date format!" RESET "\n");
            return 0;
        }
    }

    char dateNow[15];
    todayDate(dateNow);

    struct date now = parseDate(dateNow);
    struct date expiry = parseDate(expDate);

    int comparisonResult = compareDates(now, expiry);

    if (comparisonResult >= 0) {
        cursorPositionCenter(status, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(status, 88);
        printf("Status: " RED "Date must be after %s!" RESET "\n", dateNow);
        return 0;
    }

    return 1;
```

```
}

// Mencetak page introduction
void intro() {
    system("cls");

    char welcome[47] = "WELCOME TO CASHIER AND WAREHOUSE MANAGEMENT APP";
    char dev[15] = "Developed By:";
    char team1[46] = "Kevin Joseph Handoyo             - 2702355302";
    char team2[46] = "Muhammad Ibraahiim Putra Wahana  - 2702350106";

    cursorPositionCenter(8, 54);
    for (int i = 0; i < 55; i++) {
        Sleep(10);
        printf("\xb2");
    }

    for(int i = 9; i < 17; i++) {
        cursorPositionCenter(i, 54);
        Sleep(20);
        printf("\xb2");
    }

    for(int i = 9; i < 17; i++) {
        cursorPositionCenter(i, -54);
        Sleep(20);
        printf("\xb2");
    }

    cursorPositionCenter(16, 52);
    for (int i = 0; i < 53; i++) {
        Sleep(10);
        printf("\xb2");
    }

    Sleep(100);
    cursorPositionCenter(10, 46);
    printf(BOLD);
    for(int i = 0; i < strlen(welcome); i++) {
        Sleep(10);
        printf("%c", welcome[i]);
    }
    printf(RESET "\n\n");

    Sleep(100);
    cursorPositionCenter(12, 12);
    printf(BOLD ITALIC);
    for(int i = 0; i < strlen(dev); i++) {
        Sleep(10);
        printf("%c", dev[i]);
    }
    printf(RESET "\n");

    Sleep(100);
    cursorPositionCenter(13, 44);
    printf(BOLD);
    for(int i = 0; i < strlen(team1); i++) {
        Sleep(10);
        printf("%c", team1[i]);
    }
    cursorPositionCenter(14, 44);
    for(int i = 0; i < strlen(team2); i++) {
        Sleep(10);
```

```
            printf("%c", team2[i]);
        }
        printf(RESET "\n");

        Sleep(1500);
    }

    // Print menu login
    void printLoginMenu(int option) {
        cursorPositionCenter(8, 42);
        for(int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(9, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(10, 42);
        printf("\xb2        " BOLD "Welcome to " GREEN ITALIC "LengkapMart" RESET "
    \xb2\n");
        cursorPositionCenter(11, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(12, 42);
        for(int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(13, 42);
        printf("\xb2                                        \xb2\n");

        if(option == 1) {
            cursorPositionCenter(14, 42);
            printf("\xb2" BLUE "  -> 1. Login as Cashier          <-  " RESET
    "\xb2\n");
            cursorPositionCenter(15, 42);
            printf("\xb2 2. Login as Warehouse Staff          \xb2\n");
            cursorPositionCenter(16, 42);
            printf("\xb2 3. Exit                              \xb2\n");
        } else if(option == 2) {
            cursorPositionCenter(14, 42);
            printf("\xb2 1. Login as Cashier                  \xb2\n");
            cursorPositionCenter(15, 42);
            printf("\xb2" BLUE "  -> 2. Login as Warehouse Staff  <-  " RESET
    "\xb2\n");
            cursorPositionCenter(16, 42);
            printf("\xb2 3. Exit                              \xb2\n");
        } else if(option == 3) {
            cursorPositionCenter(14, 42);
            printf("\xb2 1. Login as Cashier                  \xb2\n");
            cursorPositionCenter(15, 42);
            printf("\xb2 2. Login as Warehouse Staff          \xb2\n");
            cursorPositionCenter(16, 42);
            printf("\xb2" BLUE "  -> 3. Exit                      <-  " RESET
    "\xb2\n");
        }

        cursorPositionCenter(17, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(18, 42);
        for(int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(20, 63);
        printf("Use arrow keys to navigate and press Enter to select an option.\n");
    }
```

```
// Menu awal untuk login
int loginPage() {
    int option = 1;
    int ch;

    while (1) {
        system("cls");
        today();
        printLoginMenu(option);

        ch = _getch();
        if (ch == 0 || ch == 224) {
            ch = _getch();
            switch (ch) {
                case 72: // Up
                    if (option > 1) option--;
                    break;
                case 80: // Down
                    if (option < 3) option++;
                    break;
                default:
                    break;
            }
        } else if (ch == 13) {
            // Enter
            return option;
        }
    }
}

// Input password dari user
void getPassword(char* password) {
    char ch;
    int i = 0;

    while ((ch = _getch()) != '\r') {
        if (ch == '\b') {
            if (i > 0) {
                printf("\b \b");
                i--;
            }
        } else {
            printf("*");
            password[i++] = ch;
        }
    }
    password[i] = '\0';
    printf("\n");
}

// Fungsi untuk login
int login(int try) {
    system("cls");
    today();

    char username[15], password[15];

    cursorPositionCenter(8, 42);
    for(int i = 0; i < 40; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(9, 42);
    printf("\xb2                                        \xb2\n");
```

```
    cursorPositionCenter(10, 42);
    printf("\xb2              " BOLD GREEN ITALIC "LOGIN PAGE" RESET "
\xb2\n");
    cursorPositionCenter(11, 42);
    printf("\xb2                                          \xb2\n");
    cursorPositionCenter(12, 42);
    for(int i = 0; i < 40; i++) {
        printf("\xb2");
    }
    for(int i = 13; i < 18; i++) {
        cursorPositionCenter(i, 42);
        printf("\xb2");
    }
    for(int i = 13; i < 18; i++) {
        cursorPositionCenter(i, -36);
        printf("\xb2");
    }
    cursorPositionCenter(18, 42);
    for(int i = 0; i < 40; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(14, 34);
    printf("Username >> ");
    cursorPositionCenter(16, 34);
    printf("Password >> ");

    cursorPositionCenter(14, 10);
    scanf(" %[^\n]", username);
    cursorPositionCenter(16, 10);
    getPassword(password);

    if(strcmp(username, "admin") != 0 || strcmp(password, "admin") != 0) {
        if(try == 0) return 0;
        cursorPositionCenter(20, 66);
        printf(RED "Invalid username or password! Please try again (%d more
tries/try)" RESET "\n", try);
        try--;
        cursorPositionCenter(22, 34);
        system("pause");
        login(try);
    } else {
        return 1;
    }
}

void printProductOverview(int product) {
    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);

    struct data data[totalProduct]; // Buat array untuk menyimpan data dari hash
table
    int j = 0;

    // Memindahkan data dari hash table ke dalam array struct
    for (int i = 0; i < TABLESIZE; i++) {
        struct hashTable *current = table[i];
        while (current != NULL) {
            strcpy(data[j].itemID, current->itemID);
            strcpy(data[j].itemName, current->itemName);
            strcpy(data[j].expiryDate, current->expiryDate);
            data[j].price = current->price;
```

```
            data[j].stock = current->stock;
            j++;
            current = current->next;
        }
    }

    // Menampilkan data dari array struct
    system("cls");
    today();
    cursorPositionCenter(2, 20);
    printf("PRODUCT OVERVIEW\n");
    cursorPositionCenter(3, 90);
    for(int i = 0; i < 88; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(4, 90);
    printf("\xb2 %-4s \xb2 %-5s \xb2 %-20s \xb2 %-10s \xb2 %-15s \xb2 %-15s
\xb2\n", "No.", "ID", "Product Name", "Stock", "Price", "Expiry Date");
    cursorPositionCenter(5, 90);
    for(int i = 0; i < 88; i++) {
        printf("\xb2");
    }

    int dataShown = 0;
    int count = 0;

    if(product != 1) {
        dataShown = product * 10 - 10;
    }

    // Menampilkan data dari array struct
    for (int i = dataShown; i < totalProduct; i++) {
        cursorPositionCenter(6 + count, 90);
        printf("\xb2 %-4d \xb2 %-5s \xb2 %-20s \xb2 %-10d \xb2 Rp %-12ld \xb2
%-15s \xb2\n", i+1, data[i].itemID, data[i].itemName, data[i].stock,
data[i].price, data[i].expiryDate);
        dataShown++;
        count++;
        if (dataShown % 10 == 0) break;
    }

    cursorPositionCenter(6 + count, 90);
    for(int i = 0; i < 88; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(8 + count , 58);
    printf("Use arrow keys (" GREEN "Left" RESET " / " GREEN "Right" RESET ") to
view the products list.\n");
}

// Print menu kasir
void printCashierMenu(int option) {
    cursorPositionCenter(20, 42);
    for(int i = 0; i < 40; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(21, 42);
    printf("\xb2                                        \xb2\n");
    cursorPositionCenter(22, 42);
    printf("\xb2             " BOLD GREEN ITALIC "CASHIER MENU" RESET "
\xb2\n");
    cursorPositionCenter(23, 42);
```

```
        printf("\xb2                                          \xb2\n");
        cursorPositionCenter(24, 42);
        for(int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(25, 42);
        printf("\xb2                                          \xb2\n");

        if(option == 1) {
            cursorPositionCenter(26, 42);
            printf("\xb2" BLUE "   -> 1. Purchase               <-   " RESET
    "\xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Cashflow                        \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Hash-Table View                 \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Check Expiration                \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Logout                          \xb2\n");
        } else if(option == 2) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Purchase                        \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2" BLUE "   -> 2. Cashflow               <-   " RESET
    "\xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Hash-Table View                 \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Check Expiration                \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Logout                          \xb2\n");
        } else if(option == 3) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Purchase                        \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Cashflow                        \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2" BLUE "   -> 3. Hash-Table View         <-   " RESET
    "\xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Check Expiration                \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Logout                          \xb2\n");
        } else if(option == 4) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Purchase                        \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Cashflow                        \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Hash-Table View                 \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2" BLUE "   -> 4. Check Expiration        <-   " RESET
    "\xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Logout                          \xb2\n");
        } else if(option == 5) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Purchase                        \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Cashflow                        \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Hash-Table View                 \xb2\n");
```

```
        cursorPositionCenter(29, 42);
        printf("\xb2   4. Check Expiration                \xb2\n");
        cursorPositionCenter(30, 42);
        printf("\xb2" BLUE "   -> 5. Logout                   <-   " RESET
"\xb2\n");
    }

    cursorPositionCenter(31, 42);
    printf("\xb2                                        \xb2\n");
    cursorPositionCenter(32, 42);
    for(int i = 0; i < 40; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(34, 76);
    printf("Use arrow keys (" GREEN "Up" RESET " / " GREEN "Down" RESET ") to
navigate and press " GREEN "Enter" RESET " to select an option.\n");
}

// Menu pada bagian kasir
void cashierMenu(int *opt) {
    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);
    int option = 1;
    int product = 1;
    int ch;

    while (1) {
        system("cls");
        today();
        printProductOverview(product);
        printCashierMenu(option);

        ch = _getch();
        if (ch == 0 || ch == 224) {
            ch = _getch();
            switch (ch) {
                case 72: // Up
                    if(option > 1) option--;
                    break;
                case 80: // Down
                    if(option < 5) option++;
                    break;
                case 75: // Left
                    if(product > 1) product--;
                    break;
                case 77: // Right
                    if((float)product < ((float)totalProduct / 10)) product++;
                    break;
                default:
                    break;
            }
        } else if (ch == 13) {
            // Enter
            *opt = option;
            return;
        }
    }
}

void purchaseProduct(struct data purchase[20], int *totalPurchase) {
    while(1) {
        system("cls");
```

```
        today();

        cursorPositionCenter(8, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(9, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(10, 92);
        printf("\xb2                                        " BOLD  GREEN ITALIC
"PURCHASE PRODUCT" RESET "                                        \xb2\n");
        cursorPositionCenter(11, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(12, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }
        for (int i = 0; i < 14; i++) {
            cursorPositionCenter(13 + i, 92);
            printf("\xb2\n");
        }
        for (int i = 0; i < 14; i++) {
            cursorPositionCenter(13 + i, -86);
            printf("\xb2\n");
        }
        cursorPositionCenter(27, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }

        int totalProduct = 0;
        clearHash();
        extractFile(&totalProduct);
        char id[5];

        cursorPositionCenter(16, 88);
        printf("Input total stock to purchase [1-XX] >> ");

        cursorPositionCenter(18, 88);
        printf("Status: ");

        cursorPositionCenter(20, 88);
        printf("Purchase Info: ");

        while (1) {
            cursorPositionCenter(14, 88);
            printf("Input item ID [A-Z][A-Z][1-9][1-9] >> ");
            cursorPositionCenter(14, 12);
            scanf(" %[^\n]", id);

            clearInputBuffer();

            if (validateID(id)) {
                break;
            }

            cursorPositionCenter(18, 88);
            printf(CLEAR_LINE);
            cursorPositionCenter(18, 92);
            printf("\xb2
\xb2\n");
```

```
            cursorPositionCenter(18, 88);
            printf("Status: ");
            cursorPositionCenter(18, 72);
            printf(RED "Invalid ID format! Please try again." RESET "\n");
        }

        int foundAt = -1;

        for(int i = 0; i < TABLESIZE; i++) {
            struct hashTable *node = table[i];
            while (node) {
                if (strcmp(node->itemID, id) == 0) {
                    foundAt = i; // Ditemukan
                }
                node = node->next;
            }
            if (foundAt != -1) break;
        }

        if (foundAt != -1) {
            int deleteStock;
            cursorPositionCenter(18, 72);
            printf("ID %s found in the database at index %d.\n", id, foundAt);
            struct hashTable *current = table[foundAt];
            while(strcmp(current->itemID, id) != 0) {
                current = current->next;
            }

            while(1) {
                cursorPositionCenter(16, 88);
                printf(CLEAR_LINE);
                cursorPositionCenter(16, 92);
                printf("\xb2
\xb2\n");

                cursorPositionCenter(16, 88);
                printf("Input total stock to purchase [1-%d] >> ",
current->stock);
                scanf(" %d", &deleteStock);

                clearInputBuffer();

                if(deleteStock > 0 && deleteStock <= current->stock) {
                    cursorPositionCenter(18, 88);
                    printf(CLEAR_LINE);
                    cursorPositionCenter(18, 92);
                    printf("\xb2
\xb2\n");
                    cursorPositionCenter(18, 88);
                    printf("Status: ");
                    cursorPositionCenter(18, 72);
                    printf("ID %s found in the database at index %d.\n", id,
foundAt);
                    break;
                }

                cursorPositionCenter(18, 88);
                printf(CLEAR_LINE);
                cursorPositionCenter(18, 92);
                printf("\xb2
\xb2\n");
                cursorPositionCenter(18, 88);
                printf("Status: ");
```

```
                cursorPositionCenter(18, 72);
                printf(RED "Invalid stock! Please try again." RESET "\n");
            }

            char date[15];
            todayDate(date);

            current->stock -= deleteStock;

            if(current->stock <= 0) {
                strcpy(purchase[*totalPurchase].itemID, current->itemID);
                strcpy(purchase[*totalPurchase].itemName, current->itemName);
                purchase[*totalPurchase].price = current->price;
                purchase[*totalPurchase].stock = deleteStock;
                char dateNow[15];
                todayDate(dateNow);
                strcpy(purchase[*totalPurchase].expiryDate, dateNow);

                struct hashTable *node = table[foundAt];
                struct hashTable *prev = NULL;

                while (node != NULL && strcmp(node->itemID, id) != 0) {
                    prev = node;
                    node = node->next;
                }

                if (node == NULL) {
                    return;
                }

                if (prev == NULL) {
                    table[foundAt] = node->next;
                } else {
                    prev->next = node->next;
                }

                free(node);
                cursorPositionCenter(20, 56);
                printf("%s purchased successfully on %s!\n", current->itemName,
date);
                cursorPositionCenter(21, 56);
                printf("New stock: Empty! %s deleted from database
succesfully.\n", current->itemName);
            } else {
                strcpy(purchase[*totalPurchase].itemID, current->itemID);
                strcpy(purchase[*totalPurchase].itemName, current->itemName);
                purchase[*totalPurchase].price = current->price;
                purchase[*totalPurchase].stock = deleteStock;
                char dateNow[15];
                todayDate(dateNow);
                strcpy(purchase[*totalPurchase].expiryDate, dateNow);

                cursorPositionCenter(20, 56);
                printf("%s purchased successfully on %s!\n", current->itemName,
date);
                cursorPositionCenter(21, 56);
                printf("New stock: %d\n", current->stock);
            }

            *totalPurchase = *totalPurchase + 1;

            saveToFile();
```

```
            int option = 1;
            int ch;

            cursorPositionCenter(23, 88);
            printf("Do you want to purchase more products?\n");
            while (1) {
                if(option == 1) {
                    cursorPositionCenter(24, 88);
                    printf(BOLD BLUE "[ Yes ]" RESET "\n");
                    cursorPositionCenter(25, 88);
                    printf("[ No ]\n");
                } else if(option == 2) {
                    cursorPositionCenter(24, 88);
                    printf("[ Yes ]\n");
                    cursorPositionCenter(25, 88);
                    printf(BOLD BLUE "[ No ]" RESET "\n");
                }

                ch = _getch();
                if (ch == 0 || ch == 224) {
                    ch = _getch();
                    switch (ch) {
                        case 72: // Up
                            if (option > 1) option--;
                            break;
                        case 80: // Down
                            if (option < 2) option++;
                            break;
                        default:
                            break;
                    }
                } else if (ch == 13) {
                    // Enter
                    if(option != 1) {
                        return;
                    } else break;
                }
            }
        } else {
            cursorPositionCenter(18, 72);
            printf("ID %s not found in the database. Please try again.\n", id);
            cursorPositionCenter(29, 34);
            system("pause");
            break;
        }
    }
}

void purchaseSummary(struct data purchase[20], int totalPurchase) {
    system("cls");
    today();

    cursorPositionCenter(6, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(7, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(8, 92);
    printf("\xb2                                        " BOLD GREEN ITALIC "PURCHASE
SUMMARY" RESET "                                        \xb2\n");
    cursorPositionCenter(9, 92);
```

```
    printf("\xb2
\xb2\n");
    cursorPositionCenter(10, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(12, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(11, 92);
    printf("\xb2   ID   \xb2        Item Name        \xb2  Qty  \xb2       Price
\xb2       Total Price       \xb2\n");
    cursorPositionCenter(13, 92);
    printf("\xb2        \xb2                            \xb2       \xb2
\xb2                        \xb2\n");

    int grandTotal = 0;

    for(int i = 0; i < totalPurchase; i++) {
        cursorPositionCenter(14 + (i * 2), 92);
        printf("\xb2 [%-4s] \xb2 %-23s \xb2  %-3d  \xb2 Rp %-14ld \xb2 Rp %-20ld
\xb2\n", purchase[i].itemID, purchase[i].itemName, purchase[i].stock,
purchase[i].price, purchase[i].stock * purchase[i].price);
        cursorPositionCenter(15 + (i * 2), 92);
        printf("\xb2        \xb2                            \xb2       \xb2
\xb2                        \xb2\n");
        grandTotal += purchase[i].stock * purchase[i].price;
    }

    cursorPositionCenter(14 + (totalPurchase * 2), 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(15 + (totalPurchase * 2), -4);
    printf("\xb2                                          \xb2\n");
    cursorPositionCenter(16 + (totalPurchase * 2), -4);
    printf("\xb2  " BOLD GREEN "Grand Total : Rp %-20d " RESET "\xb2\n",
grandTotal);
    cursorPositionCenter(17 + (totalPurchase * 2), -4);
    printf("\xb2                                          \xb2\n");
    cursorPositionCenter(18 + (totalPurchase * 2), -4);
    for (int i = 0; i < 42; i++) {
        printf("\xb2");
    }

    appendToFile(purchase, totalPurchase);

    cursorPositionCenter(20 + (totalPurchase * 2), 34);
    system("pause");
}

void cashflow(int page) {
    cursorPositionCenter(2, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(3, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(4, 92);
```

```
    printf("\xb2                                                  " BOLD GREEN ITALIC
"CASHFLOW" RESET "                                                  \xb2\n");
    cursorPositionCenter(5, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(6, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(8, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(9, 92);
    printf("\xb2                   \xb2          \xb2
\xb2            \xb2                   \xb2\n");

    cursorPositionCenter(7, 92);
    printf("\xb2   Purchase Date   \xb2    ID    \xb2        Item Name
\xb2  Quantity  \xb2     Earnings     \xb2\n");

    int dataShown = 0;
    int count = 0;

    if(page != 1) {
        dataShown = page * 10 - 10;
    }

    struct data cashflow[100];
    int totalProduct = 0;
    int totalEarnings = 0;

    importCashflow(cashflow, &totalProduct);

    for(int i = 0; i < totalProduct; i++) {
        totalEarnings += cashflow[i].stock * cashflow[i].price;
    }

    for(int i = dataShown; i < totalProduct; i++) {
        cursorPositionCenter(10 + (count * 2), 92);
        printf("\xb2    %-10s     \xb2  [%-4s]  \xb2  %-20s   \xb2     %-3d
\xb2   Rp %-10d  \xb2\n", cashflow[i].expiryDate, cashflow[i].itemID,
cashflow[i].itemName, cashflow[i].stock, cashflow[i].stock * cashflow[i].price);
        cursorPositionCenter(11 + (count * 2), 92);
        printf("\xb2                    \xb2          \xb2
\xb2            \xb2                   \xb2\n");
        dataShown++;
        count++;
        if (dataShown % 10 == 0) break;
    }

    cursorPositionCenter(10 + (count * 2), 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(11 + (count * 2), -16);
    printf("\xb2                                    \xb2\n");
    cursorPositionCenter(12 + (count * 2), -16);
    printf("\xb2  " BOLD GREEN "Total Earnings : Rp %-11d " RESET "\xb2\n",
totalEarnings);
    cursorPositionCenter(13 + (count * 2), -16);
    printf("\xb2                                    \xb2\n");
```

```
    cursorPositionCenter(14 + (count * 2), -16);
    for (int i = 0; i < 36; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(11 + (count * 2), 92);
    printf("\xb2 Page %d of %.0f \xb2\n", page, ceil(((float)totalProduct) /
10));
    cursorPositionCenter(12 + (count * 2), 92);
    for(int i = 0; i < 15; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(17 + (count * 2), 96);
    printf("Use arrow keys (" GREEN "Left" RESET " / " GREEN "Right" RESET ") to
navigate through the pages and press " GREEN "Enter" RESET " to go back to
menu.\n");
}

void cashflowControl() {
    int page = 1;
    int ch;

    struct data check[100];
    int totalProduct = 0;

    importCashflow(check, &totalProduct);

    while (1) {
        system("cls");
        today();
        cashflow(page);

        ch = _getch();
        if (ch == 0 || ch == 224) {
            ch = _getch();
            switch (ch) {
                case 75: // Left
                    if(page > 1) page--;
                    break;
                case 77: // Right
                    if((float)page < ((float)totalProduct / 10)) page++;
                    break;
                default:
                    break;
            }
        } else if (ch == 13) {
            // Enter
            return;
        }
    }
}

void hashTableView(int product) {
    cursorPositionCenter(2, 132);
    for(int i = 0; i < 130; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(3, 132);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(4, 132);
```

```
    printf("\xb2                                                                    " BOLD
GREEN ITALIC  "Hash-Table" RESET BOLD " View" RESET "
\xb2\n");
    cursorPositionCenter(5, 132);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(6, 132);
    for(int i = 0; i < 130; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(7, 132);
    printf("\xb2 %-5s \xb2
Products Data                                                  \xb2\n",
"Index");
    cursorPositionCenter(8, 132);
    for(int i = 0; i < 130; i++) {
        printf("\xb2");
    }
    printf("\n");

    clearHash();
    int totalProduct = 0;
    extractFile(&totalProduct);

    if(product == 1) {
        for (int i = 0; i < (TABLESIZE / 2); i++) {
            cursorPositionCenter(9 + i, 132);
            printf("\xb2 [%-3d] \xb2 ", i);
            struct hashTable* current = table[i];
            while (current != NULL) {
                printf("(%s)%s -> ", current->itemID, current->itemName);
                current = current->next;
            }
            printf("NULL\n");
            cursorPositionCenter(9 + i, -126);
            printf("\xb2\n", i);
        }
        cursorPositionCenter(35, 132);
        printf("\xb2 Page %d of 2 \xb2\n", product);
    } else if(product == 2) {
        int position = 0;
        for (int i = (TABLESIZE / 2); i < TABLESIZE; i++) {
            cursorPositionCenter(9 + position, 132);
            printf("\xb2 [%-3d] \xb2 ", i);
            struct hashTable* current = table[i];
            while (current != NULL) {
                printf("(%s)%s -> ", current->itemID, current->itemName);
                current = current->next;
            }
            printf("NULL\n");
            cursorPositionCenter(9 + position, -126);
            printf("\xb2\n", i);
            position++;
        }
        cursorPositionCenter(35, 132);
        printf("\xb2 Page %d of 2 \xb2\n", product);
    }

    cursorPositionCenter(34, 132);
    for(int i = 0; i < 130; i++) {
        printf("\xb2");
    }
    printf("\n");
```

```
    cursorPositionCenter(36, 132);
    for(int i = 0; i < 15; i++) {
        printf("\xb2");
    }

    cursorPositionCenter(36, 84);
    printf("Use arrow keys (" GREEN "Left" RESET " / " GREEN "Right" RESET ") to
navigate through the pages and press " GREEN "Enter" RESET " to go back to
menu.\n");
}

void hashTableControl() {
    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);
    int product = 1;
    int ch;

    while (1) {
        system("cls");
        today();
        hashTableView(product);

        ch = _getch();
        if (ch == 0 || ch == 224) {
            ch = _getch();
            switch (ch) {
                case 75: // Left
                    if(product > 1) product--;
                    break;
                case 77: // Right
                    if(product < 2) product++;
                    break;
                default:
                    break;
            }
        } else if (ch == 13) {
            // Enter
            return;
        }
    }
}

void checkExpiration() {
    system("cls");
    today();

    cursorPositionCenter(8, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(9, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(10, 92);
    printf("\xb2                                          " BOLD GREEN ITALIC  "CHECK
EXPIRATION" RESET "                                          \xb2\n");
    cursorPositionCenter(11, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(12, 92);
    for (int i = 0; i < 90; i++) {
```

```
            printf("\xb2");
        }
        for (int i = 0; i < 8; i++) {
            cursorPositionCenter(13 + i, 92);
            printf("\xb2\n");
        }
        for (int i = 0; i < 8; i++) {
            cursorPositionCenter(13 + i, -86);
            printf("\xb2\n");
        }
        cursorPositionCenter(21, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }

        int totalProduct = 0;
        clearHash();
        extractFile(&totalProduct);
        char id[5];

        cursorPositionCenter(16, 88);
        printf("Status: ");

        cursorPositionCenter(18, 88);
        printf("Expiration Info: ");

        while (1) {
            cursorPositionCenter(14, 88);
            printf("Input item ID [A-Z][A-Z][1-9][1-9] >> ");
            cursorPositionCenter(14, 12);
            scanf(" %[^\n]", id);

            clearInputBuffer();

            if (validateID(id)) {
                break;
            }

            cursorPositionCenter(14, 88);
            printf(CLEAR_LINE);
            cursorPositionCenter(14, 92);
            printf("\xb2
    \xb2\n");
            cursorPositionCenter(16, 72);
            printf(RED "Invalid ID format! Please try again." RESET "\n");
        }

        int foundAt = -1;

        for(int i = 0; i < TABLESIZE; i++) {
            struct hashTable *node = table[i];
            while (node) {
                if (strcmp(node->itemID, id) == 0) {
                    foundAt = i; // Ditemukan
                }
                node = node->next;
            }
            if (foundAt != -1) break;
        }

        if (foundAt != -1) {
            char dateNow[15];
            cursorPositionCenter(16, 72);
```

```
            printf("ID %s found in the database at index %d.\n", id, foundAt);
            struct hashTable *current = table[foundAt];
            while(strcmp(current->itemID, id) != 0) {
                current = current->next;
            }
            todayDate(dateNow);
            struct date now = parseDate(dateNow);
            struct date product = parseDate(current->expiryDate);

            int comparisonResult = compareDates(now, product);

            if (comparisonResult < 0) {
                cursorPositionCenter(18, 54);
                printf(GREEN "%s is not yet expired." RESET "\n", current->itemName);
            } else if (comparisonResult > 0) {
                cursorPositionCenter(18, 54);
                printf(RED "%s is already expired." RESET "\n", current->itemName);
                cursorPositionCenter(19, 54);
                printf("Delete this product as soon as possible!\n");
            } else {
                cursorPositionCenter(18, 54);
                printf(BLUE "Today is the last day before %s expired." RESET "\n",
    current->itemName);
            }
        } else {
            cursorPositionCenter(16, 72);
            printf("ID %s not found in the database. Please try again.\n", id);
        }

        cursorPositionCenter(23, 34);
        system("pause");
    }

    // Fungsi utama untuk kasir
    void cashier() {
        system("cls");

        if(login(3) == 1) {
            int opt;
            while(1) {
                cashierMenu(&opt);

                if(opt == 1) {
                    struct data purchase[20];
                    int totalPurchase = 0;
                    purchaseProduct(purchase, &totalPurchase);
                    if(totalPurchase != 0) purchaseSummary(purchase, totalPurchase);
                } else if(opt == 2) {
                    cashflowControl();
                } else if(opt == 3) {
                    hashTableControl();
                } else if(opt == 4) {
                    checkExpiration();
                } else if(opt == 5) {
                    return;
                }
            }
        } else {
            return;
        }
    }

    void printStaffMenu(int option) {
```

```
        cursorPositionCenter(20, 42);
        for (int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(21, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(22, 42);
        printf("\xb2              " BOLD GREEN ITALIC "STAFF MENU" RESET "
    \xb2\n");
        cursorPositionCenter(23, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(24, 42);
        for (int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(25, 42);
        printf("\xb2                                        \xb2\n");

        if (option == 1) {
            cursorPositionCenter(26, 42);
            printf("\xb2" BLUE "   -> 1. Add Item               <-   " RESET
    "\xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Add Stock                      \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Delete Stock                   \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Hash-Table View                \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Check Expiration               \xb2\n");
            cursorPositionCenter(31, 42);
            printf("\xb2   6. Update Data                    \xb2\n");
            cursorPositionCenter(32, 42);
            printf("\xb2   7. Logout                         \xb2\n");
        } else if (option == 2) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Add Item                       \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2" BLUE "   -> 2. Add Stock               <-   " RESET
    "\xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Delete Stock                   \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Hash-Table View                \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Check Expiration               \xb2\n");
            cursorPositionCenter(31, 42);
            printf("\xb2   6. Update Data                    \xb2\n");
            cursorPositionCenter(32, 42);
            printf("\xb2   7. Logout                         \xb2\n");
        } else if (option == 3) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Add Item                       \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Add Stock                      \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2" BLUE "   -> 3. Delete Stock            <-   " RESET
    "\xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Hash-Table View                \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Check Expiration               \xb2\n");
            cursorPositionCenter(31, 42);
```

```
            printf("\xb2   6. Update Data                       \xb2\n");
            cursorPositionCenter(32, 42);
            printf("\xb2   7. Logout                            \xb2\n");
        } else if (option == 4) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Add Item                          \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Add Stock                         \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Delete Stock                      \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2" BLUE "   -> 4. Hash-Table View         <-   " RESET
    "\xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Check Expiration                  \xb2\n");
            cursorPositionCenter(31, 42);
            printf("\xb2   6. Update Data                       \xb2\n");
            cursorPositionCenter(32, 42);
            printf("\xb2   7. Logout                            \xb2\n");
        } else if (option == 5) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Add Item                          \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Add Stock                         \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Delete Stock                      \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Hash-Table View                   \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2" BLUE "   -> 5. Check Expiration        <-   " RESET
    "\xb2\n");
            cursorPositionCenter(31, 42);
            printf("\xb2   6. Update Data                       \xb2\n");
            cursorPositionCenter(32, 42);
            printf("\xb2   7. Logout                            \xb2\n");
        } else if (option == 6) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Add Item                          \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Add Stock                         \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Delete Stock                      \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Hash-Table View                   \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Check Expiration                  \xb2\n");
            cursorPositionCenter(31, 42);
            printf("\xb2" BLUE "   -> 6. Update Data             <-   " RESET
    "\xb2\n");
            cursorPositionCenter(32, 42);
            printf("\xb2   7. Logout                            \xb2\n");
        } else if (option == 7) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Add Item                          \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Add Stock                         \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Delete Stock                      \xb2\n");
            cursorPositionCenter(29, 42);
            printf("\xb2   4. Hash-Table View                   \xb2\n");
            cursorPositionCenter(30, 42);
            printf("\xb2   5. Check Expiration                  \xb2\n");
            cursorPositionCenter(31, 42);
```

```
        printf("\xb2   6. Update Data                        \xb2\n");
        cursorPositionCenter(32, 42);
        printf("\xb2" BLUE "   -> 7. Logout                    <-   " RESET
"\xb2\n");
    }

    cursorPositionCenter(33, 42);
    printf("\xb2                                          \xb2\n");
    cursorPositionCenter(34, 42);
    for (int i = 0; i < 40; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(36, 76);
    printf("Use arrow keys (" GREEN "Up" RESET " / " GREEN "Down" RESET ") to
navigate and press " GREEN "Enter" RESET " to select an option.\n");
}

// Menu pada bagian warehouse staff
void staffMenu(int *opt) {
    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);
    int option = 1;
    int product = 1;
    int ch;

    while (1) {
        system("cls");
        today();
        printProductOverview(product);
        printStaffMenu(option);

        ch = _getch();
        if (ch == 0 || ch == 224) {
            ch = _getch();
            switch (ch) {
                case 72: // Up
                    if(option > 1) option--;
                    break;
                case 80: // Down
                    if(option < 7) option++;
                    break;
                case 75: // Left
                    if(product > 1) product--;
                    break;
                case 77: // Right
                    if((float)product < ((float)totalProduct / 10)) product++;
                    break;
                default:
                    break;
            }
        } else if (ch == 13) {
            // Enter
            *opt = option;
            return;
        }
    }
}

void addItem() {
    system("cls");
    today();
```

```
        char dateNow[15];
        todayDate(dateNow);

        cursorPositionCenter(8, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(9, 92);
        printf("\xb2
    \xb2\n");
        cursorPositionCenter(10, 92);
        printf("\xb2                                          " BOLD GREEN ITALIC "ADD
    ITEM" RESET "                                          \xb2\n");
        cursorPositionCenter(11, 92);
        printf("\xb2
    \xb2\n");
        cursorPositionCenter(12, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(13, 92);
        printf("\xb2
    \xb2\n");
        for (int i = 0; i < 12; i++) {
            cursorPositionCenter(14 + i, 92);
            printf("\xb2\n");
        }
        for (int i = 0; i < 12; i++) {
            cursorPositionCenter(14 + i, -86);
            printf("\xb2\n");
        }
        cursorPositionCenter(24, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(25, 92);
        printf("\xb2
    \xb2\n");
        cursorPositionCenter(26, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }

        cursorPositionCenter(14, 88);
        printf("Input new product name (3 - 15 Characters) >> ");
        cursorPositionCenter(16, 88);
        printf("Input starting stocks [1-100] >> ");
        cursorPositionCenter(18, 88);
        printf("Input price (Min. Value: 1000) >> ");
        cursorPositionCenter(20, 88);
        printf("Input expiry date (After %s) [Format: DD/MM/YYYY] >> ", dateNow);
        cursorPositionCenter(22, 88);
        printf("Generated ID:\n");
        cursorPositionCenter(25, 88);
        printf("Status:\n");

        char name[15];
        int stock;
        long int price;
        char expDate[15];

        do {
            cursorPositionCenter(14, 92);
```

```
        printf("\xb2
\xb2\n");
        cursorPositionCenter(14, 88);
        printf(BLUE "Input new product name (3 - 15 Characters) >> " RESET);
        cursorPositionCenter(14, -4);
        scanf(" %[^\n]", name);
    } while(!validateName(name));

    cursorPositionCenter(14, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(14, 88);
    printf("Input new product name (3 - 15 Characters) >> %s\n", name);

    do {
        cursorPositionCenter(16, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(16, 88);
        printf(BLUE "Input starting stocks [1-100] >> " RESET);
        cursorPositionCenter(16, 22);
        scanf(" %d", &stock);
    } while(!validateStock(stock));

    cursorPositionCenter(16, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(16, 88);
    printf("Input starting stocks [1-100] >> %d\n", stock);

    do {
        cursorPositionCenter(18, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(18, 88);
        printf(BLUE "Input price (Min. Value: 1000) >> " RESET);
        scanf(" %ld", &price);
    } while(!validatePrice(price, 25));

    cursorPositionCenter(18, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(18, 88);
    printf("Input price (Min. Value: 1000) >> %d\n", price);

    do {
        cursorPositionCenter(20, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(20, 88);
        printf(BLUE "Input expiry date (After %s) [Format: DD/MM/YYYY] >> "
RESET, dateNow);
        cursorPositionCenter(20, -34);
        scanf(" %[^\n]", expDate);
    } while(!validateExpDate(expDate, 25));

    cursorPositionCenter(20, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(20, 88);
    printf("Input expiry date (After %s) [Format: DD/MM/YYYY] >> ", dateNow);

    char id[5];
```

```
    generateId(name, id);

    insert(id, name, stock, price, expDate);

    saveToFile();

    cursorPositionCenter(22, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(22, 88);
    printf(BLUE "Generated ID: %s" RESET "\n", id);

    cursorPositionCenter(25, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(25, 88);
    printf("Status: " GREEN "Product added to database!" RESET "\n");

    cursorPositionCenter(28, 34);
    system("pause");
}

void addStock() {
    system("cls");
    today();

    cursorPositionCenter(8, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(9, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(10, 92);
    printf("\xb2                                          " BOLD GREEN ITALIC "ADD
STOCK" RESET "                                        \xb2\n");
    cursorPositionCenter(11, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(12, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    for (int i = 0; i < 8; i++) {
        cursorPositionCenter(13 + i, 92);
        printf("\xb2\n");
    }
    for (int i = 0; i < 8; i++) {
        cursorPositionCenter(13 + i, -86);
        printf("\xb2\n");
    }
    cursorPositionCenter(21, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }

    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);
    char id[5];

    cursorPositionCenter(16, 88);
    printf("Status: ");
```

```
    cursorPositionCenter(18, 88);
    printf("Stock Info: ");

    while (1) {
        cursorPositionCenter(14, 88);
        printf("Input item ID [A-Z][A-Z][1-9][1-9] >> ");
        cursorPositionCenter(14, 12);
        scanf(" %[^\n]", id);

        clearInputBuffer();

        if (validateID(id)) {
            break;
        }

        cursorPositionCenter(14, 88);
        printf(CLEAR_LINE);
        cursorPositionCenter(14, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(16, 72);
        printf(RED "Invalid ID format! Please try again." RESET "\n");
    }

    int foundAt = -1;

    for(int i = 0; i < TABLESIZE; i++) {
        struct hashTable *node = table[i];
        while (node) {
            if (strcmp(node->itemID, id) == 0) {
                foundAt = i; // Ditemukan
            }
            node = node->next;
        }
        if (foundAt != -1) break;
    }

    if (foundAt != -1) {
        int newStock;
        cursorPositionCenter(16, 72);
        printf("ID %s found in the database at index %d.\n", id, foundAt);
        struct hashTable *current = table[foundAt];
        while(strcmp(current->itemID, id) != 0) {
            current = current->next;
        }

        while(1) {
            cursorPositionCenter(14, 88);
            printf(CLEAR_LINE);
            cursorPositionCenter(14, 92);
            printf("\xb2
\xb2\n");

            cursorPositionCenter(14, 88);
            printf("Input total stock to add [1-100] >> ");
            scanf(" %d", &newStock);

            clearInputBuffer();

            if(newStock > 0 && newStock < 101) {
                break;
            }
```

```
            cursorPositionCenter(16, 72);
            printf(RED "Invalid stock! Please try again." RESET "\n");
        }

        char date[15];
        todayDate(date);

        current->stock += newStock;

        cursorPositionCenter(18, 64);
        printf("%s stock added successfully on %s!\n", current->itemName, date);
        cursorPositionCenter(19, 64);
        printf("New stock: %d\n", current->stock);


        saveToFile();
    } else {
        cursorPositionCenter(16, 72);
        printf("ID %s not found in the database. Please try again.\n", id);
    }

    cursorPositionCenter(23, 34);
    system("pause");
}

void deleteStock() {
    system("cls");
    today();

    cursorPositionCenter(8, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(9, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(10, 92);
    printf("\xb2                                          " BOLD GREEN ITALIC "DELETE
STOCK" RESET "                                        \xb2\n");
    cursorPositionCenter(11, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(12, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    for (int i = 0; i < 8; i++) {
        cursorPositionCenter(13 + i, 92);
        printf("\xb2\n");
    }
    for (int i = 0; i < 8; i++) {
        cursorPositionCenter(13 + i, -86);
        printf("\xb2\n");
    }
    cursorPositionCenter(21, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }

    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);
```

```
    char id[5];

    cursorPositionCenter(16, 88);
    printf("Status: ");

    cursorPositionCenter(18, 88);
    printf("Stock Info: ");

    while (1) {
        cursorPositionCenter(14, 88);
        printf(CLEAR_LINE);
        cursorPositionCenter(14, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(14, 88);
        printf("Input item ID [A-Z][A-Z][1-9][1-9] >> ");
        cursorPositionCenter(14, 12);
        scanf(" %[^\n]", id);

        clearInputBuffer();

        if (validateID(id)) {
            break;
        }

        cursorPositionCenter(16, 88);
        printf(CLEAR_LINE);
        cursorPositionCenter(16, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(16, 88);
        printf("Status: ");
        cursorPositionCenter(16, 72);
        printf(RED "Invalid ID format! Please try again." RESET "\n");
    }

    int foundAt = -1;

    for(int i = 0; i < TABLESIZE; i++) {
        struct hashTable *node = table[i];
        while (node) {
            if (strcmp(node->itemID, id) == 0) {
                foundAt = i; // Ditemukan
            }
            node = node->next;
        }
        if (foundAt != -1) break;
    }

    if (foundAt != -1) {
        int deleteStock;
        cursorPositionCenter(16, 72);
        printf("ID %s found in the database at index %d.\n", id, foundAt);
        struct hashTable *current = table[foundAt];
        while(strcmp(current->itemID, id) != 0) {
            current = current->next;
        }

        while(1) {
            cursorPositionCenter(14, 88);
            printf(CLEAR_LINE);
            cursorPositionCenter(14, 92);
```

```
            printf("\xb2
\xb2\n");

            cursorPositionCenter(14, 88);
            printf("Input total stock to delete [1-%d] >> ", current->stock);
            scanf(" %d", &deleteStock);

            clearInputBuffer();

            if(deleteStock > 0 && deleteStock < current->stock + 1) {
                break;
            }

            cursorPositionCenter(16, 88);
            printf(CLEAR_LINE);
            cursorPositionCenter(16, 92);
            printf("\xb2
\xb2\n");
            cursorPositionCenter(16, 88);
            printf("Status: ");
            cursorPositionCenter(16, 72);
            printf(RED "Invalid stock! Please try again." RESET "\n");
        }

        char date[15];
        todayDate(date);

        current->stock -= deleteStock;

        if(current->stock <= 0) {

            struct hashTable *node = table[foundAt];
            struct hashTable *prev = NULL;

            while (node != NULL && strcmp(node->itemID, id) != 0) {
                prev = node;
                node = node->next;
            }

            if (node == NULL) {
                return;
            }

            if (prev == NULL) {
                table[foundAt] = node->next;
            } else {
                prev->next = node->next;
            }

            free(node);
            cursorPositionCenter(18, 64);
            printf("%s deleted from database successfully on %s!\n",
current->itemName, date);
        } else {
            cursorPositionCenter(18, 64);
            printf("%s stock deleted successfully on %s!\n", current->itemName,
date);
            cursorPositionCenter(19, 64);
            printf("New stock: %d\n", current->stock);
        }

        saveToFile();
    } else {
```

```
            cursorPositionCenter(16, 72);
            printf("ID %s not found in the database. Please try again.\n", id);
        }

        cursorPositionCenter(23, 34);
        system("pause");
    }

    void printUpdateMenu(int option) {
        cursorPositionCenter(20, 42);
        for(int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(21, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(22, 42);
        printf("\xb2           " BOLD GREEN ITALIC "UPDATE DATA MENU" RESET "
    \xb2\n");
        cursorPositionCenter(23, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(24, 42);
        for(int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(25, 42);
        printf("\xb2                                        \xb2\n");

        if(option == 1) {
            cursorPositionCenter(26, 42);
            printf("\xb2" BLUE "   -> 1. Update Price            <-   " RESET
    "\xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Update Expiry Date              \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Go Back                         \xb2\n");
        } else if(option == 2) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Update Price                    \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2" BLUE "   -> 2. Update Expiry Date      <-   " RESET
    "\xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2   3. Go Back                         \xb2\n");
        } else if(option == 3) {
            cursorPositionCenter(26, 42);
            printf("\xb2   1. Update Price                    \xb2\n");
            cursorPositionCenter(27, 42);
            printf("\xb2   2. Update Expiry Date              \xb2\n");
            cursorPositionCenter(28, 42);
            printf("\xb2" BLUE "   -> 3. Go Back                <-   " RESET
    "\xb2\n");
        }

        cursorPositionCenter(29, 42);
        printf("\xb2                                        \xb2\n");
        cursorPositionCenter(30, 42);
        for (int i = 0; i < 40; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(32, 76);
        printf("Use arrow keys (" GREEN "Up" RESET " / " GREEN "Down" RESET ") to
    navigate and press " GREEN "Enter" RESET " to select an option.\n");
    }
```

```
void updatePrice() {
    system("cls");
    today();

    cursorPositionCenter(8, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    cursorPositionCenter(9, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(10, 92);
    printf("\xb2                                        " BOLD GREEN ITALIC  "UPDATE
DATA PRICE" RESET "                                       \xb2\n");
    cursorPositionCenter(11, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(12, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }
    for (int i = 0; i < 9; i++) {
        cursorPositionCenter(13 + i, 92);
        printf("\xb2\n");
    }
    for (int i = 0; i < 9; i++) {
        cursorPositionCenter(13 + i, -86);
        printf("\xb2\n");
    }
    cursorPositionCenter(22, 92);
    for (int i = 0; i < 90; i++) {
        printf("\xb2");
    }

    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);
    char id[5];

    cursorPositionCenter(16, 88);
    printf("Input new item price (Min. Value: 1000) >> ");

    cursorPositionCenter(18, 88);
    printf("Status: ");

    cursorPositionCenter(20, 88);
    printf("Update Info: ");

    while (1) {
        cursorPositionCenter(14, 88);
        printf(BLUE "Input item ID [A-Z][A-Z][1-9][1-9] >> " RESET);
        cursorPositionCenter(14, 12);
        scanf(" %[^\n]", id);

        clearInputBuffer();

        if (validateID(id)) {
            break;
        }

        cursorPositionCenter(14, 88);
        printf(CLEAR_LINE);
```

```
        cursorPositionCenter(14, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(18, 72);
        printf(RED "Invalid ID format! Please try again." RESET "\n");
    }

    cursorPositionCenter(14, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(14, 88);
    printf("Input item ID [A-Z][A-Z][1-9][1-9] >> %s", id);

    int foundAt = -1;

    for(int i = 0; i < TABLESIZE; i++) {
        struct hashTable *node = table[i];
        while (node) {
            if (strcmp(node->itemID, id) == 0) {
                foundAt = i; // Ditemukan
            }
            node = node->next;
        }
        if (foundAt != -1) break;
    }

    if (foundAt != -1) {
        long int newPrice;
        cursorPositionCenter(18, 72);
        printf(GREEN "ID %s found in the database at index %d." RESET "\n", id,
foundAt);
        struct hashTable *current = table[foundAt];
        while(strcmp(current->itemID, id) != 0) {
            current = current->next;
        }

        cursorPositionCenter(20, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(20, 88);
        printf("Update Info: %s current price is Rp %ld\n", current->itemName,
current->price);

        do {
            cursorPositionCenter(16, 92);
            printf("\xb2
\xb2\n");
            cursorPositionCenter(16, 88);
            printf(BLUE "Input new %s price (Min. Value: 1000) >> " RESET,
current->itemName);
            scanf(" %ld", &newPrice);
        } while(!validatePrice(newPrice, 18));

        current->price = newPrice;

        saveToFile();

        cursorPositionCenter(16, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(16, 88);
        printf("Input new %s price (Min. Value: 1000) >> %ld\n",
current->itemName, newPrice);
```

```
            cursorPositionCenter(20, 92);
            printf("\xb2
\xb2\n");
            cursorPositionCenter(20, 88);
            printf("Update Info: %s new price is Rp %ld\n", current->itemName,
newPrice);
            cursorPositionCenter(18, 92);
            printf("\xb2
\xb2\n");
            cursorPositionCenter(18, 88);
            printf("Status: " GREEN "%s price updated in the database!" RESET "\n",
current->itemName);
        } else {
            cursorPositionCenter(18, 72);
            printf(RED "ID %s not found in the database. Please try again." RESET
"\n", id);
        }

        cursorPositionCenter(25, 34);
        system("pause");
    }

    void updateExpiryDate() {
        system("cls");
        today();

        cursorPositionCenter(8, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }
        cursorPositionCenter(9, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(10, 92);
        printf("\xb2                                        " BOLD GREEN ITALIC  "UPDATE
DATA EXPIRY DATE" RESET "                                       \xb2\n");
        cursorPositionCenter(11, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(12, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }
        for (int i = 0; i < 9; i++) {
            cursorPositionCenter(13 + i, 92);
            printf("\xb2\n");
        }
        for (int i = 0; i < 9; i++) {
            cursorPositionCenter(13 + i, -86);
            printf("\xb2\n");
        }
        cursorPositionCenter(22, 92);
        for (int i = 0; i < 90; i++) {
            printf("\xb2");
        }

        int totalProduct = 0;
        clearHash();
        extractFile(&totalProduct);
        char id[5];

        char dateNow[15];
        todayDate(dateNow);
```

```
    cursorPositionCenter(16, 88);
    printf("Input new item expiry date (After %s) [DD/MM/YYYY] >> ", dateNow);

    cursorPositionCenter(18, 88);
    printf("Status: ");

    cursorPositionCenter(20, 88);
    printf("Update Info: ");

    while (1) {
        cursorPositionCenter(14, 88);
        printf(BLUE "Input item ID [A-Z][A-Z][1-9][1-9] >> " RESET);
        cursorPositionCenter(14, 12);
        scanf(" %[^\n]", id);

        clearInputBuffer();

        if (validateID(id)) {
            break;
        }

        cursorPositionCenter(14, 88);
        printf(CLEAR_LINE);
        cursorPositionCenter(14, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(18, 72);
        printf(RED "Invalid ID format! Please try again." RESET "\n");
    }

    cursorPositionCenter(14, 92);
    printf("\xb2
\xb2\n");
    cursorPositionCenter(14, 88);
    printf("Input item ID [A-Z][A-Z][1-9][1-9] >> %s", id);

    int foundAt = -1;

    for(int i = 0; i < TABLESIZE; i++) {
        struct hashTable *node = table[i];
        while (node) {
            if (strcmp(node->itemID, id) == 0) {
                foundAt = i; // Ditemukan
            }
            node = node->next;
        }
        if (foundAt != -1) break;
    }

    if (foundAt != -1) {
        char newDate[15];
        cursorPositionCenter(18, 72);
        printf(GREEN "ID %s found in the database at index %d." RESET "\n", id,
foundAt);
        struct hashTable *current = table[foundAt];
        while(strcmp(current->itemID, id) != 0) {
            current = current->next;
        }

        cursorPositionCenter(20, 92);
        printf("\xb2
\xb2\n");
```

```
        cursorPositionCenter(20, 88);
        printf("Update Info: %s current expiry date is %s\n", current->itemName,
current->expiryDate);

        do {
            cursorPositionCenter(16, 92);
            printf("\xb2
\xb2\n");
            cursorPositionCenter(16, 88);
            printf(BLUE "Input new %s expiry date (After %s) [DD/MM/YYYY] >> "
RESET, current->itemName, dateNow);
            scanf(" %[^\n]", newDate);
        } while(!validateExpDate(newDate, 18));

        strcpy(current->expiryDate, newDate);

        saveToFile();

        cursorPositionCenter(16, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(16, 88);
        printf("Input new %s expiry date (After %s) [DD/MM/YYYY] >> %s",
current->itemName, dateNow, newDate);
        cursorPositionCenter(20, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(20, 88);
        printf("Update Info: %s new expiry date is %s\n", current->itemName,
newDate);
        cursorPositionCenter(18, 92);
        printf("\xb2
\xb2\n");
        cursorPositionCenter(18, 88);
        printf("Status: " GREEN "%s expiry date updated in the database!" RESET
"\n", current->itemName);
    } else {
        cursorPositionCenter(18, 72);
        printf(RED "ID %s not found in the database. Please try again." RESET
"\n", id);
    }

    cursorPositionCenter(25, 34);
    system("pause");
}

void updateData() {
    int totalProduct = 0;
    clearHash();
    extractFile(&totalProduct);
    int option = 1;
    int product = 1;
    int ch;

    while (1) {
        system("cls");
        today();
        printProductOverview(product);
        printUpdateMenu(option);

        ch = _getch();
        if (ch == 0 || ch == 224) {
            ch = _getch();
```

```
            switch (ch) {
                case 72: // Up
                    if(option > 1) option--;
                    break;
                case 80: // Down
                    if(option < 3) option++;
                    break;
                case 75: // Left
                    if(product > 1) product--;
                    break;
                case 77: // Right
                    if((float)product < ((float)totalProduct / 10)) product++;
                    break;
                default:
                    break;
            }
        } else if (ch == 13) {
            if(option == 1) {
                updatePrice();
                break;
            } else if(option == 2) {
                updateExpiryDate();
                break;
            } else if(option == 3) {
                break;
            }
        }
    }
}

// Fungsi utama untuk warehouse staff
void staff() {
    system("cls");

    if(login(3) == 1) {
        int opt;
        while(1) {
            staffMenu(&opt);

            if(opt == 1) {
                addItem();
            } else if(opt == 2) {
                addStock();
            } else if(opt == 3) {
                deleteStock();
            } else if(opt == 4) {
                hashTableControl();
            } else if(opt == 5) {
                checkExpiration();
            } else if(opt == 6) {
                updateData();
            } else if(opt == 7) {
                return;
            }
        }
    } else {
        return;
    }
}

void printExitChoice(int option) {
    if(option == 1) {
        cursorPositionCenter(23, 46);
```

```
            printf(BOLD BLUE "[ Yes ]" RESET "\n");
            cursorPositionCenter(24, 46);
            printf("[ No ]\n");
        } else if(option == 2) {
            cursorPositionCenter(23, 46);
            printf("[ Yes ]\n");
            cursorPositionCenter(24, 46);
            printf(BOLD BLUE "[ No ]" RESET "\n");
        }
    }

    // Opsi untuk menutup aplikasi
    int exitProgram() {
        int option = 1;
        int ch;

        cursorPositionCenter(22, 46);
        printf("Are you sure you want to quit the application?\n");
        while (1) {
            printExitChoice(option);

            ch = _getch();
            if (ch == 0 || ch == 224) {
                ch = _getch();
                switch (ch) {
                    case 72: // Up
                        if (option > 1) option--;
                        printf(CLEAR_LINE CLEAR_LINE);
                        break;
                    case 80: // Down
                        if (option < 2) option++;
                        printf(CLEAR_LINE CLEAR_LINE);
                        break;
                    default:
                        break;
                }
            } else if (ch == 13) {
                // Enter
                if(option == 1) {
                    system("cls");

                    char exit1[16] = "Exiting the app";
                    char exit2[31] = "Thankyou for using this app :)";
                    cursorPositionCenter(14, 14);
                    printf(BOLD);
                    for(int i = 0; i < strlen(exit1); i++) {
                        Sleep(10);
                        printf("%c", exit1[i]);
                    }
                    Sleep(200);
                    cursorPositionCenter(15, 30);
                    for(int i = 0; i < strlen(exit2); i++) {
                        Sleep(10);
                        printf("%c", exit2[i]);
                    }
                    printf(RESET);
                    Sleep(2500);
                    return 0;
                } else return 1;
            }
        }
    }
```

```
int main() {
    clearHash();
    // intro();

    do {
        int opt = loginPage();

        if(opt == 1) {
            cashier();
        } else if(opt == 2) {
            staff();
        } else if(opt == 3) {
            if (!exitProgram()) return 0;
        }
    } while (1);

    return 0;
}
```

BAB V

5. Reference

- tutorialpoints.com, “Hash Table Program in C – Tutorialspoint” < https://www.tutorialspoint.com/data_structures_algorithms/hash_table_program_in_c.htm#:~:text=Hash%20Table%20is%20a%20data,index%20of%20the%20desired%20data
- geekforgeeks.org, “Hashing Data Structure“ <https://www.geeksforgeeks.org/hashing-data-structure/
- geekforgeeks.org, “Hashing | Set 1 (Introduction)“ < https://www.geeksforgeeks.org/hashing-set-1-introduction/
- educative.io, “What is chaining in hash tables ?” https://www.educative.io/edpresso/what-is-chaining-in-hash-tables)
- tutorialpoints.com, “Hash Function and Hash Tables” <https://www.tutorialspoint.com/Hash-Functions-and-Hash-Tables
- educative.io , “What is hashing” < https://www.educative.io/edpresso/what-is-hashing
  techiedelight.com, “Print Current date and time in C” https://www.techiedelight.com/print-current-date-and-time-in-c/
- geekforgeeks.org, “Linked List Data Structure” https://www.geeksforgeeks.org/linked-list-data-structure/

6. **Assesment**

LEMBAR PENILAIAN

PEMBUATAN MENU KASIR DENGAN METODE HASHING

MATA KULIAH COMP6362004 – DATA STRUCTURES

KELAS LB-20

Semester Genap 2023/2024

DAFTAR MAHASISWA	NILAI				BOBOT				KREDIT				TOTAL KREDIT
	1	2	3	4	1	2	3	4	1	2	3	4	
2702355302 - Kevin Joseph Handoyo													
2702350106 – Muhammad Ibraahiim Putra Wahana													
TOTAL													

KETERANGAN :

- **Skala Penilaian : 0 sd 100**
- **Komponen**
    1. **: Laporan**
    2. **: Produk**
    3. **: Pengetahuan**
    4. **: Solusi**

Malang, …… - ……. ……..

<ttd>

Chasandra Puspitasari

G. Other

1. **Jobdesc Anggota**

   2702355302 - Kevin Joseph Handoyo :
   - Purchase Product
   - Cashflow
   - Check Expiration
   - Add Item
   - Hashing Logic
   - Hash-Table View
   - Validasi Input
   - Update Price
   - Update Expiry Date

   2702350106 – Muhammad Ibraahiim Putra Wahana :
   - Laporan
   - User Interface
   - Add Stock
   - Delete Stock
   - Database
   - Design Menu
   - File Manipulation
   - Login Logic

2. **Video Presentasi**

   https://binusianorg-my.sharepoint.com/personal/kevin_handoyo001_binus_ac_id/_layouts/15/guestaccess.aspx?share=EVKPFDsShsFKkD5t1qKKpEYBhtOTbZGL8B0Dehv8MFEgTg&nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=CpBR6j

3. **PPT Presentasi**

   https://www.canva.com/design/DAGIQP-Bej0/HyL50RLiOL3SKj5q7GWIJQ/edit?utm_content=DAGIQP-Bej0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton