



Laporan Project Team POINT OF SALES

Praktikum Programa Komputer
2023

Presented By

FAHRUL KHARIS AFFANDI (I0322041)

FATIHAAH ASY SYIFA (I0322043)

GEA NADA ADERANGGA (I0322049)

LUQMAN AL HAKIM (I0322069)

BAB I

DESKRIPSI MASALAH

Bab ini menjelaskan mengenai deskripsi masalah-masalah yang diangkat sebagai hal yang melandasi pembuatan program Point of Sales.

1.1 Pendahuluan

Subbab ini menjelaskan tentang latar belakang dari program Point of Sales.

Point of Sales (POS) merupakan sebuah program yang dirancang untuk mengatasi berbagai masalah yang terkait dengan pencatatan transaksi antara penjual dan pembeli. Masalah-masalah yang sering dihadapi dengan sistem pencatatan manual adalah data yang rentan hilang atau rusak, kesalahan perhitungan manusia yang menghasilkan data yang tidak akurat, ketidakefisienan dalam proses pencatatan manual, dan kurangnya pengawasan terhadap pertumbuhan profit dalam rentang waktu tertentu. Selain itu, pencatatan manual mengenai penambahan modal dan penjualan produk juga tidak terintegrasi dengan keuangan toko secara keseluruhan.

Demi mengatasi masalah-masalah tersebut, diperlukan solusi yang efektif. Pertama, dibutuhkan sistem *login* yang dapat membatasi akses masuk ke program. Fitur keamanan ini berguna untuk mencegah orang yang tidak berwenang membuka dan memodifikasi data yang ada di program untuk tujuan tertentu yang merugikan pemilik usaha. Dalam sistem *login*, pengguna perlu memasukkan kredensial seperti *username* dan *password* untuk dapat mengakses program.

Selanjutnya, perlu diciptakan program kasir yang dapat memfasilitasi proses pencatatan otomatis transaksi penjualan produk di suatu toko atau bisnis. Program kasir ini akan dilengkapi dengan fitur-fitur seperti menambah item ke keranjang jika stok masih tersedia, menghitung harga total, menghitung uang sisa belanja, menghitung diskon atau potongan harga, serta membuat *invoice* atau nota sebagai bukti transaksi. Dengan adanya program kasir ini, proses pencatatan transaksi dapat dilakukan secara otomatis dan akurat, mengurangi risiko *human error* dalam perhitungan.

Selain itu, perlu juga dibuat program yang berbasis pada pengelolaan ketersediaan produk dalam gudang. Program ini akan memungkinkan pemilik toko

untuk menambah, mengubah, dan menghapus data produk dengan mudah. Fokus kami dalam rancangan ini adalah toko sepatu. Dengan adanya program pengelolaan stok produk ini, pemilik toko dapat memantau dan mengatur ketersediaan produk dalam hal ini adalah sepatu dengan lebih efisien, sehingga dapat menghindari kekurangan atau kelebihan stok yang tidak diinginkan.

Selanjutnya, program penjualan yang memungkinkan pemilik toko untuk melakukan pencatatan dan manajemen penjualan sepatu atau layanan juga sangat diperlukan. Program ini akan mencatat profit penjualan per bulan serta jumlah sepatu yang terjual dalam rentang waktu tertentu. Dengan adanya program penjualan ini, pemilik toko dapat memantau dan menganalisis performa penjualan sepatu, serta melakukan strategi bisnis yang lebih efektif berdasarkan data yang akurat.

Selain program penjualan, program *finance* juga sangat penting dalam mengelola keuangan bisnis. Program *finance* ini memungkinkan pemilik toko untuk mencatat dan mengelola keuangan bisnis secara menyeluruh. Dalam program ini, pengguna dapat mencatat penambahan modal ke program yang nantinya akan dijumlahkan ke saldo. Selain itu, saldo akan bertambah jika terjadi penjualan sepatu dan berkurang jika terjadi pembelian sepatu. Program *finance* juga akan mencatat otomatis hal-hal yang berkaitan dengan saldo, seperti pengeluaran, pemasukan, dan pembayaran hutang. Dengan adanya program *finance*, pemilik toko dapat memiliki gambaran yang jelas tentang kondisi keuangan bisnisnya, melakukan analisis keuangan yang lebih baik, serta membuat keputusan yang tepat dalam mengelola dan mengoptimalkan keuangan toko.

Secara keseluruhan, solusi-solusi yang telah disebutkan di atas dapat membantu mengatasi berbagai masalah yang sering timbul dalam pencatatan transaksi manual. Dengan menggunakan program POS yang memiliki fitur-fitur seperti sistem login, program kasir, pengelolaan stok sepatu, program penjualan, dan program *finance*, pemilik toko dapat meningkatkan efisiensi, akurasi, dan keamanan dalam melakukan pencatatan dan manajemen bisnis mereka. Dalam bentuk GUI yang *user-friendly*, program ini akan memberikan kemudahan penggunaan bagi operator kasir dan pemilik usaha, sehingga mereka dapat fokus pada operasional bisnis dan mengambil keputusan yang lebih baik berdasarkan data

yang akurat dan terpercaya. Meskipun belum terintegrasi dengan dompet digital, solusi-solusi ini sudah memberikan langkah awal yang signifikan dalam meningkatkan pengelolaan toko dan keuangan secara keseluruhan.

1.2 Rumusan Masalah

Subbab ini menjelaskan tentang rumusan masalah dari program Point of Sales. Rumusan masalah dalam laporan ini dapat disampaikan sebagai berikut:

1. Bagaimana solusi untuk menyelesaikan masalah ketidakefektifan pencatatan transaksi dan stok secara manual?
2. Bagaimana diagram alir yang digunakan dalam penyelesaian masalah ketidakefektifan pencatatan transaksi dan stok secara manual?
3. Bagaimana kode program yang digunakan dalam penyelesaian masalah ketidakefektifan pencatatan transaksi dan stok secara manual?

1.3 Tujuan

Subbab ini menjelaskan tentang tujuan dari program Point of Sales. Tujuan dalam laporan ini dapat disampaikan sebagai berikut:

1. Bagaimana solusi untuk menyelesaikan masalah ketidakefektifan pencatatan transaksi dan stok secara manual?
2. Bagaimana diagram alir yang digunakan dalam penyelesaian masalah ketidakefektifan pencatatan transaksi dan stok secara manual?
3. Bagaimana kode program yang digunakan dalam penyelesaian masalah ketidakefektifan pencatatan transaksi dan stok secara manual?

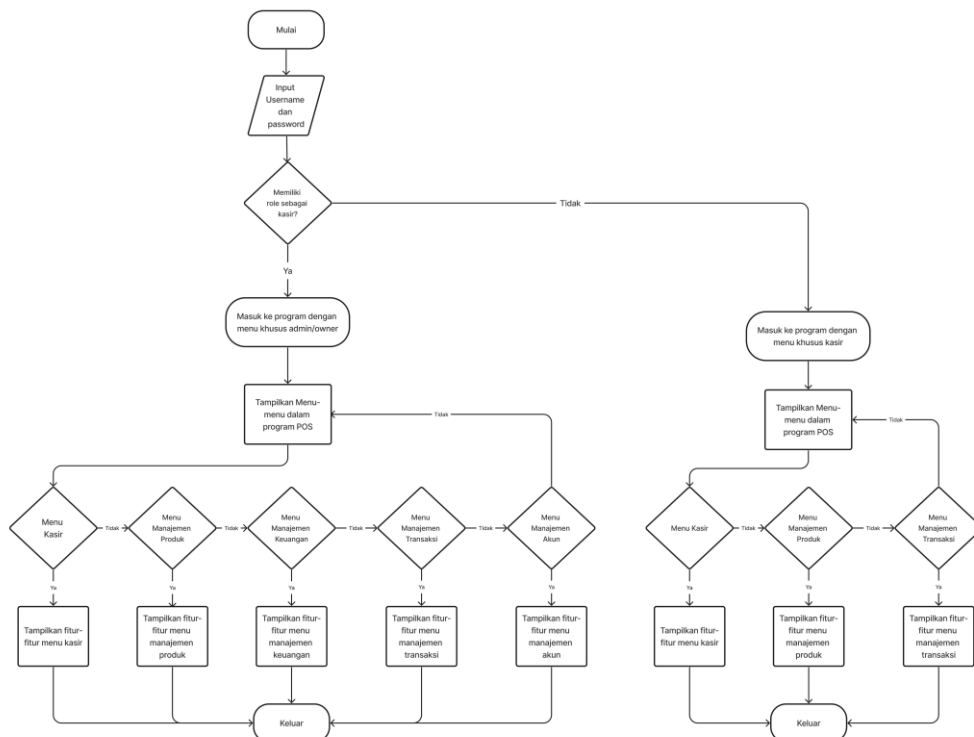
BAB II FLOWCHART

Bab ini menjelaskan mengenai *flowchart* pada program Point of Sales.

2.1 *Flowchart* Umum

Subbab ini menjelaskan mengenai keseluruhan program Point of Sales yang digambarkan ke dalam suatu *flowchart* umum. Melalui *flowchart* ini, dapat diketahui tentang alur berjalannya seluruh program Point of Sales, yang di dalamnya terdapat berbagai menu dan fitur seperti autentikasi *login*, kasir, manajemen produk, manajemen keuangan, manajemen transaksi, dan manajemen akun pegawai.

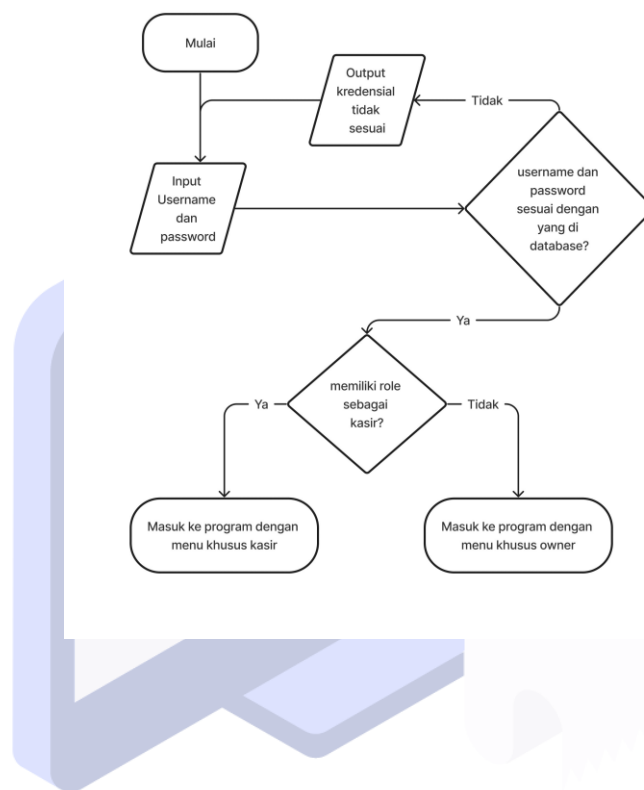
Flowchart Umum



2.2 Flowchart Autentikasi Login

Subbab ini menjelaskan mengenai halaman *login*. *Login* adalah proses autentikasi yang terdapat pada program yang berguna untuk membatasi akses masuk ke program, fitur keamanan ini untuk mencegah orang yang tidak berkenan membuka dan memodifikasi data yang ada di program untuk tujuan tertentu yang merugikan pemilik usaha, untuk masuk ke program diperlukan memasukkan kredensial pengguna, seperti *username* dan *password*.

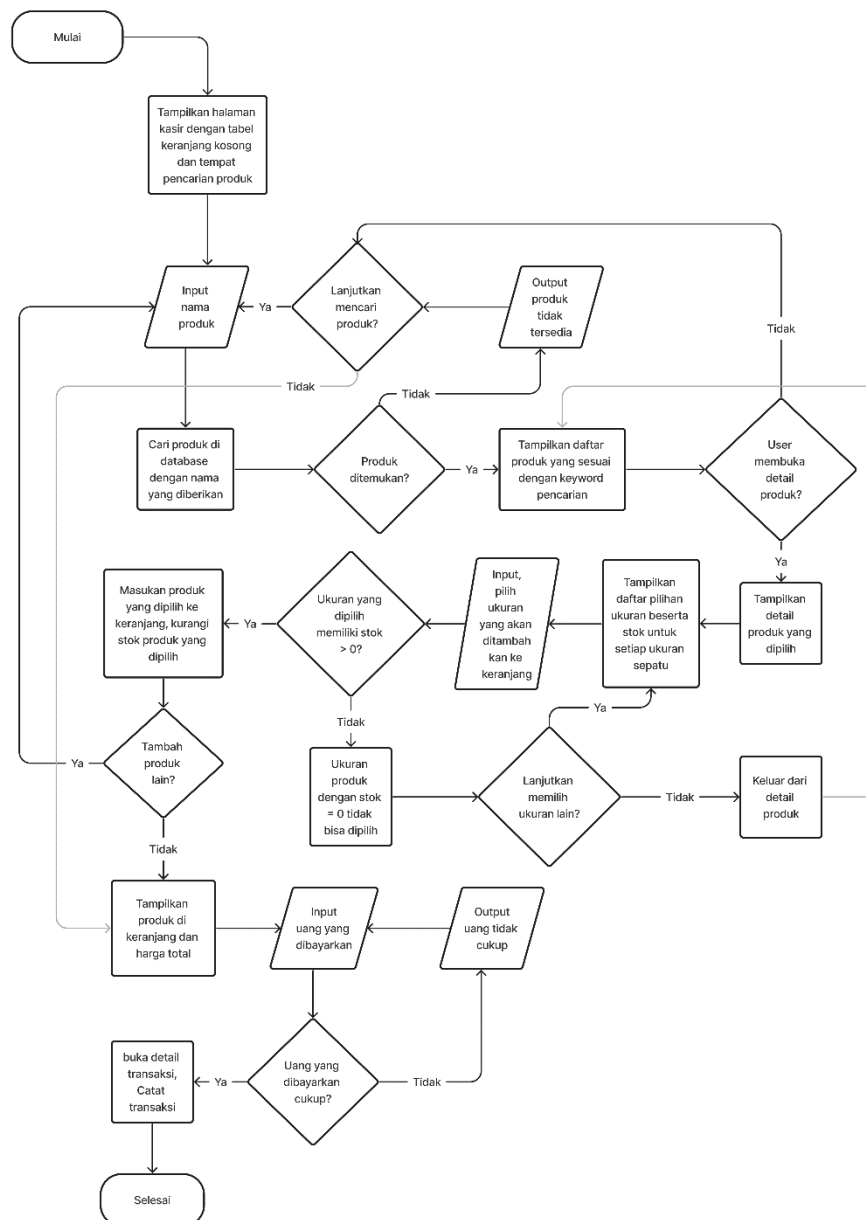
Flowchart Autentikasi Login



2.3 Flowchart Menu Kasir

Subbab ini menjelaskan mengenai alur fitur halaman kasir pada program Point of Sales.

Flowchart Kasir

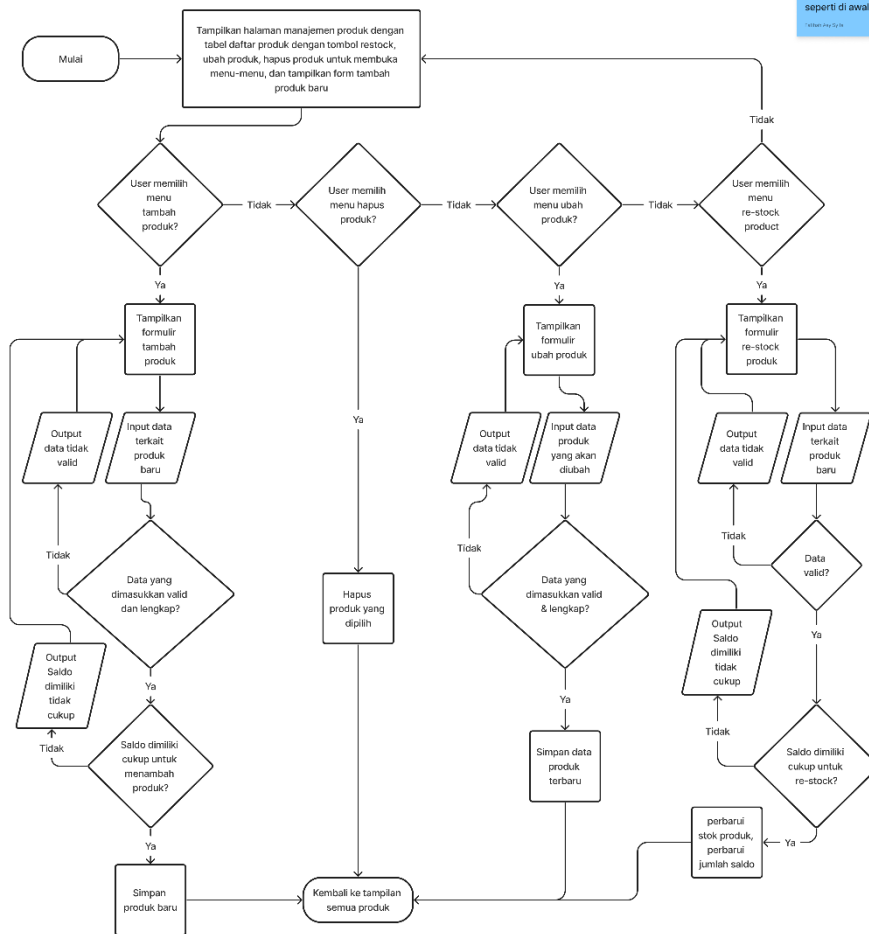


Pada menu kasir terdapat fitur cari produk, tambah *item* ke keranjang, daftar *item* yang dibeli, hapus *item*, detail transaksi, dan simpan transaksi.

2.4 Flowchart Menu Manajemen Produk

Subbab ini menjelaskan mengenai alur fitur menu manajemen produk pada program Point of Sales.

Flowchart Menu Manajemen Produk



Karena program berbentuk GUI, ketika user sudah masuk halaman manajemen produk, akan terdapat tampilan berupa tabel daftar produk beserta tombol untuk mengakses menu-menu, jika user tidak memilih menu-menu tersebut, berarti tidak terjadi apa-apa, tetap menampilkan tabel daftar produk seperti di awal

Program yang berbasis pada pengelolaan ketersediaan produk dalam gudang. Pengelolaan stok produk seperti:

a. Menambah produk baru

Fungsi yang menerima input berupa informasi terkait produk, seperti nama produk, harga beli dan harga jual produk, dan data-data lainnya. Fungsi akan memeriksa apakah produk sudah ada dalam daftar produk. Jika iya, program akan memberikan pesan bahwa produk sudah ada. Jika tidak, produk

baru akan ditambahkan ke dalam daftar dengan menggunakan nama produk sebagai kunci. Dengan adanya fitur ini, pengguna dapat menambahkan produk baru ke dalam daftar produk dengan memasukkan informasi terkait data-data yang dibutuhkan.

b. *Restock* dan *resize*

Fitur ini berguna untuk mengubah jumlah stok dan ukuran sepatu yang masih ada dalam gudang. Dengan adanya fitur ini, stok yang dijual akan selalu *ter-update*.

c. Mengubah data produk

Fungsi yang digunakan untuk mengubah informasi terkait produk yang telah dibuat. Program akan menampilkan *form* yang sudah terisi dengan data lama, pengguna dapat mengganti data di *form* yang telah disediakan, lalu melakukan simpan produk, maka otomatis data produk akan tersimpan dengan data baru yang telah dimasukkan.

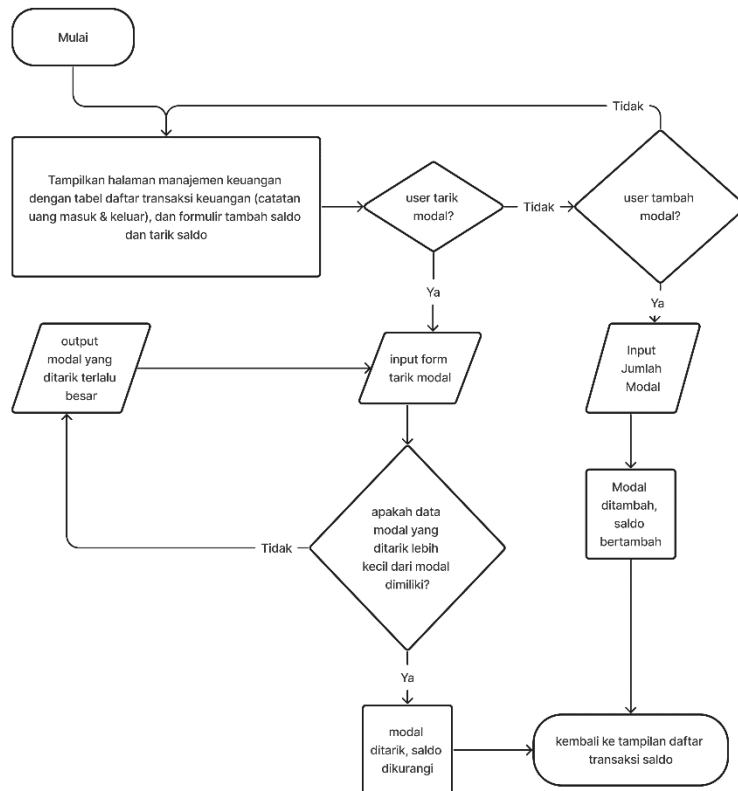
d. Menghapus produk

Fungsi yang digunakan untuk menghapus produk yang telah dibuat, produk akan dihapus secara tampilan atau dengan kata lain dilakukan *soft-delete*. Produk yang dihapus akan ditandai dengan tanda khusus sehingga tidak ditampilkan di daftar produk, akan tetapi jika dilihat di *file csv*, produk tersebut masih ada.

2.5 Flowchart Menu Manajemen Keuangan

Subbab ini menjelaskan mengenai alur fitur menu manajemen keuangan pada program Point of Sales.

Flowchart Menu Manajemen Keuangan

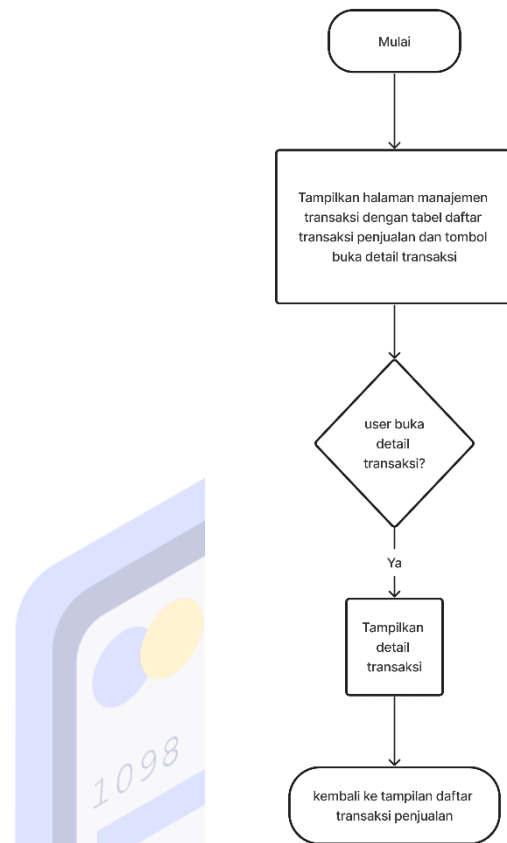


Program yang memungkinkan pemilik toko untuk melakukan pencatatan dan manajemen keuangan bisnis yang meliputi:

- Pengguna dapat mencatatkan penambahan modal ke program yang nantinya akan dijumlahkan ke saldo.
- Saldo akan bertambah jika terjadi penjualan produk dan saldo akan berkurang jika terjadi pembelian produk (kulakan).
- Akan ada pencatatan otomatis tentang hal-hal yang berkaitan dengan saldo.

2.6 Flowchart Menu Manajemen Transaksi

Subbab ini menjelaskan mengenai alur menu manajemen transaksi pada program Point of Sales.

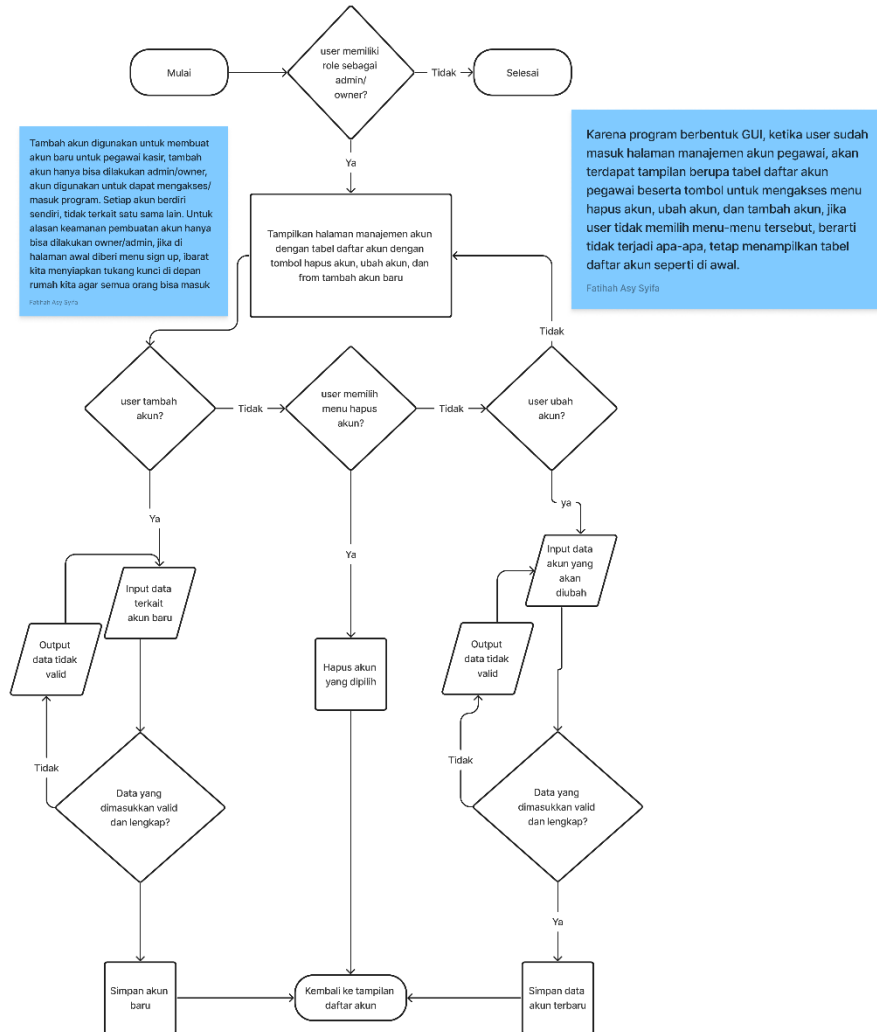


Menu manajemen transaksi dapat diibaratkan seperti menu cetak struk transaksi. Menu ini berguna untuk merekap data hasil transaksi lalu mencetaknya.

2.7 Flowchart Menu Manajemen Akun Pegawai

Subbab ini menjelaskan mengenai alur fitur menu manajemen akun pegawai pada program Point of Sales.

Flowchart Menu Manajemen Akun Pegawai



Tambah akun digunakan untuk membuat akun baru untuk pegawai kasir, tambah akun hanya bisa dilakukan *admin* atau *owner*, akun digunakan untuk dapat mengakses program. Setiap akun berdiri sendiri, tidak terkait satu sama lain. Untuk alasan keamanan pembuatan akun hanya bisa dilakukan *owner* atau *admin*, jika di halaman awal diberi menu *sign up*, ibarat kita menyiapkan tukang kunci di depan rumah kita agar semua orang bisa masuk.

BAB III

KODE PROGRAM

Bab ini menjelaskan mengenai kode-kode program yang digunakan dalam program Point of Sales.

3.1 Kode Program File Pages

Subbab ini menjelaskan mengenai *file* pages yang digunakan untuk menjalankan keseluruhan program Point of Sales.

```
import pandas as pd
from PyQt6 import uic
from PyQt6.QtWidgets import QApplication, QMainWindow

from kasir import Kasir
from produk import Produk
from keuangan import Keuangan
from transaksi import Transaksi
from akun import Akun
from warning import Warning

class LoginWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi('login.ui', self)
        self.df_temp_user = pd.read_csv('data/temp_user.csv')
        self.button_login.clicked.connect(self.authenticate)

    def showWarning(self, message):
        self.warning = Warning(message)
        self.warning.show()

    def authenticate(self):
        username = self.input_username.text()
        password = self.input_password.text()

        df = pd.read_csv('data/user.csv')
        users = df[df["username"] == username]

        if not users.empty:
            user = users.iloc[0]
            if user.username == username and user.password == password:
                new_row = [user.id, user.role, user.name,
                user.username]
                self.df_temp_user.drop(self.df_temp_user.index,
                inplace=True)
                self.df_temp_user.loc[0] = new_row
                self.df_temp_user.to_csv('data/temp_user.csv',
                index=False)
```

```

        self.mainApp = MainWindow()
        self.mainApp.show()
        self.close()
    else:
        self.showWarning("Username / Password salah")
else:
    self.showWarning("Username / Password salah")

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi('pages.ui', self)

        self.kasir = Kasir()
        self.stackedWidget.addWidget(self.kasir)
        self.button_kasir.clicked.connect(lambda: self.move_to(0))

        self.produk = Produk()
        self.stackedWidget.addWidget(self.produk)
        self.button_produk.clicked.connect(lambda:
self.move_to(1))

        self.keuangan = Keuangan()
        self.stackedWidget.addWidget(self.keuangan)
        self.button_keuangan.clicked.connect(lambda:
self.move_to(2))

        self.transaksi = Transaksi()
        self.stackedWidget.addWidget(self.transaksi)
        self.button_transaksi.clicked.connect(lambda:
self.move_to(3))

        self.akun = Akun()
        self.stackedWidget.addWidget(self.akun)
        self.button_akun.clicked.connect(lambda: self.move_to(4))

        self.showMenu()
        self.button_keluar.clicked.connect(self.close)

self.kasir.createdNewTransaction.connect(self.transaksi.initTable)
self.kasir.createdNewTransaction.connect(self.keuangan.initTable)

self.produk.makeTransactions.connect(self.keuangan.initTable)

    def move_to(self, index):
        self.stackedWidget.setCurrentIndex(index)

    def showMenu(self):
        self.df_temp_user = pd.read_csv('data/temp_user.csv')
        if self.df_temp_user.loc[0]["role"] == "cashier":
            self.button_keuangan.setVisible(False)
            self.button_akun.setVisible(False)

if __name__ == '__main__':

```

```
app = QApplication([])
window = LoginWindow()
window.show()
app.exec()
```

pages.py

Kode program tersebut berfungsi untuk menjalankan keseluruhan program Point of Sales. Di dalamnya terdapat *syntax* yang digunakan untuk memanggil seluruh *library* dan modul yang digunakan pada *file* pages. File ini digunakan untuk mengakses halaman *login* yang membutuhkan data kredensial dan akan diautentikasi sesuai data yang ada pada *file csv* pendukung, sekaligus mengakses *main window* ketika proses autentikasi telah berhasil. Pada *main window* inilah pengguna dapat mengakses keseluruhan menu yang ada pada program Point of Sales.

Pada program ini, terdapat dua *role* yang dapat menjalankan program Point of Sales, yaitu admin dan kasir. Perbedaannya terletak pada aksesibilitas yang dimiliki oleh kedua *role* tersebut. Apabila dari data *login* diketahui bahwa pengguna memiliki *role* sebagai admin, maka pengguna bisa mengakses semua menu yang ada pada program ini. Sementara, apabila diketahui bahwa *role*-nya adalah sebagai kasir, maka pengguna tidak bisa mengakses menu keuangan dan akun.

3.2 Kode Program Autentikasi Login

Subbab ini menjelaskan mengenai kode program yang digunakan dalam menjalankan autentikasi *login* pada program Point of Sales.

```
class LoginWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi('login.ui', self)
        self.df_temp_user = pd.read_csv('data/temp_user.csv')
        self.button_login.clicked.connect(self.authenticate)

    def showWarning(self, message):
        self.warning = Warning(message)
        self.warning.show()

    def authenticate(self):
        username = self.input_username.text()
        password = self.input_password.text()

        df = pd.read_csv('data/user.csv')
        users = df[df["username"] == username]
```

```

        if not users.empty:
            user = users.iloc[0]
            if user.username == username and user.password ==
password:
                new_row = [user.id, user.role, user.name,
user.username]
                self.df_temp_user.drop(self.df_temp_user.index,
inplace=True)
                self.df_temp_user.loc[0] = new_row
                self.df_temp_user.to_csv('data/temp_user.csv',
index=False)

                self.mainApp = MainWindow()
                self.mainApp.show()
                self.close()
            else:
                self.showWarning("Username / Password salah")
        else:
            self.showWarning("Username / Password salah")

```

pages.py

Kode program autentikasi login terdapat pada file Pages. Melalui kode ini, data kredensial pengguna akan diautentikasi dengan mencocokkan data yang telah ada pada *file csv* pendukung, yaitu *file* 'user.csv'. Kemudian, data pengguna yang sedang melakukan *login* akan disimpan di dalam *file* 'temp_user.csv'. Berikut ini tampilan dari isi kedua *file csv* tersebut.

```

id,role,name,username,password
1,admin,admin,admin,admin123
2,cashier,cashier 1,cashier1,cashier123
3,cashier,cashier 2,cashier2,cashier123
4,cashier,cashier 3,cashier3,cashier123

```

user.csv

```

id,role,name,username
1,admin,0,admin

```

temp_user.csv

3.3 Kode Program Menu Kasir

Subbab ini menjelaskan mengenai kode program yang digunakan dalam menjalankan menu kasir pada program Point of Sales.

```

import pandas as pd
from PyQt6 import QtCore as Qtc
from PyQt6 import uic
from PyQt6.QtCore import Qt, QAbstractTableModel, QDateTime
from PyQt6.QtGui import QIntValidator

```



```

from PyQt6.QtWidgets import QWidget

from kasir_select_item import SelectItem
from transaksi_detail import DetailTransaksi

class SearchModel(QAbstractTableModel):
    def __init__(self, data):
        super(SearchModel, self).__init__()
        self._data = data

    def rowCount(self, index):
        return self._data.shape[0]

    def columnCount(self, parnet=None):
        return self._data.shape[1]

    def data(self, index, role=Qt.ItemDataRole.DisplayRole):
        if index.isValid():
            if role == Qt.ItemDataRole.DisplayRole or role == Qt.ItemDataRole.EditRole:
                value = self._data.iloc[index.row(), index.column()]
                return str(value)

    def headerData(self, col, orientation, role):
        if orientation == Qt.Orientation.Horizontal and role == Qt.ItemDataRole.DisplayRole:
            return self._data.columns[col]

class ItemModel(QAbstractTableModel):
    def __init__(self, data):
        super(ItemModel, self).__init__()
        self._data = data

    def rowCount(self, index):
        return self._data.shape[0]

    def columnCount(self, parnet=None):
        return self._data.shape[1] - 2

    def data(self, index, role=Qt.ItemDataRole.DisplayRole):
        if index.isValid():
            if role == Qt.ItemDataRole.DisplayRole or role == Qt.ItemDataRole.EditRole:
                value = self._data.iloc[index.row(), index.column()]
                return str(value)

    def headerData(self, col, orientation, role):
        if orientation == Qt.Orientation.Horizontal and role == Qt.ItemDataRole.DisplayRole:
            return self._data.columns[col]

class Kasir(QWidget):
    createdNewTransaction = QtC.pyqtSignal(str)
    def __init__(self, *args, **kwargs):

```

```

super().__init__(*args, **kwargs)
uic.loadUi('kasir.ui', self)
self.df_temp_user = pd.read_csv('data/temp_user.csv')

self.button_search.clicked.connect(self.search)
self.button_select.clicked.connect(self.select)
self.button_hapus.clicked.connect(self.hapusItem)

self.button_simpan_transaksi.clicked.connect(self.simpanTransaksi)

self.input_given_money.setValidator(QIntValidator(1,
1_999_999_999, self))

self.input_given_money.editingFinished.connect(self.setChangeMoney
)

self.initItemTable()

def initSearchTable(self, keyword):
self.df_search = pd.read_csv('data/produk.csv')
df = self.df_search
query = (df["name"].str.contains(keyword, case=False)) |
(df["merk"].str.contains(keyword, case=False))
self.searchResult = df[query].reset_index(drop=True)

self.searchModel = SearchModel(self.searchResult)
self.tableView_search.setModel(self.searchModel)

def initItemTable(self):
self.df_temp_item = pd.read_csv('data/temp_item.csv')
self.itemModel = ItemModel(self.df_temp_item)
self.tableView_item.setModel(self.itemModel)
self.total_price = self.df_temp_item["total_price"].sum()
self.output_total_price.setText(str(self.total_price))

def search(self):
keyword = self.input_search.text()
if keyword != "":
self.initSearchTable(keyword)

def select(self):
if len(self.tableView_search.selectedIndexes()) > 0:
rows = sorted(set(index.row() for index in
self.tableView_search.selectedIndexes()))
row = rows[0]
data = self.searchResult.loc[row]
self.selectItem = SelectItem(data["id"])

self.selectItem.submitClicked.connect(self.initItemTable)
self.selectItem.show()

def hapusItem(self):
if len(self.tableView_item.selectedIndexes()) > 0:
rows = sorted(set(index.row() for index in
self.tableView_item.selectedIndexes()))
row = rows[0]

# mengembalikan stok
data_size = self.df_temp_item.loc[row]

```

```

        df_size = pd.read_csv('data/size.csv')
        df_size.loc[df_size["id"] == data_size["id_size"],
"stok"] += data_size["qty"]
        df_size.to_csv('data/size.csv', index=False)

        # menghapus item
        self.df_temp_item =
self.df_temp_item.drop(self.df_temp_item.index[row])
        self.df_temp_item.to_csv("data/temp_item.csv",
index=False)
        self.initItemTable()

    def setChangeMoney(self):
        given_money = self.input_given_money.text()
        self.output_change_money.setText(str(int(given_money) -
self.total_price))

    def simpanTransaksi(self):
        total_price = self.total_price
        given_money = self.input_given_money.text()
        change_money = int(given_money) - self.total_price
        date =
QDateTime.currentDateTime().toString(Qt.DateFormat.ISODate)
        operator = self.df_temp_user.loc[0, "username"]
        df_transaksi = pd.read_csv('data/transaksi.csv')

        if len(df_transaksi) == 0:
            new_id = 1
        else:
            new_id = df_transaksi.iloc[-1]["id"] + 1

        new_row = [int(new_id), total_price, given_money,
change_money, date, operator]
        df_transaksi.loc[len(df_transaksi)] = new_row
        df_transaksi.to_csv("data/transaksi.csv", index=False)

        self.makeTransaction(total_price)
        self.saveItems(new_id)
        self.removeTempItem()
        self.clearForm()
        self.detailTransaksi = DetailTransaksi(new_id)
        self.detailTransaksi.show()

    def saveItems(self, id_transaksi):
        df_item = pd.read_csv('data/item.csv')
        for i in range(len(self.df_temp_item)):
            if len(df_item) == 0:
                new_id = 1
            else:
                new_id = int(df_item.iloc[-1]["id"] + 1)
            id_transaction = int(id_transaksi)
            product_name = self.df_temp_item.loc[i, "name"]
            product_merk = self.df_temp_item.loc[i, "merk"]
            product_price = self.df_temp_item.loc[i, "price"]
            product_discount = self.df_temp_item.loc[i,
"discount"]
            qty = self.df_temp_item.loc[i, "qty"]
            size = self.df_temp_item.loc[i, "size"]
            total_price = self.df_temp_item.loc[i, "total_price"]

```

```

        new_row = [new_id, id_transaction, product_name,
product_merk, product_price, product_discount, qty, size,
                    total_price]
        df_item.loc[len(df_item)] = new_row
        df_item.to_csv("data/item.csv", index=False)

    def removeTempItem(self):
        self.df_temp_item.drop(self.df_temp_item.index,
inplace=True)
        self.df_temp_item.to_csv('data/temp_item.csv',
index=False)
        self.initItemTable()

    def clearForm(self):
        self.output_total_price.setText("0")
        self.input_given_money.clear()
        self.output_change_money.setText("-")

    def makeTransaction(self, amount):
        df = pd.read_csv('data/keuangan.csv')
        new_id = len(df) + 1
        date =
QDateTime.currentDateTime().toString(Qt.DateFormat.ISODate)
        operator = self.df_temp_user.loc[0, "username"]
        new_row = [new_id, "in", amount, date, operator]
        df.loc[new_id] = new_row
        df.to_csv("data/keuangan.csv", index=False)
        self.createNewTransaction.emit("ok")

```

kasir.py

```

import pandas as pd
from PyQt6 import uic
from PyQt6.QtCore import Qt, QAbstractTableModel
from PyQt6 import QtCore as Qtc
from PyQt6.QtGui import QIntValidator
from PyQt6.QtWidgets import QWidget

from warning import Warning

class PandasModel(QAbstractTableModel):
    def __init__(self, data, keyword):
        super(PandasModel, self).__init__()
        self._data = data
        self.keyword = keyword

    def rowCount(self, index):
        return self._data.shape[0]

    def columnCount(self, parnet=None):
        return self._data.shape[1]

    def data(self, index, role=Qt.ItemDataRole.DisplayRole):
        if index.isValid():
            if role == Qt.ItemDataRole.DisplayRole or role ==
Qt.ItemDataRole.EditRole:
                value = self._data.iloc[index.row(),
index.column()]
                return str(value)

```

```

def headerData(self, col, orientation, role):
    if orientation == Qt.Orientation.Horizontal and role ==
Qt.ItemDataRole.DisplayRole:
        return self._data.columns[col]

class SelectItem(QWidget):
    submitClicked = Qtc.pyqtSignal(str)
    def __init__(self, selected_id):
        super().__init__()
        self.selected_id = selected_id
        uic.loadUi('kasir_select_item.ui', self)
        self.initTable()
        self.button_tambah_item.clicked.connect(self.tambahItem)
        self.input_qty.setValidator(QIntValidator(1,
1_999_999_999, self))

    def initTable(self):
        self.df = pd.read_csv('data/size.csv')
        self.data_model = self.df[self.df["id_product"] ==
self.selected_id].reset_index(drop=True)
        self.model = PandasModel(self.data_model,
self.selected_id)
        self.tableView.setModel(self.model)

    def tambahItem(self):
        if len(self.tableView.selectedIndexes()) > 0:
            rows = sorted(set(index.row() for index in
self.tableView.selectedIndexes()))
            row = rows[0]
            data_size = self.data_model.loc[row]
            df_produk = pd.read_csv('data/produk.csv')
            data_produk = df_produk.loc[df_produk["id"] ==
self.selected_id].iloc[0]
            qty = self.input_qty.text()
            if qty == "":
                self.showWarning("Quantity tidak boleh kosong")
            else:
                qty = int(self.input_qty.text())
                if qty > data_size["stok"]:
                    print("stok tidak cukup")
                else:
                    final_total_price = ((100 -
data_produk["discount"]) / 100) * qty * data_produk["sell_price"]

                    new_row = [data_produk["name"],
data_produk["merk"],
data_size["size"],
qty,
data_produk["discount"],
data_produk["sell_price"],
"{:.2f}".format(final_total_price),
self.selected_id,
data_size["id"]]

                    # menyimpan ke temp item
                    df_temp_item =
pd.read_csv('data/temp_item.csv')
                    df_temp_item.loc[len(df_temp_item)] = new_row

```

```

df_temp_item.to_csv('data/temp_item.csv',
index=False)

# mengurangi stok
self.df.loc[self.df["id"] == data_size["id"],
"stok"] -= qty

self.df.to_csv('data/size.csv', index=False)

self.submitClicked.emit("ok")
self.close()

def showWarning(self, message):
    self.warning = Warning(message)
    self.warning.show()

```

kasir_select_item.py

Kode program ini adalah kode yang digunakan untuk menjalankan menu kasir. Pengguna dapat mencari dan memilih barang yang dibutuhkan, kemudian akan muncul tampilan daftar *size* dan stok yang tersedia untuk bisa dipilih kembali oleh pengguna. Barang yang telah dipilih akan muncul pada tabel daftar *item* yang telah tersedia di halaman kasir, yang mana pada daftar tersebut akan ditampilkan detail produk yang dipilih beserta total harga yang harus dibayar oleh pembeli. Fitur ini lebih dikenal dengan fitur keranjang. Pengguna juga bisa menghapus barang yang telah dipilih apabila dibutuhkan. Setelah selesai, pengguna dapat memasukkan uang yang diterima dari pembeli. Apabila uang tersebut melebihi total harga produk, maka akan muncul nominal uang yang harus dikembalikan kepada pembeli.

Pada menu kasir ini, dibutuhkan beberapa *file csv* pendukung yang dapat menampung data-data yang dibutuhkan. *File-file* tersebut diantaranya adalah *file* 'item.csv' untuk menyimpan data barang yang dibeli oleh pembeli, *file* 'keuangan.csv' untuk menyimpan data uang masuk dan keluar, *file* 'produk.csv' untuk menyimpan data-data produk yang dijual, *file* 'size.csv' untuk menyimpan data *size* dan stok barang, *file* 'temp_item.csv' untuk menyimpan data barang yang dimasukkan ke keranjang sebelum terjadinya transaksi, *file* 'temp_user.csv' untuk menyimpan data pengguna yang sedang melakukan *login*, dan *file* 'transaksi.csv' untuk menyimpan data transaksi yang telah dibuat. Berikut ini tampilan dari *file csv* pendukung yang dibutuhkan pada menu kasir.

```
id,id_transaction,product_name,product_merk,product_price,product_discount,qty,size,total_price
1,9,Adidas Blue,Adidas,1250000,5,2,38,2375000.0
2,9,Puma Futsal,Puma,1250000,0,4,40,5000000.0
3,2,Adidas Blue,Adidas Futsal,1250000,5,2,38,2375000.0
4,2,Adidas,Adidas Badminton,1250000,0,1,40,1250000.0
5,3,Adidas Blue,Adidas Futsal,1250000,5,1,38,1187500.0
6,3,Adidas,Adidas Badminton,1250000,0,2,40,2500000.0
7,4,Adidas Blue,Adidas Futsal,1250000,5,2,41,2375000.0
8,5,Adidas Blue,Adidas Futsal,1250000,5,2,42,2375000.0
9,6,Adidas,Adidas Badminton,1250000,0,4,40,5000000.0
10,9,Adidas Blue,Adidas Futsal,1250000,5,2,42,2375000.0
11,9,Adidas Blue,Adidas Futsal,1250000,5,2,41,2375000.0
12,10,Adidas Blue,Adidas Futsal,1250000,5,1,37,1187500.0
13,11,Adidas Blue,Adidas Futsal,1250000,5,2,43,2375000.0
14,12,Adidas Blue,Adidas Futsal,1250000,5,2,40,2375000.0
15,13,Adidas Blue,Adidas Futsal,1250000,5,1,41,1187500.0
```

item.csv

```
id,category,amount,date,operator
1,in,10000000.0,,
2,in,5000000.0,,
3,out,-1000000.0,,
4,out,-5000000.0,,
5,out,-5000000.0,,
6,out,-2000000.0,,
7,out,-2000000.0,,
8,out,-5000000.0,,
9,out,-2000000.0,,
10,out,-2000000.0,,
11,out,-2000000.0,,
12,out,-20000000.0,,
13,in,2375000.0,,
14,in,2375000.0,,
15,in,5000000.0,,
16,in,2375000.0,,
17,in,4750000.0,,
18,in,4750000.0,,
19,in,1187500.0,,
20,in,20000000.0,2023-06-16T17:09:48,
21,in,2375000.0,,
22,in,123456.0,2023-06-16T17:50:46,admin
23,in,2375000.0,2023-06-16T21:08:00,admin
24,in,1187500.0,2023-06-16T21:43:51,admin
25,out,-1000000.0,2023-06-16T22:37:10,admin
26,out,-1000000.0,2023-06-16T22:37:35,admin
27,out,-1000000.0,2023-06-17T09:01:32,admin
28,out,-1000000.0,2023-06-17T09:01:41,admin
29,in,500000.0,2023-06-17T09:02:32,admin
```

keuangan.csv

```
id,merk,name,buy_price,sell_price,discount,color
1,Adidas Futsal,Adidas Blue,1000000,1250000,5,red
2,Puma,Puma Futsal,1000000,1250000,0,red
3,Puma Badminton,Puma,1000000,1250000,0,blue
4,Adidas Badminton,Adidas,1000000,1250000,0,blue
5,Adidas Blue Badminton,Adidas,1000000,1250000,0,blue
6,Chuck Taylor All Star,Converse,1000000,1200000,0,black
```

produk.csv

```
id,id_product,size,stok
1,1,38,2
2,1,39,3
3,1,40,3
4,1,41,3
5,1,42,0
6,2,38,1
7,2,39,1
8,2,40,1
9,2,41,1
10,2,42,1
11,1,43,5
12,3,40,5
13,3,41,2
14,1,37,1
15,4,40,13
16,6,40,2
```

size.csv

```
name,merk,size,qty,discount,price,total_price,id_product,id_size
```

temp_item.csv

```
id,role,name,username
1,admin,0,admin
```

temp_user.csv

```
id,total_price,given_money,change_money,date,operator
1,7375000.0,7400000,25000.0,1,
2,3625000.0,3700000,75000.0,1,
3,3687500.0,3700000,12500.0,1,
4,2375000.0,2400000,25000.0,1,
5,2375000.0,2400000,25000.0,1,
6,5000000.0,5000000,0.0,1,
7,2375000.0,2400000,25000.0,1,
8,4750000.0,5000000,250000.0,1,
9,4750000.0,5000000,250000.0,1,
10,1187500.0,1200000,12500.0,1,
11,2375000.0,2400000,25000.0,2023-06-16T17:10:08,
12,2375000.0,2400000,25000.0,2023-06-16T21:08:00,admin
13,1187500.0,1200000,12500.0,2023-06-16T21:43:51,admin
```

transaksi.csv

3.4 Kode Program Menu Manajemen Produk

Subbab ini menjelaskan mengenai kode program yang digunakan dalam menjalankan menu manajemen produk pada program Point of Sales.

```
import pandas as pd
from PyQt6 import uic
from PyQt6 import QtCore as Qtc
from PyQt6.QtCore import Qt, QAbstractTableModel
from PyQt6.QtGui import QIntValidator
from PyQt6.QtWidgets import QWidget

from produk_edit import EditProduk
from produk_size import SizeProduk
```



```

from warning import Warning

class PandasModel(QAbstractTableModel):
    def __init__(self, data):
        super(PandasModel, self).__init__()
        self._data = data

    def rowCount(self, index):
        return self._data.shape[0]

    def columnCount(self, parent=None):
        return self._data.shape[1]

    def data(self, index, role=Qt.ItemDataRole.DisplayRole):
        if index.isValid():
            if role == Qt.ItemDataRole.DisplayRole or role == Qt.ItemDataRole.EditRole:
                value = self._data.iloc[index.row(), index.column()]
                return str(value)

    def headerData(self, col, orientation, role):
        if orientation == Qt.Orientation.Horizontal and role == Qt.ItemDataRole.DisplayRole:
            return self._data.columns[col]

class Produk(QWidget):
    makeTransactions = QtC.pyqtSignal(str)

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        uic.loadUi('produk.ui', self)
        self.input_baru_buy_price.setValidator(QIntValidator(1, 1_999_999_999, self))
        self.input_baru_sell_price.setValidator(QIntValidator(1, 1_999_999_999, self))
        self.input_baru_discount.setValidator(QIntValidator(1, 1_999_999_999, self))

    self.button_tambah_produk.clicked.connect(self.tambahProduk)
    self.button_hapus_produk.clicked.connect(self.hapusProduk)
    self.button_edit_produk.clicked.connect(self.ubahProduk)
    self.button_stok_produk.clicked.connect(self.stokProduk)

    self.initTable()
    self.df_size = pd.read_csv('data/size.csv')

    def initTable(self):
        self.df_produk = pd.read_csv('data/produk.csv')
        self.model = PandasModel(self.df_produk)
        self.tableView.setModel(self.model)

    def clearForm(self):
        self.input_baru_name.clear()
        self.input_baru_merk.clear()
        self.input_baru_buy_price.clear()
        self.input_baru_sell_price.clear()

```

```

self.input_baru_discount.clear()
self.input_baru_color.clear()

def tambahProduk(self):
    name = self.input_baru_name.text()
    merk = self.input_baru_merk.text()
    buy_price = self.input_baru_buy_price.text()
    sell_price = self.input_baru_sell_price.text()
    discount = self.input_baru_discount.text()
    color = self.input_baru_color.text()

    is_empty = name == "" or merk == "" or buy_price == "" or
sell_price == "" or discount == "" or color == ""

    if is_empty:
        self.showWarning("Form tidak boleh kosong")
    else:
        buy_price = int(self.input_baru_buy_price.text())
        sell_price = int(self.input_baru_sell_price.text())
        discount = int(self.input_baru_discount.text())

        if len(self.df_produk) == 0:
            new_id = 1
        else:
            new_id = int(self.df_produk.iloc[-1]["id"] + 1)

        new_row = [new_id, name, merk, buy_price, sell_price,
discount, color]

        self.df_produk.loc[len(self.df_produk)] = new_row
        self.clearForm()
        self.df_produk.to_csv("data/produk.csv", index=False)
        self.initTable()

def hapusProduk(self):
    if len(self.tableView.selectedIndexes()) > 0:
        rows = sorted(set(index.row() for index in
self.tableView.selectedIndexes()))
        row = rows[0]

        self.df_produk =
self.df_produk.drop(self.df_produk.index[row])
        self.df_produk.to_csv("data/produk.csv", index=False)
        self.initTable()

def ubahProduk(self):
    if len(self.tableView.selectedIndexes()) > 0:
        rows = sorted(set(index.row() for index in
self.tableView.selectedIndexes()))
        row = rows[0]
        data = self.df_produk.loc[row]

        self.edit_produk = EditProduk()
        self.edit_produk.input_edit_row.setHidden(True)
        self.edit_produk.input_edit_row.setText(str(row))

        self.edit_produk.input_edit_name.setText(data["name"])
        self.edit_produk.input_edit_merk.setText(data["merk"])

```

```

self.edit_produk.input_edit_sell_price.setText(str(data["sell_price"]))

self.edit_produk.input_edit_discount.setText(str(data["discount"]))

self.edit_produk.input_edit_color.setText(data["color"])

        self.edit_produk.submitClicked.connect(self.initTable)
        self.edit_produk.displayInfo()

    def stokProduk(self):
        if len(self.tableView.selectedIndexes()) > 0:
            rows = sorted(set(index.row() for index in
self.tableView.selectedIndexes()))
            row = rows[0]
            data = self.df_produk.loc[row]
            self.sizeProduk = SizeProduk(data[0])
            self.sizeProduk.show()
            self.sizeProduk.makeRestock.connect(self.sendSignal)

    def showWarning(self, message):
        self.warning = Warning(message)
        self.warning.show()

    def sendSignal(self):
        self.makeTransactions.emit("ok")

```

produk.py

Kode program di atas merupakan kode program yang berfungsi untuk menjalankan menu manajemen produk pada program. Pada menu ini, pengguna dapat menambahkan produk baru pada kolom-kolom yang tersedia. Pengguna juga dapat menghapus produk apabila dibutuhkan dengan memencet tombol hapus produk yang tercantum di menu ini.

```

import pandas as pd
from PyQt6 import uic
from PyQt6 import QtCore as Qtc
from PyQt6.QtWidgets import QWidget

from warning import Warning

class EditProduk(QWidget):
    submitClicked = Qtc.pyqtSignal(str)

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        uic.loadUi('produk_edit.ui', self)
        self.df = pd.read_csv('data/produk.csv')
        self.button_edit_produk.clicked.connect(self.saveProduk)

    def displayInfo(self):

```

```

self.show()

def saveProduk(self):
    row = self.input_edit_row.text()
    data = self.df.loc[int(row)]
    name = self.input_edit_name.text()
    merk = self.input_edit_merk.text()
    sell_price = self.input_edit_sell_price.text()
    discount = self.input_edit_discount.text()
    color = self.input_edit_color.text()

    is_empty = name == "" or merk == "" or sell_price == "" or
discount == "" or color == ""

    if is_empty:
        self.showWarning("Form tidak boleh kosong")
    else:
        sell_price = int(self.input_edit_sell_price.text())
        discount = int(self.input_edit_discount.text())
        edited_row = [data["id"], name, merk,
data["buy_price"], sell_price, discount, color]
        self.df.loc[int(row)] = edited_row
        self.df.to_csv("data/produk.csv", index=False)
        self.submitClicked.emit("ok")
        self.close()

def showWarning(self, message):
    self.warning = Warning(message)
    self.warning.show()

```

produk_edit.py

Kode program di atas berfungsi untuk mengubah detail produk yang sudah ada sesuai dengan kebutuhan pengguna. Perubahan tersebut dapat dilakukan dengan memencet tombol ubah produk yang ada pada menu manajemen produk.

```

import pandas as pd
from PyQt6 import uic
from PyQt6 import QtCore as Qtc
from PyQt6.QtCore import Qt, QAbstractTableModel, QDateTime
from PyQt6.QtGui import QIntValidator
from PyQt6.QtWidgets import QWidget

from warning import Warning

class PandasModel(QAbstractTableModel):
    def __init__(self, data, selected_id):
        super(PandasModel, self).__init__()
        self._data = data
        self.selected_id = selected_id

    def rowCount(self, index):
        return len(self._data.loc[self._data["id_product"] ==
self.selected_id])

```

```

def columnCount(self, parnet=None):
    return self._data.shape[1]

def data(self, index, role=Qt.ItemDataRole.DisplayRole):
    if index.isValid():
        if role == Qt.ItemDataRole.DisplayRole or role == Qt.ItemDataRole.EditRole:
            value = self._data.loc[self._data["id_product"] == self.selected_id].iloc[index.row(), index.column()]
            return str(value)

def headerData(self, col, orientation, role):
    if orientation == Qt.Orientation.Horizontal and role == Qt.ItemDataRole.DisplayRole:
        return self._data.columns[col]

class SizeProduk(QWidget):
    makeRestock = Qt.pyqtSignal(str)

    def __init__(self, selected_id):
        super().__init__()
        self.selected_id = selected_id
        uic.loadUi('produk_size.ui', self)
        self.df_temp_user = pd.read_csv('data/temp_user.csv')

        self.initTable()

        self.input_baru_size.setValidator(QIntValidator(1, 1_999_999_999, self))
        self.input_baru_stok.setValidator(QIntValidator(1, 1_999_999_999, self))
        self.input_restock.setValidator(QIntValidator(1, 1_999_999_999, self))
        self.groupBox_restock.setVisible(False)

        self.button_tambah_size.clicked.connect(self.tambahSize)
        self.button_restock.clicked.connect(self.restock)

        df_produk = pd.read_csv('data/produk.csv')
        self.buy_price = df_produk.loc[df_produk["id"] == selected_id].iloc[0]["buy_price"]

    def initTable(self):
        self.df = pd.read_csv('data/size.csv')
        self.model = PandasModel(self.df, self.selected_id)
        self.tableView.setModel(self.model)

    def tambahSize(self):
        size = self.input_baru_size.text()
        stok = self.input_baru_stok.text()

        if size == "" or stok == "":
            self.showWarning("Size / Stok tidak boleh kosong")
        elif len(self.df.loc[self.df["id_product"] == self.selected_id].loc[self.df["size"] == int(size)]) > 0:
            self.showWarning("Size yang sama sudah ada")
        else:
            size = int(self.input_baru_size.text())

```

```

        stok = int(self.input_baru_stok.text())

        current_saldo = int(self.getSaldo())

        if current_saldo < int(stok * self.buy_price):
            self.showWarning("Saldo tidak cukup")
        else:
            if len(self.df) == 0:
                new_id = 1
            else:
                new_id = int(self.df.iloc[-1]["id"] + 1)

            new_row = [new_id, self.selected_id, size, stok]

            self.df.loc[len(self.df)] = new_row
            self.makeTransaction(int(stok * self.buy_price))
            self.input_baru_size.clear()
            self.input_baru_stok.clear()
            self.df.to_csv("data/size.csv", index=False)
            self.initTable()

    def restock(self):
        if len(self.tableView.selectedIndexes()) > 0:
            self.groupBox_restock.setVisible(True)

self.button_restock_save.clicked.connect(self.restockSave)

    def restockSave(self):
        if len(self.tableView.selectedIndexes()) > 0:
            rows = sorted(set(index.row() for index in
self.tableView.selectedIndexes()))
            row = rows[0]
            temp_df = self.df.sort_values(by=['id_product'])
            temp_id = temp_df.loc[temp_df["id_product"] ==
self.selected_id].reset_index(drop=True).loc[row]["id"]
            data = self.df.loc[self.df["id"] == temp_id].iloc[0]

            added_stock = self.input_restock.text()
            if added_stock == "":
                self.showWarning("Input Stok tidak boleh kosong")
            else:
                added_stock = int(self.input_restock.text())
                current_saldo = int(self.getSaldo())

                if current_saldo < int(added_stock *
self.buy_price):
                    self.showWarning("Saldo tidak cukup")
                else:
                    edited_row = [data["id"], data["id_product"],
data["size"], int(data["stok"] + added_stock)]

                    self.df.loc[self.df["id"] == data["id"]] =
edited_row

                    self.df.to_csv("data/size.csv", index=False)
                    self.makeTransaction(int(added_stock *
self.buy_price))

                    self.input_restock.clear()
                    self.groupBox_restock.setVisible(False)
                    self.initTable()

```

```

def getSaldo(self):
    df = pd.read_csv('data/keuangan.csv')
    return df["amount"].sum()

def makeTransaction(self, amount):
    df = pd.read_csv('data/keuangan.csv')
    new_id = len(df) + 1
    date =
QDateTime.currentDateTime().toString(Qt.DateFormat.ISODate)
operator = self.df_temp_user.loc[0, "username"]
new_row = [new_id, "out", 0-amount, date, operator]
df.loc[new_id] = new_row
df.to_csv("data/keuangan.csv", index=False)
self.makeRestock.emit("ok")

def showWarning(self, message):
    self.warning = Warning(message)
    self.warning.show()

```

produk_size.py

Kode program tersebut digunakan untuk melakukan penambahan maupun perubahan pada stok produk beserta *size* yang tersedia. Fitur ini dapat diakses pada fitur stok dan *size* yang ada pada manajemen produk. Menu ini tentu juga memerlukan beberapa file csv pendukung, diantaranya file 'produk.csv', file 'size.csv', file 'keuangan.csv', dan file 'temp_user.csv'. Tampilan dari semua file tersebut ada di bawah ini.

```

id,merk,name,buy_price,sell_price,discount,color
1,Adidas Futsal,Adidas Blue,1000000,1250000,5,red
2,Puma,Puma Futsal,1000000,1250000,0,red
3,Puma Badminton,Puma,1000000,1250000,0,blue
4,Adidas Badminton,Adidas,1000000,1250000,0,blue
5,Adidas Blue Badminton,Adidas,1000000,1250000,0,blue
6,Chuck Taylor All Star,Converse,1000000,1200000,0,black

```

produk.csv

```

id,id_product,size,stok
1,1,38,2
2,1,39,3
3,1,40,3
4,1,41,3
5,1,42,0
6,2,38,1
7,2,39,1
8,2,40,1
9,2,41,1
10,2,42,1
11,1,43,5
12,3,40,5
13,3,41,2

```

```
14,1,37,1
15,4,40,13
16,6,40,2
```

size.csv

```
id,category,amount,date,operator
1,in,100000000.0,,
2,in,50000000.0,,
3,out,-1000000.0,,
4,out,-5000000.0,,
5,out,-5000000.0,,
6,out,-2000000.0,,
7,out,-2000000.0,,
8,out,-5000000.0,,
9,out,-2000000.0,,
10,out,-2000000.0,,
11,out,-2000000.0,,
12,out,-20000000.0,,
13,in,2375000.0,,
14,in,2375000.0,,
15,in,5000000.0,,
16,in,2375000.0,,
17,in,4750000.0,,
18,in,4750000.0,,
19,in,1187500.0,,
20,in,20000000.0,2023-06-16T17:09:48,
21,in,2375000.0,,
22,in,123456.0,2023-06-16T17:50:46,admin
23,in,2375000.0,2023-06-16T21:08:00,admin
24,in,1187500.0,2023-06-16T21:43:51,admin
25,out,-1000000.0,2023-06-16T22:37:10,admin
26,out,-1000000.0,2023-06-16T22:37:35,admin
27,out,-1000000.0,2023-06-17T09:01:32,admin
28,out,-1000000.0,2023-06-17T09:01:41,admin
29,in,500000.0,2023-06-17T09:02:32,admin
```

keuangan.csv

```
id,role,name,username
1,admin,0,admin
```

temp_user.csv

3.5 Kode Program Menu Manajemen Keuangan

Subbab ini menjelaskan mengenai kode program yang digunakan dalam menjalankan menu manajemen keuangan pada program Point of Sales.

```
import pandas as pd
from PyQt6 import uic
from PyQt6.QtCore import Qt, QAbstractTableModel, QDateTime
from PyQt6.QtGui import QIntValidator
from PyQt6.QtWidgets import QWidget

from warning import Warning

class PandasModel(QAbstractTableModel):
```



```

def __init__(self, data):
    super(PandasModel, self).__init__()
    self._data = data

def rowCount(self, index):
    return self._data.shape[0]

def columnCount(self, parent=None):
    return self._data.shape[1]

def data(self, index, role=Qt.ItemDataRole.DisplayRole):
    if index.isValid():
        if role == Qt.ItemDataRole.DisplayRole or role ==
Qt.ItemDataRole.EditRole:
            value = self._data.iloc[index.row(),
index.column()]
            return str(value)

    def headerData(self, col, orientation, role):
        if orientation == Qt.Orientation.Horizontal and role ==
Qt.ItemDataRole.DisplayRole:
            return self._data.columns[col]

class Keuangan(QWidget):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        uic.loadUi('keuangan.ui', self)
        self.df_temp_user = pd.read_csv('data/temp_user.csv')
        self.initTable()

    def initTable(self):
        self.df = pd.read_csv('data/keuangan.csv')
        self.model = PandasModel(self.df)
        self.tableView.setModel(self.model)

        self.input_tambah_saldo.setValidator(QIntValidator(1,
1_999_999_999, self))
        self.input_tarik_saldo.setValidator(QIntValidator(1,
1_999_999_999, self))

        self.button_tambah_saldo.clicked.connect(self.tambahSaldo)
        self.button_tarik_saldo.clicked.connect(self.tarikSaldo)

        self.label_saldo.setText(str(self.hitungSaldo()))

    def tambahSaldo(self):
        amount = self.input_tambah_saldo.text()
        if amount is None or amount == "":
            self.showWarning("Amount tidak boleh kosong")
        else:
            amount = int(self.input_tambah_saldo.text())
            new_id = len(self.df) + 1
            date =
QDateTime.currentDateTime().toString(Qt.DateFormat.ISODate)
            operator = self.df_temp_user.loc[0, "username"]
            new_row = [new_id, "in", amount, date, operator]
            self.df.loc[new_id] = new_row
            self.input_tambah_saldo.clear()

```

```

        self.df.to_csv("data/keuangan.csv", index=False)
        self.initTable()

    def tarikSaldo(self):
        amount = self.input_tarik_saldo.text()
        if amount is None or amount == "":
            self.showWarning("Amount tidak boleh kosong")
        else:
            amount = 0 - int(self.input_tarik_saldo.text())
            new_id = len(self.df) + 1
            date =
QDateTime.currentDateTime().toString(Qt.DateFormat.ISODate)
            operator = self.df_temp_user.loc[0, "username"]
            new_row = [new_id, "out", amount, date, operator]
            self.df.loc[new_id] = new_row
            self.input_tarik_saldo.clear()
            self.df.to_csv("data/keuangan.csv", index=False)
            self.initTable()

    def hitungSaldo(self):
        return self.df["amount"].sum()

    def showWarning(self, message):
        self.warning = Warning(message)
        self.warning.show()

```

keuangan.py

Pada menu manajemen keuangan, terdapat tabel yang mencatat segala aktivitas keluar-masuknya uang, baik dari penambahan atau pengurangan modal maupun dari transaksi yang terjadi. Pengguna juga dapat melakukan penambahan ataupun penarikan modal di menu ini dengan cara mengisi jumlah saldo yang diinginkan. Menu ini membutuhkan *file csv* pendukung yang terdiri atas *file* 'keuangan.csv' dan *file* 'temp_user.csv'. Di bawah ini adalah tampilan dari kedua *file* tersebut.

```

id,category,amount,date,operator
1,in,100000000.0,,
2,in,50000000.0,,
3,out,-1000000.0,,
4,out,-5000000.0,,
5,out,-5000000.0,,
6,out,-2000000.0,,
7,out,-2000000.0,,
8,out,-5000000.0,,
9,out,-2000000.0,,
10,out,-2000000.0,,
11,out,-2000000.0,,
12,out,-20000000.0,,
13,in,2375000.0,,
14,in,2375000.0,,
15,in,5000000.0,,

```

```

16,in,2375000.0,,
17,in,4750000.0,,
18,in,4750000.0,,
19,in,1187500.0,,
20,in,20000000.0,2023-06-16T17:09:48,
21,in,2375000.0,,
22,in,123456.0,2023-06-16T17:50:46,admin
23,in,2375000.0,2023-06-16T21:08:00,admin
24,in,1187500.0,2023-06-16T21:43:51,admin
25,out,-1000000.0,2023-06-16T22:37:10,admin
26,out,-1000000.0,2023-06-16T22:37:35,admin
27,out,-1000000.0,2023-06-17T09:01:32,admin
28,out,-1000000.0,2023-06-17T09:01:41,admin
29,in,500000.0,2023-06-17T09:02:32,admin

```

keuangan.csv

```

id,role,name,username
1,admin,0,admin

```

temp_user.csv

3.6 Kode Program Menu Manajemen Transaksi

Subbab ini menjelaskan mengenai kode program yang digunakan dalam menjalankan menu manajemen transaksi pada program Point of Sales.

```

import pandas as pd
from PyQt6 import uic
from PyQt6.QtCore import Qt, QAbstractTableModel
from PyQt6.QtWidgets import QWidget

from transaksi_detail import DetailTransaksi

class PandasModel(QAbstractTableModel):
    def __init__(self, data):
        super(PandasModel, self).__init__()
        self._data = data

    def rowCount(self, index):
        return self._data.shape[0]

    def columnCount(self, parent=None):
        return self._data.shape[1]

    def data(self, index, role=Qt.ItemDataRole.DisplayRole):
        if index.isValid():
            if role == Qt.ItemDataRole.DisplayRole or role == Qt.ItemDataRole.EditRole:
                value = self._data.iloc[index.row(), index.column()]
                return str(value)

    def headerData(self, col, orientation, role):
        if orientation == Qt.Orientation.Horizontal and role == Qt.ItemDataRole.DisplayRole:

```

```

        return self._data.columns[col]

class Transaksi(QWidget):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        uic.loadUi('transaksi.ui', self)
        self.initTable()

    def initTable(self):
        self.df = pd.read_csv('data/transaksi.csv')
        self.model = PandasModel(self.df)
        self.tableView.setModel(self.model)

self.button_detail_transaksi.clicked.connect(self.detailTransaksi)

    def detailTransaksi(self):
        if len(self.tableView.selectedIndexes()) > 0:
            rows = sorted(set(index.row() for index in
self.tableView.selectedIndexes()))
            row = rows[0]
            data = self.df.loc[row]
            self.detailTransaksi = DetailTransaksi(data[0])
            self.detailTransaksi.show()

```

transaksi.py

```

import pandas as pd
from PyQt6 import uic
from PyQt6.QtCore import Qt, QAbstractTableModel
from PyQt6.QtWidgets import QWidget

class PandasModel(QAbstractTableModel):
    def __init__(self, data, selected_id):
        super(PandasModel, self).__init__()
        self._data = data
        self.selected_id = selected_id

    def rowCount(self, index):
        return len(self._data.loc[self._data["id_transaction"] ==
self.selected_id])

    def columnCount(self, parent=None):
        return self._data.shape[1]

    def data(self, index, role=Qt.ItemDataRole.DisplayRole):
        if index.isValid():
            if role == Qt.ItemDataRole.DisplayRole or role ==
Qt.ItemDataRole.EditRole:
                value =
self._data.loc[self._data["id_transaction"] ==
self.selected_id].iloc[index.row(), index.column()]
                return str(value)

    def headerData(self, col, orientation, role):
        if orientation == Qt.Orientation.Horizontal and role ==
Qt.ItemDataRole.DisplayRole:
            return self._data.columns[col]

```

```

class DetailTransaksi(Qwidget):
    def __init__(self, selected_id):
        super().__init__()
        self.selected_id = selected_id
        uic.loadUi('transaksi_detail.ui', self)
        self.initTable()
        self.initDetailInfo()

    def initTable(self):
        self.df_item = pd.read_csv('data/item.csv')
        self.model = PandasModel(self.df_item, self.selected_id)
        self.tableView.setModel(self.model)

    def initDetailInfo(self):
        self.df_transaksi = pd.read_csv('data/transaksi.csv')
        data = self.df_transaksi.loc[self.df_transaksi["id"] ==
self.selected_id].iloc[0]
        self.output_id.setText(str(data["id"]))
        self.output_total_price.setText(str(data["total_price"]))
        self.output_date.setText(str(data["date"]))
        self.output_given_money.setText(str(data["given_money"]))

self.output_change_money.setText(str(data["change_money"]))
        self.output_operator.setText(str(data["operator"]))

```

transaksi_detail.py

Kode program di atas berguna untuk menjalankan menu manajemen transaksi. Di dalamnya terdapat tabel yang mencatat terjadinya transaksi atas barang yang dibeli oleh pembeli. Pengguna juga dapat melihat detail transaksi yang terjadi melalui fitur tombol detail yang ada pada program. Menu ini membutuhkan *file-file csv* pendukung diantaranya *file* 'transaksi.csv' dan *file* 'item.csv' yang ditampilkan di bawah ini.

```

id,total_price,given_money,change_money,date,operator
1,7375000.0,7400000,25000.0,1,
2,3625000.0,3700000,75000.0,1,
3,3687500.0,3700000,12500.0,1,
4,2375000.0,2400000,25000.0,1,
5,2375000.0,2400000,25000.0,1,
6,5000000.0,5000000,0.0,1,
7,2375000.0,2400000,25000.0,1,
8,4750000.0,5000000,250000.0,1,
9,4750000.0,5000000,250000.0,1,
10,1187500.0,1200000,12500.0,1,
11,2375000.0,2400000,25000.0,2023-06-16T17:10:08,
12,2375000.0,2400000,25000.0,2023-06-16T21:08:00,admin
13,1187500.0,1200000,12500.0,2023-06-16T21:43:51,admin

```

transaksi.csv

```

id,id_transaction,product_name,product_merk,product_price,product_
discount,qty,size,total_price

```

```

1,9,Adidas Blue,Adidas,1250000,5,2,38,2375000.0
2,9,Puma Futsal,Puma,1250000,0,4,40,5000000.0
3,2,Adidas Blue,Adidas Futsal,1250000,5,2,38,2375000.0
4,2,Adidas,Adidas Badminton,1250000,0,1,40,1250000.0
5,3,Adidas Blue,Adidas Futsal,1250000,5,1,38,1187500.0
6,3,Adidas,Adidas Badminton,1250000,0,2,40,2500000.0
7,4,Adidas Blue,Adidas Futsal,1250000,5,2,41,2375000.0
8,5,Adidas Blue,Adidas Futsal,1250000,5,2,42,2375000.0
9,6,Adidas,Adidas Badminton,1250000,0,4,40,5000000.0
10,9,Adidas Blue,Adidas Futsal,1250000,5,2,42,2375000.0
11,9,Adidas Blue,Adidas Futsal,1250000,5,2,41,2375000.0
12,10,Adidas Blue,Adidas Futsal,1250000,5,1,37,1187500.0
13,11,Adidas Blue,Adidas Futsal,1250000,5,2,43,2375000.0
14,12,Adidas Blue,Adidas Futsal,1250000,5,2,40,2375000.0
15,13,Adidas Blue,Adidas Futsal,1250000,5,1,41,1187500.0

```

item.csv

3.7 Kode Program Menu Manajemen Akun Pegawai

Subbab ini menjelaskan mengenai kode program yang digunakan dalam menjalankan menu manajemen akun pegawai pada program Point of Sales.

```

import pandas as pd
from PyQt6 import uic
from PyQt6.QtCore import Qt, QAbstractTableModel
from PyQt6.QtWidgets import QWidget

from warning import Warning

class PandasModel(QAbstractTableModel):
    def __init__(self, data):
        super(PandasModel, self).__init__()
        self._data = data

    def rowCount(self, index):
        return self._data.shape[0]

    def columnCount(self, parent=None):
        return self._data.shape[1]

    def data(self, index, role=Qt.ItemDataRole.DisplayRole):
        if index.isValid():
            if role == Qt.ItemDataRole.DisplayRole or role == Qt.ItemDataRole.EditRole:
                value = self._data.iloc[index.row(), index.column()]
                return str(value)

    def headerData(self, col, orientation, role):
        if orientation == Qt.Orientation.Horizontal and role == Qt.ItemDataRole.DisplayRole:
            return self._data.columns[col]

```

```

class Akun(Qwidget):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        uic.loadUi('akun.ui', self)
        self.groupBox_edit_akun.setVisible(False)
        self.df = pd.read_csv('data/user.csv')
        self.initTable()

    def initTable(self):
        self.model = PandasModel(self.df)
        self.tableView.setModel(self.model)
        self.button_tambah_akun.clicked.connect(self.tambahAkun)
        self.button_edit_akun.clicked.connect(self.ubahAkun)
        self.button_hapus_akun.clicked.connect(self.hapusAkun)

    def tambahAkun(self):
        name = self.input_tambah_name.text()
        username = self.input_tambah_username.text()
        password = self.input_tambah_password.text()
        if name == "" or username == "" or password == "":
            self.showWarning("Nama / Username / Password tidak boleh kosong")
        else:
            if len(self.df) == 0:
                new_id = 1
            else:
                new_id = self.df.iloc[-1]["id"] + 1

            new_row = [new_id, "cashier", name, username, password]

            self.df.loc[len(self.df)] = new_row
            self.df.to_csv("data/user.csv", index=False)
            self.input_tambah_name.clear()
            self.input_tambah_username.clear()
            self.input_tambah_password.clear()
            self.initTable()

    def hapusAkun(self):
        if len(self.tableView.selectedIndexes()) > 0:
            rows = sorted(set(index.row() for index in self.tableView.selectedIndexes()))
            row = rows[0]

            self.df = self.df.drop(self.df.index[row])
            self.df.to_csv("data/user.csv", index=False)
            self.initTable()

    def ubahAkun(self):
        if len(self.tableView.selectedIndexes()) > 0:
            rows = sorted(set(index.row() for index in self.tableView.selectedIndexes()))
            row = rows[0]
            data = self.df.loc[row]
            self.input_edit_name.setText(data["name"])
            self.input_edit_username.setText(data["username"])
            self.input_edit_password.setText(data["password"])

            self.groupBox_edit_akun.setVisible(True)

```

```

self.button_save_edit_akun.clicked.connect(self.editAkunSave)

def editAkunSave(self):
    if len(self.tableView.selectedIndexes()) > 0:
        rows = sorted(set(index.row() for index in
self.tableView.selectedIndexes()))
        row = rows[0]
        data = self.df.loc[row]

        name = self.input_edit_name.text()
        username = self.input_edit_username.text()
        password = self.input_edit_password.text()

        if name == "" or username == "" or password == "":
            self.showWarning("Nama / Username / Password tidak
boleh kosong")
        else:
            edited_row = [data["id"], data["role"], name,
username, password]
            self.df.loc[row] = edited_row
            self.df.to_csv("data/user.csv", index=False)
            self.input_edit_name.clear()
            self.input_edit_username.clear()
            self.input_edit_password.clear()
            self.groupBox_edit_akun.setVisible(False)
            self.initTable()

def showWarning(self, message):
    self.warning = Warning(message)
    self.warning.show()

```

akun.py

Kode program tersebut digunakan untuk menjalankan menu manajemen akun yang ada pada program Point of Sales. Pada menu ini, pengguna dapat membuat akun baru, menghapus akun, dan melakukan perubahan pada akun yang sudah ada. Tentunya dibutuhkan *file csv* pendukung berupa *file* 'user.csv' yang ditampilkan seperti di bawah ini.

```

id,role,name,username,password
1,admin,admin,admin,admin123
2,cashier,cashier 1,cashier1,cashier123
3,cashier,cashier 2,cashier2,cashier123
4,cashier,cashier 3,cashier3,cashier123

```

user.csv

3.8 Kode Program Laman Warning

Subbab ini menjelaskan mengenai kode program yang digunakan dalam menjalankan fitur *warning* atau peringatan pada program Point of Sales.


```

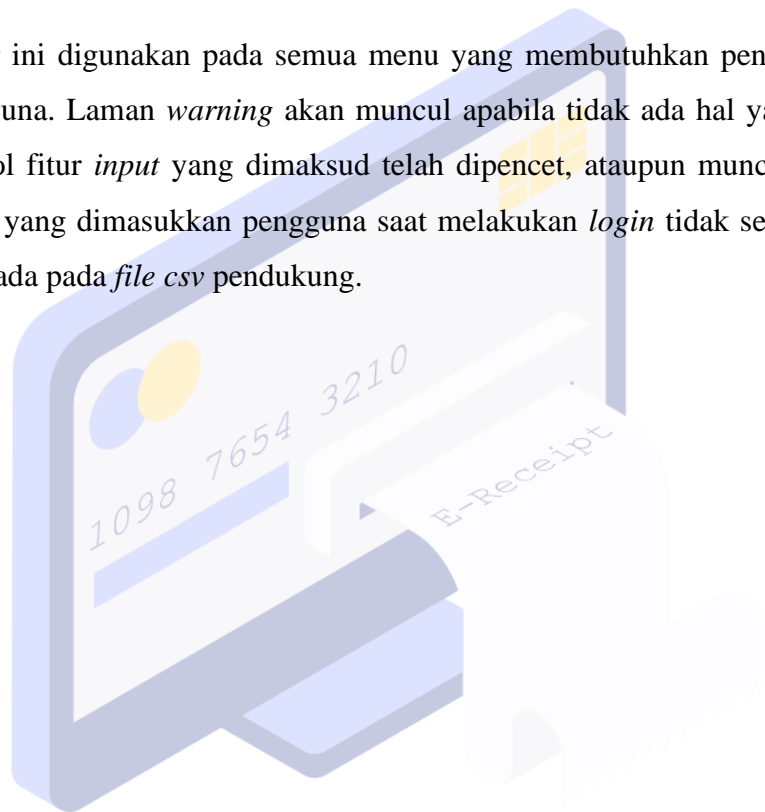
from PyQt6 import uic
from PyQt6.QtWidgets import QWidget

class Warning(QWidget):
    def __init__(self, message):
        super().__init__()
        self.message = message
        print(self.message)
        uic.loadUi('warning.ui', self)
        self.label_warning.setText(str(self.message))
        self.button_close.clicked.connect(self.close)

```

warning.py

Fitur ini digunakan pada semua menu yang membutuhkan pengisian *input* dari pengguna. Laman *warning* akan muncul apabila tidak ada hal yang di-*input* saat tombol fitur *input* yang dimaksud telah dipencet, ataupun muncul saat data kredensial yang dimasukkan pengguna saat melakukan *login* tidak sesuai dengan data yang ada pada *file csv* pendukung.



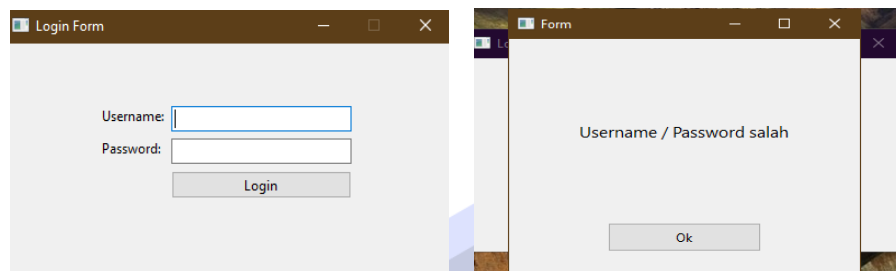
BAB IV

HASIL RUNNING PROGRAM

Bab ini menjelaskan mengenai hasil *running* dari program Point of Sales.

4.1 Hasil Running Autentikasi Login

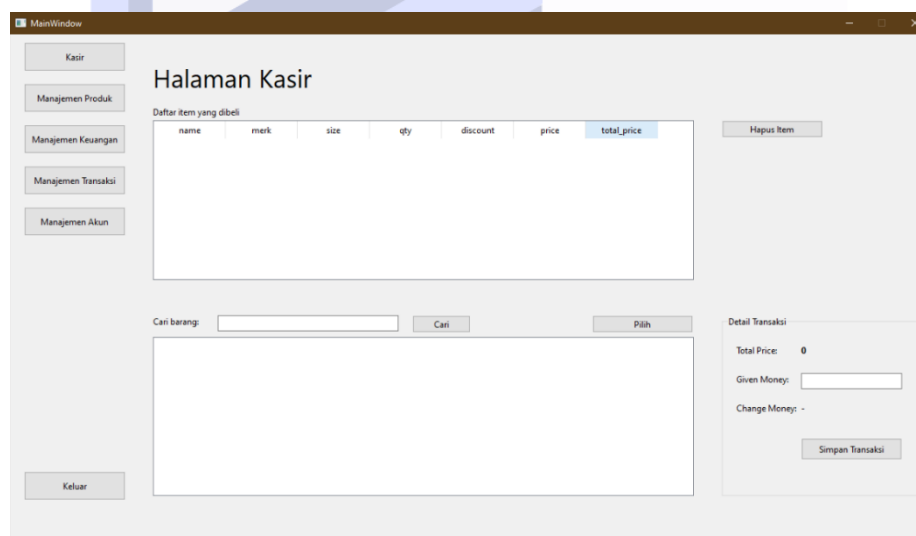
Subbab ini menjelaskan mengenai hasil *running* dari autentikasi *login* pada program Point of Sales.

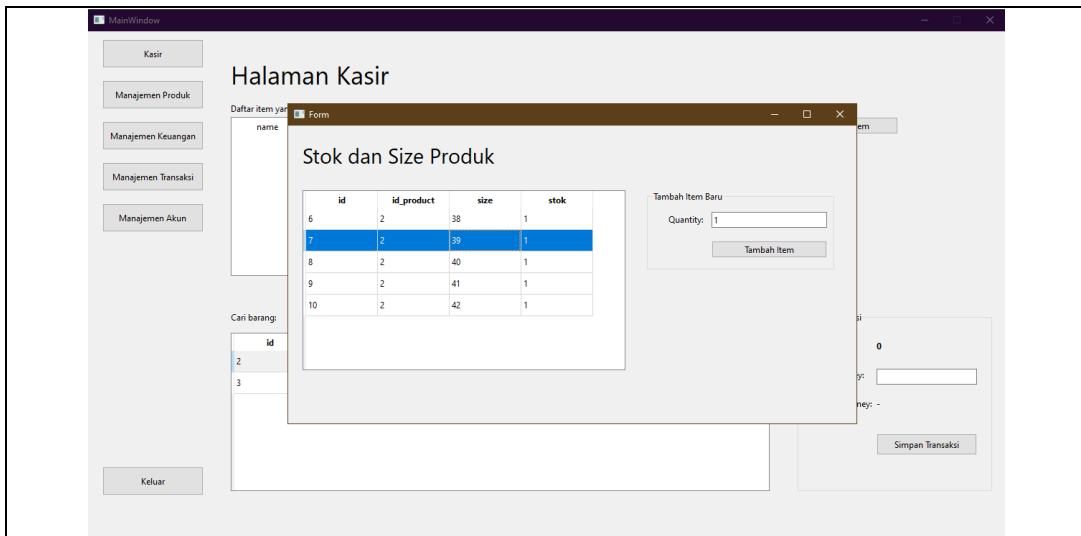


Gambar diatas adalah tampilan UI Login Form(kiri) yang nantinya user diminta untuk memasukkan username dan passwordnya, kemudian jika password atau username yang dimasukkan tidak sesuai maka akan muncul peringatan “Username/Password salah”.

4.2 Hasil Running Menu Kasir

Subbab ini menjelaskan mengenai hasil *running* dari menu kasir pada program Point of Sales.

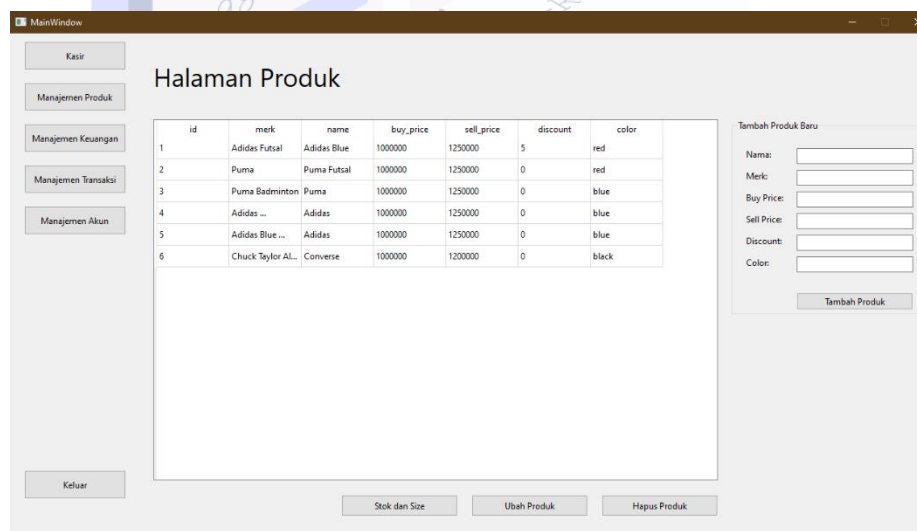


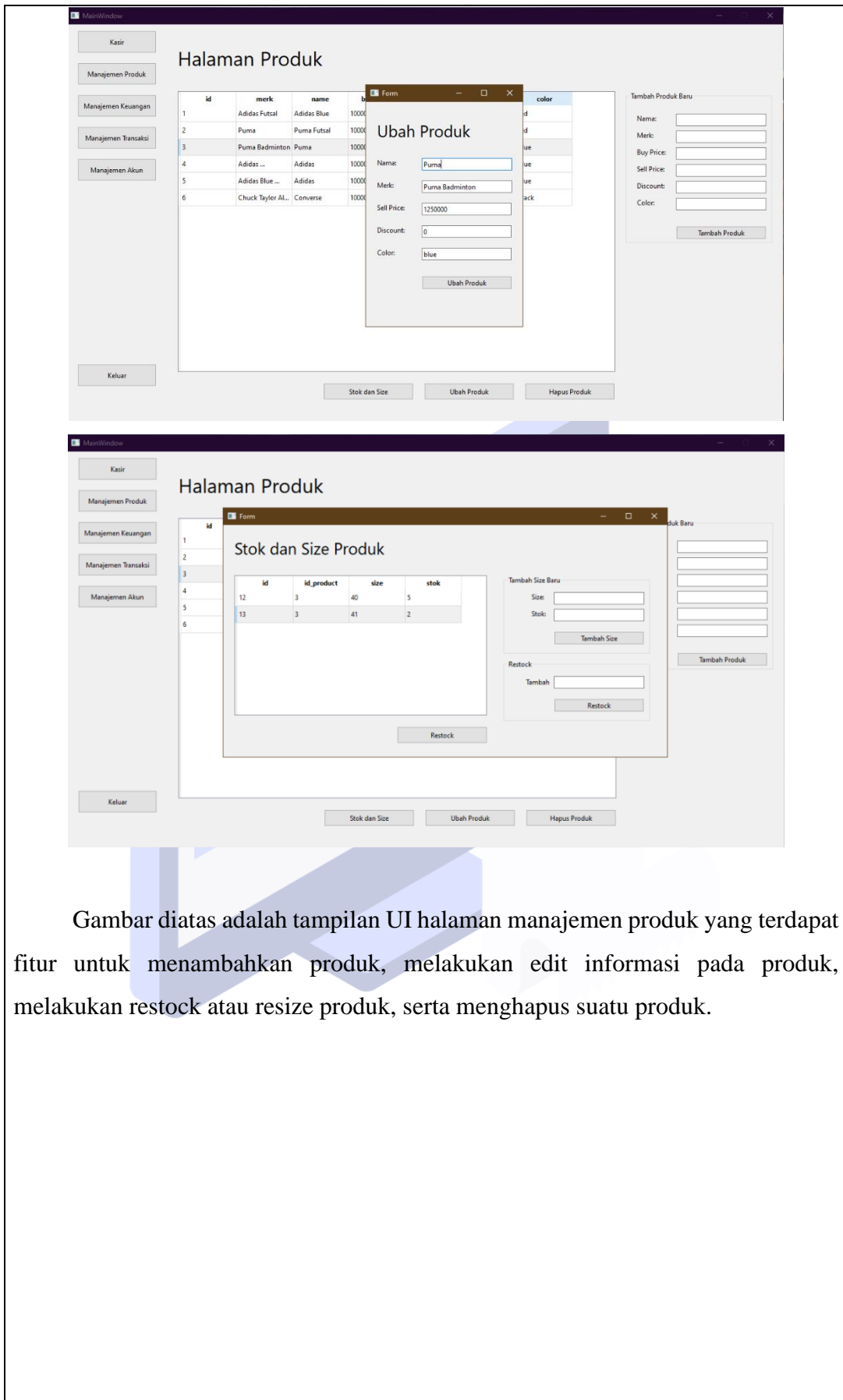


Gambar diatas adalah tampilan UI halaman kasir, pada halaman ini terdapat fitur untuk mencari barang, menghitung kembalian, dan mengetahui detail transaksi yang sedang dilakukan.

4.3 Hasil Running Menu Manajemen Produk

Subbab ini menjelaskan mengenai hasil *running* dari menu manajemen produk pada program Point of Sales.

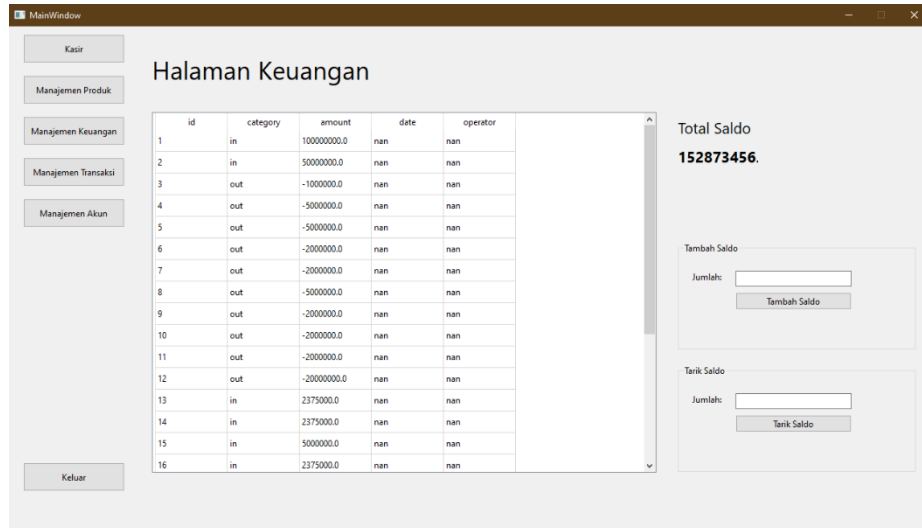




Gambar diatas adalah tampilan UI halaman manajemen produk yang terdapat fitur untuk menambahkan produk, melakukan edit informasi pada produk, melakukan restock atau resize produk, serta menghapus suatu produk.

4.4 Hasil Running Menu Manajemen Keuangan

Subbab ini menjelaskan mengenai hasil *running* dari menu manajemen keuangan pada program Point of Sales.



id	category	amount	date	operator
1	in	10000000.0	nan	nan
2	in	50000000.0	nan	nan
3	out	-1000000.0	nan	nan
4	out	-5000000.0	nan	nan
5	out	-5000000.0	nan	nan
6	out	-2000000.0	nan	nan
7	out	-2000000.0	nan	nan
8	out	-5000000.0	nan	nan
9	out	-2000000.0	nan	nan
10	out	-2000000.0	nan	nan
11	out	-2000000.0	nan	nan
12	out	-2000000.0	nan	nan
13	in	2375000.0	nan	nan
14	in	2375000.0	nan	nan
15	in	5000000.0	nan	nan
16	in	2375000.0	nan	nan

Total Saldo
152873456.

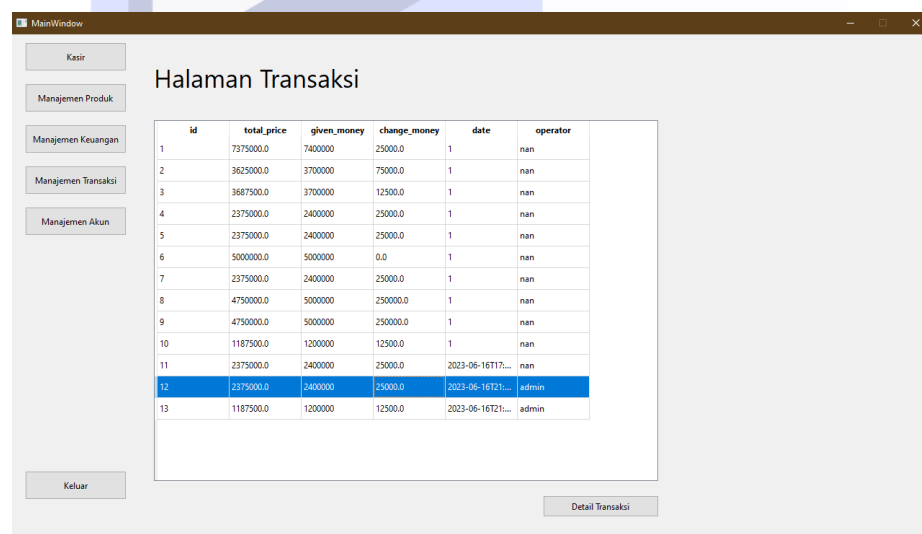
Tambah Saldo
Jumlah:
Tambah Saldo

Tarik Saldo
Jumlah:
Tarik Saldo

Gambar diatas adalah tampilan UI halaman manajemen keuangan yang mana dapat menampilkan daftar transaksi keluar dan masuk, pada halaman ini juga disediakan fitur untuk menambahkan aset/modal serta terdapat fitur untuk menarik saldo.

4.5 Hasil Running Menu Manajemen Transaksi

Subbab ini menjelaskan mengenai hasil *running* dari menu manajemen transaksi pada program Point of Sales.



id	total_price	given_money	change_money	date	operator
1	7375000.0	7400000	25000.0	1	nan
2	3625000.0	3700000	75000.0	1	nan
3	3687500.0	3700000	12500.0	1	nan
4	2375000.0	2400000	25000.0	1	nan
5	2375000.0	2400000	25000.0	1	nan
6	5000000.0	5000000	0.0	1	nan
7	2375000.0	2400000	25000.0	1	nan
8	4750000.0	5000000	250000.0	1	nan
9	4750000.0	5000000	250000.0	1	nan
10	1187500.0	1200000	12500.0	1	nan
11	2375000.0	2400000	25000.0	2023-06-16T17:...	nan
12	2375000.0	2400000	25000.0	2023-06-16T21:...	admin
13	1187500.0	1200000	12500.0	2023-06-16T21:...	admin

Detail Transaksi

Detail Transaksi

Daftar item yang dibeli

id	id_transaction	product_name	product_merk	product_price	product_discount	qty	size	total_price
14	12	Adidas Blue	Adidas Futsal	1250000	5	2	40	2375000.0

Detail Transaksi

Id: 12 Given Money: 2400000
 Total Price: 2375000.0 Change Money: 25000.0
 Date: 2023-06-16T21:08:00 Operator: admin

Gambar diatas adalah tampilan UI halaman manajemen transaksi, pada halaman ini *user* dapat mencari dan mengetahui seluruh detail transaksi yang sudah dilakukan.

4.6 Hasil Running Menu Manajemen Akun Pegawai

Subbab ini menjelaskan mengenai hasil *running* dari menu manajemen akun pegawai pada program Point of Sales.

Halaman Akun

id	role	name	username	password
1	admin	admin	admin	admin123
2	cashier	cashier 1	cashier1	cashier123
3	cashier	cashier 2	cashier2	cashier123
4	cashier	cashier 3	cashier3	cashier123

Tambah Akun Baru

Name:
 Username:
 Password:

Edit Akun

Name:
 Username:
 Password:

Gambar diatas adalah tampilan UI halaman manajemen akun pegawai, halaman ini hanya bisa diakses oleh sang pemilik usaha melalui akun khusus, disini *user* dapat mengelola informasi milik akun-akun lain seperti nama, username dan password serta tersedia fitur untuk menghapus suatu akun.