LAPORAN UJIAN AKHIR SEMESTER

PEMROGRAMAN BERORIENTASI OBJEK

DISUSUN OLEH :

Kelompok 4:

1. Natasya Salsabilla (G1A022023)
2. Alif Nurhidayat (G1A022073)
3. Saniyyah Zhafirah (G1A022081)

Asisten Dosen :

1. Randi Julian S (G1A019066)

Dosen Pengampu :

1. Arie Vatresia, S.T,.M.TI, Ph.D.
2. Mochammad Yusa, S. Kom., M. Kom.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2023

LANDASAN TEORI

Python adalah bahasa pemrograman tingkat tinggi, interpretatif, dan general-purpose yang pertama kali dirilis pada tahun 1991 oleh Guido van Rossum. Python didesain untuk mudah dibaca dan ditulis, dengan sintaks yang sederhana dan jelas, sehingga cocok untuk pemula yang baru belajar pemrograman. Python didesain untuk memudahkan pengguna dalam mengembangkan program dengan sintaks yang sederhana dan mudah dipahami. Python juga memiliki sejumlah fitur yang membuatnya populer di kalangan pengembang, seperti dukungan yang kuat untuk pengembangan ilmu data dan kecerdasan buatan, banyak library dan framework yang tersedia, dan dapat dijalankan pada berbagai platform.

Python juga memiliki filosofi "baterai sudah termasuk" yang berarti banyak fitur dan modul bawaan yang tersedia dalam bahasa itu sendiri, sehingga pengguna tidak perlu menginstal banyak dependencies tambahan untuk memulai mengembangkan aplikasi. Python dapat digunakan untuk mengembangkan berbagai jenis perangkat lunak, mulai dari aplikasi desktop, aplikasi web, hingga kecerdasan buatan. Python juga mendukung paradigma pemrograman berorientasi objek (OOP), sehingga pengguna dapat mengorganisir kode menjadi unit yang lebih besar dan lebih terstruktur. Python juga gratis dan open source, sehingga siapa saja dapat mengunduh dan menggunakannya tanpa biaya. Karena banyaknya keuntungan yang ditawarkan oleh Python, banyak perusahaan dan organisasi besar seperti Google, YouTube, Instagram, dan NASA menggunakan Python sebagai bahasa pemrograman utama mereka.

Guido menciptakan Python saat masih bekerja di Centrum Wiskunde & Informatica (CWI) di Belanda. Awalnya, Guido menciptakan Python sebagai proyek hobi untuk membuat bahasa pemrograman yang mudah dibaca dan dipelajari. Nama "Python" berasal dari candaan Guido terhadap acara televisi Inggris "Monty Python's Flying Circus". Guido menyukai acara tersebut dan memilih nama "Python" sebagai nama bahasa pemrograman barunya.

Python pertama kali dirilis ke publik pada tahun 1991 dengan versi 0.9.0. Pada awalnya, Python tidak populer dan hanya digunakan oleh sejumlah kecil pengembang perangkat lunak. Namun, seiring berjalannya waktu, Python semakin populer dan digunakan dalam berbagai aplikasi dan industri. Pada tahun 2000, versi Python 2.0 dirilis dengan beberapa fitur baru, termasuk garbage collector dan list comprehensions. Versi ini juga memperkenalkan dukungan untuk Unicode dan memperbarui model objek internal Python. Pada tahun 2008, dirilis Python 3.0. Versi ini adalah versi yang tidak kompatibel dengan Python 2.x. Python 3.0 dirancang untuk memperbaiki sejumlah masalah dalam Python 2.x termasuk dukungan Unicode yang lebih baik, penghapusan beberapa fitur yang dianggap ti-

dak kompatibel dengan Python 2.x. Python 3.0 dirancang untuk memperbaiki sejumlah masalah dalam Python 2.x, termasuk dukungan Unicode yang lebih baik, penghapusan beberapa fitur yang dianggap tidak penting, dan pengenalan beberapa fitur baru. Dan saat ini, Python menjadi salah satu bahasa pemrograman yang paling populer di dunia dan digunakan dalam berbagai aplikasi, termasuk pemrograman web, ilmu data, kecerdasan buatan, dan banyak lagi. Python juga telah menjadi bahasa pemrograman yang populer untuk pengembangan aplikasi di lingkungan open source. Python dikembangkan dan didukung oleh Python Software Foundation (PSF), sebuah organisasi nirlaba yang didedikasikan untuk pengembangan dan promosi Python. Python juga memiliki komunitas pengembang yang besar dan aktif, yang terus mengembangkan dan memperbarui bahasa ini.

Python memiliki struktur yang sederhana dan mudah dipahami oleh pemula. Struktur program Python terdiri dari beberapa elemen utama, yaitu komentar, variabel, tipe data, operator, percabangan, perulangan, dan fungsi. Komentar digunakan untuk memberikan penjelasan atau dokumentasi pada kode. Variabel digunakan untuk menyimpan nilai atau data dalam program. Tipe data berfungsi untuk menentukan jenis nilai atau data yang dapat diterima oleh suatu variabel. Operator digunakan untuk melakukan operasi pada variabel atau nilai. Percabangan digunakan untuk memeriksa kondisi tertentu dan melakukan tindakan yang berbeda sesuai dengan kondisi tersebut. Perulangan digunakan untuk melakukan tindakan yang sama berulang kali. Fungsi digunakan untuk mengorganisir kode menjadi unit yang lebih kecil dan mudah dipahami. Dengan memahami struktur dasar Python, pengembang dapat membuat program yang lebih kompleks dan berfungsi dengan baik.

Python juga dapat digunakan dengan berbagai cara, tergantung pada keperluan dan platform yang digunakan. Untuk menggunakan Python secara umum, pengguna dapat mengunduh dan menginstal Python interpreter dari situs resmi Python. Setelah terinstal, pengguna dapat membuat dan menjalankan program Python menggunakan text editor atau IDE (Integrated Development Environment) yang tersedia. Langkah selanjutnya adalah menuliskan kode program Python menggunakan sintaks dan struktur yang telah dipelajari, kemudian menyimpannya dengan ekstensi ".py". Program Python dapat dijalankan dengan cara membuka terminal atau command prompt, menavigasi ke direktori program, dan mengetik perintah "python nama_program.py". Pengguna juga dapat menggunakan Python melalui aplikasi atau framework yang dibangun dengan menggunakan Python, seperti Jupyter Notebook, Flask, atau Django. Dalam penggunaan Python, sangat penting untuk memahami sintaks dan struktur dasar Python, serta menguasai pustaka atau library yang relevan dengan keperluan atau proyek yang sedang dikerjakan, dengan menguasai bahasa python, akan memudahkan pengguna untuk mengaplikasikan kode.

SOAL DAN PEMBAHASAN

Fasilitas pemesanan makanan menggunakan python : aplikasi ini membantu seseorang memesan makanan dari rumah tanpa membuang waktu di restoran.

Jawab:

Printscreen source code:

```
from PyQt6.QtWidgets import QScrollArea, QDialog, QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QFileDialog
from PyQt6.QtGui import QPixmap, QGuiApplication, QIcon
from PyQt6.QtCore import Qt
from pathlib import Path
from datetime import datetime
import os, shutil

class printText(QDialog):
    def __init__(self, item, order, count, dict = "nonSub"):
        super().__init__()
        self.item = item
        self.order = order
        self.setWindowTitle('Checkout')
        self.setGeometry(100,100,300,200)
        self.dict = dict
        self.itemsCounterPrinted = count
        self.itemsCounterParser = "Mi[{}XCVHELL{}]".format(self.itemsCounterPrinted, self.dict)
        self.PrintText()
        self.showD()

    def leftStringFormater(self, text, amnt):
        length = len(text)
        return text + " "*(amnt-length)

    def midStringFormater(self, text, amnt):
        length = len(text)
        length = abs((amnt-length)//2)
        return " "*length+text+" "*length

    def rightStringFormater(self, text, amnt):
        length = len(text)
        return " "*(amnt-length) + text

    def PrintText(self):
        length = 36
        self.text = ""
```

```
    def PrintText(self):
        length = 36
        self.text = ""
        self.text += "STARLA x STORE\n"
        self.text += "Ruko Starla Jati Asih\n"
        self.text += "(047)\n"
        self.text += ("-"*length) + "\n"
        Notes = "Kode Nota:     {} ".format(self.itemsCounterParser)
        Notes.center(14)
        self.text += (Notes)

        ###
        dataString = f"\n931242       {str(datetime.now().strftime('%Y-%m-%d %H:%M:%S'))}\n"
        dataString.center(length)
        dataString.replace(" ", "-")
        ###
        self.text += dataString
        self.text += ("-"*length) + "\n"
        self.text.center(length)

        total = 0
        self.text += "\n"
        self.text += "\n"
        for orderKey in self.order:
            text0 = "{}".format(self.item[orderKey].name)
            text0 = self.leftStringFormater(text0, 15)
            text1 = "{}".format("x"+str(self.order[orderKey]))
            text1 = self.rightStringFormater(text1, 8)
            text2 = "{}".format("Rp. "+str(self.item[orderKey].price))
            text2 = self.rightStringFormater(text2, 13)
            self.text += (text0+text1+text2+"\n")
            total += (self.order[orderKey] * self.item[orderKey].price)

        self.text += (("\n"+"-"*length) + "\n")
        self.text += ("TOTAL : Rp. {}\n".format(str(total)))
        self.text += ("-"*length) + "\n"
```

```
BERIKAN NOTA INI KEPADA KASIR
UNTUK MELAKUKAN PEMESANAN

THANK YOU
PLEASE BUY AGAIN
'''
        thanks.center(length)
        self.text += thanks

    def showD(self):
        self.itemsCounterPrinted +=1
        self.label = QLabel(self.text)
        self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.label.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

        self.button = QPushButton("Cetak")
        self.button.clicked.connect(self.callF)
        VBox = QVBoxLayout()
        VBox.addWidget(self.label)
        VBox.addWidget(self.button)
        self.setLayout(VBox)

    def callF(self):
        filename, _ = QFileDialog.getSaveFileName(self, 'Save File')
        if filename:
            with open(filename, 'w') as f:
                f.write(self.text)
                ###print("Saved to {}".format(filename))
```

```
class ItemNodes():
    def __init__(self, name, amnt = 0, price = 0, img = "", information = ""):
        self.name = name
        self.amnt = amnt
        self.price = price
        self.img = img
        self.information = information

    ### def check(self):
    ###print("ItemNode Check: {} {} {} {}".format(self.amnt, self.price, self.img, self.information))


class ItemsCore():
    def __init__(self):
        self.itemNodesHolder = {}
        self.orderedItems = {}
        self.filpath = (str(__file__).replace(Path(__file__).name, "")+"src\\").replace("\\", "/")+"items.conf"
        ## self.itemNodesHolder["dav"] = ItemNodes(12, 12000, "C:/Users/KillerKing/Downloads/Screenshot-2022-09-1
        ## self.itemNodesHolder["pena"] = ItemNodes(15, 600, )

    def checkAval(self, key, amnt):
        return amnt < self.itemNodesHolder[key].amnt

    def get(self):
        return dict(self.itemNodesHolder)

    def getOrder(self):
        return dict(self.orderedItems)

    def incItem(self, key):
        key = key.lower()
        ###print(self.orderedItems)
        if key not in self.orderedItems:
            self.orderedItems[key] = 1
            ###print(self.orderedItems[key])
            return True
        elif self.orderedItems[key] < self.itemNodesHolder[key].amnt:
```

deringProgram.py > ...

```
    def decItem(self, key):
        key = key.lower()
        if key not in self.orderedItems:
            return False
        self.orderedItems[key] -= 1
        ###print(self.orderedItems[key])
        if self.orderedItems[key] < 1:
            del self.orderedItems[key]
        return True

    def saveCurrentItems(self):
        strs = ""
        for key in self.itemNodesHolder:
            strs += (key + " [!=-=Holder=-=!] " + self.itemNodesHolder[key].name +" [!=-=Holder=-=!] "+str(self.itemNodesHolder[key].amnt)+" [!=-=Holder=-=!] "+str(self.i
        ###print(strs)
        try:
            with open(self.filpath, 'w') as f:
                f.write(strs)
                ###print('Saved to {}'.format(filpath))
                return True
        except(FileNotFoundError):
            ###print("ERROR!, {}".format(filpath))
            return

    def loadItems(self):
        ##print(filpath)
        with open(self.filpath, 'r') as f:
            data = f.read()
            ##print(data)
            data = data.split("\n")
            for rdata in data:
                ##print(rdata)
                rdata = rdata.split(" [!=-=Holder=-=!] ")
                ##print(rdata)
                try:
                    self.itemNodesHolder[rdata[0]] = ItemNodes(rdata[1], int(rdata[2]), int(rdata[3]), rdata[4], rdata[5])
                except(IndexError):
```

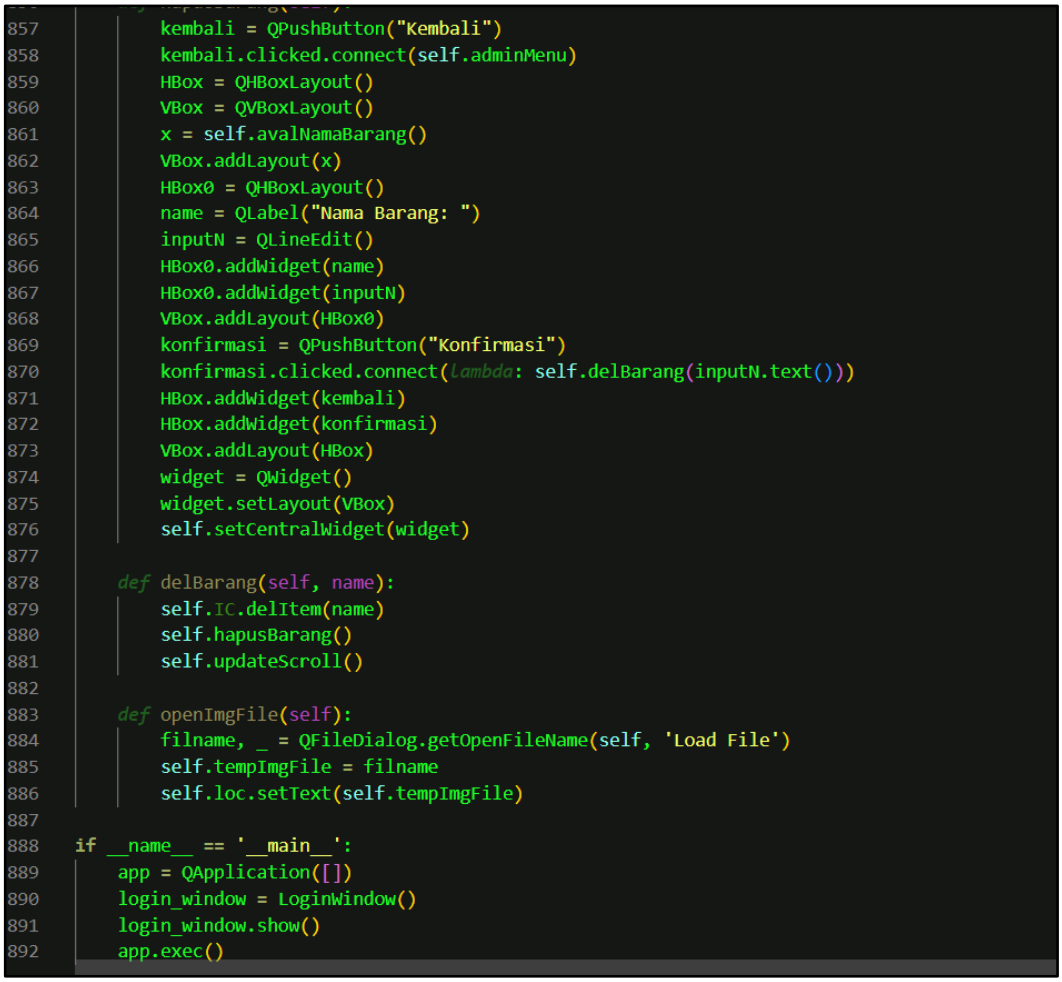
```
    def addItem(self, key, name, amnt, price, img, information):

        ###print("addI, : | {} | {} | {} | {} | {} |".format(key, amnt, price, img, information))
        ###print(key, amnt, price, img, information)
        pureimg = img.strip().split("/")
        pureimg = pureimg[-1]
        if not os.path.isfile(self.filpath+pureimg):
            shutil.copy(img, self.filpath+pureimg)
            ##print("Image Moved Successfully")
        self.itemNodesHolder[key.lower()] = ItemNodes(name, amnt, price, pureimg, information)
        ##print (self.itemNodesHolder[key])
        self.saveCurrentItems()
        return True

    def delItem(self, key):
        try:
            if os.path.isfile(self.filpath+self.itemNodesHolder[key].img):
                os.remove(self.filpath+self.itemNodesHolder[key].img)
            del self.itemNodesHolder[key]
            self.saveCurrentItems()
            return True
        except(KeyError):
            return False

    def resetOrder(self):
        self.orderedItems = {}

class ErrorPopUp(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Error!')
        self.setGeometry(100,100,300,200)

    def error(self, error):
        VBox = QVBoxLayout()
        label0 = QLabel(error)
```

```
    def error(self, error):
        VBox = QVBoxLayout()
        label0 = QLabel(error)
        label0.setAlignment(Qt.AlignmentFlag.AlignCenter)
        VBox.addWidget(label0)
        self.setLayout(VBox)

class itemBoxDetail(QWidget):
    def __init__(self, name, itemAmnt, price):
        super().__init__()
        self.type = ""
        self.name = QLabel(str(name)+", ")
        self.amntBox = QLabel("   x"+str(itemAmnt)+",")
        self.priceBox = QLabel(str(price))
        HBox = QHBoxLayout()
        HBox.addWidget(self.name)
        HBox.addWidget(self.amntBox)
        HBox.addWidget(self.priceBox)
        self.setLayout(HBox)

class itemDetail(QWidget):
    def __init__(self):
        super().__init__()
        self.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

    def updateItem(self, name, itemAmnt, price):
        self.type = ""
        self.nameIT = QLabel(str(name)+", ")
        self.amntBox = QLabel(str(itemAmnt+","))
        self.priceBox = QLabel(str(price))
        HBox = QHBoxLayout()
```

```
264
265     def updateItem(self, information):
266         super().__init__()
267         scrolls = Scrolly()
268         self.label = QLabel(information)
269         self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)
270         self.label.setStyleSheet("""
271     QLabel {
272         border: 1px solid black;
273         border-radius: 0px;
274     }
275 """)
276         scrolls.add_widget
277         widget = QWidget()
278         HBox = QHBoxLayout(self.label)
279         widget.setLayout(HBox)
280         scrolls.add_widget(widget)
281         HBox = QHBoxLayout()
282         HBox.addWidget(scrolls)
283         self.layout(HBox)
284
285 class Scrolly(QScrollArea):
286     def __init__(self):
287         super().__init__()
288         self.i = 0
289         self.widget = QWidget()
290         self.widget.setLayout(QVBoxLayout())
291         self.setVerticalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOn)
292         self.setHorizontalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOff)
293         self.setWidgetResizable(True)
294         self.setWidget(self.widget)
295
296     def add_widget(self, widget):
297         self.widget.layout().addWidget(widget)
298         self.update()
299         self.i += 1
```

```python
class ImagedItemNode(QWidget):
    def __init__(self, key, amnt, price, img, info, IC, types = "nonSub"):
        super().__init__()
        self.amnt = amnt
        self.img = (str(__file__).replace(Path(__file__).name, "")+"src/") + img
        self.subWidget = QWidget()
        self.key = key
        self.price = price
        self.info = info
        self.IC = IC
        self.types = types

        # Key #
        self.label = QLabel(key)
        self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.label.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

        # Counter #
        self.counter = 0
        self.counterBox = QLabel("Dipesan: {}".format(str(self.counter)))
        self.counterBox.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.counterBox.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

        # Price #pixmap
        self.priceBox = QLabel("Rp. " + str(self.price))
```

```python
class LoginWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        ### ESSENTIALS

        self.user = {"user":"user"}  #Default User Username & Password: user:user
        self.admin = {"admin":"admin"} #Default Admin Username & Password: admin:admin
        self.error = ""
        self.filePath = ""
        self.tempImgFile = ""
        self.loginType = "user"
        self.IC = ItemsCore()
        self.scrolls = Scrolly()
        self.curr_dir = (str(__file__).replace(Path(__file__).name, "")).replace("\\", "/")
        self.cfg_dir = self.curr_dir + "/Essentials/cfg.main"
        self.checkoutCount = 0

        ### ESSENTIALS

        self.iconLink = self.curr_dir.replace("/", "\\")+"src\Icon.png"
        self.setWindowIcon(QIcon(self.iconLink))
        self.setGeometry(500,250,500,200)
        self.MainMenu()

    def scrollWidget(self):
        self.scrolls = Scrolly()
        data = self.IC.get()
        ###print(data)
        for key in data:
            amnt = int(data[key].amnt)
            price = int(data[key].price)
            img = data[key].img
            info = data[key].information
            ###print(amnt, price, img, info)
            self.scrolls.add_widget(ImagedItemNode(data[key].name, amnt, price, img, info, self.IC, "nonSub"))
        return self.scrolls
```



```
857         kembali = QPushButton("Kembali")
858         kembali.clicked.connect(self.adminMenu)
859         HBox = QHBoxLayout()
860         VBox = QVBoxLayout()
861         x = self.avalNamaBarang()
862         VBox.addLayout(x)
863         HBox0 = QHBoxLayout()
864         name = QLabel("Nama Barang: ")
865         inputN = QLineEdit()
866         HBox0.addWidget(name)
867         HBox0.addWidget(inputN)
868         VBox.addLayout(HBox0)
869         konfirmasi = QPushButton("Konfirmasi")
870         konfirmasi.clicked.connect(lambda: self.delBarang(inputN.text()))
871         HBox.addWidget(kembali)
872         HBox.addWidget(konfirmasi)
873         VBox.addLayout(HBox)
874         widget = QWidget()
875         widget.setLayout(VBox)
876         self.setCentralWidget(widget)
877 
878     def delBarang(self, name):
879         self.IC.delItem(name)
880         self.hapusBarang()
881         self.updateScroll()
882 
883     def openImgFile(self):
884         filname, _ = QFileDialog.getOpenFileName(self, 'Load File')
885         self.tempImgFile = filname
886         self.loc.setText(self.tempImgFile)
887 
888 if __name__ == '__main__':
889     app = QApplication([])
890     login_window = LoginWindow()
891     login_window.show()
892     app.exec()
```

Gambar 1.1 Print Screen Source code

Source code:

```
from PyQt6.QtWidgets import QScrollArea, QDialog, QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QFileDialog
from PyQt6.QtGui import QPixmap, QGuiApplication, QIcon
from PyQt6.QtCore import Qt
from pathlib import Path
from datetime import datetime
import os, shutil


class printText(QDialog):
    def __init__(self, item, order, count, dict = "nonSub"):
        super().__init__()
        self.item = item
        self.order = order
        self.setWindowTitle('Checkout')
```

```
        self.setGeometry(100,100,300,200)
        self.dict = dict
        self.itemsCounterPrinted = count
        self.itemsCounterParser      =      "Mi[{}XCVHELL{}]".format(self.itemsCounterPrinted,
self.dict)
        self.PrintText()
        self.showD()

    def leftStringFormater(self, text, amnt):
        length = len(text)
        return text + " "*(amnt-length)

    def midStringFormater(self, text, amnt):
        length = len(text)
        length = abs((amnt-length)//2)
        return " "*length+text+" "*length

    def rightStringFormater(self, text, amnt):
        length = len(text)
        return " "*(amnt-length) + text

    def PrintText(self):
        length = 36
        self.text = ""
        self.text += "STARLA x STORE\n"
        self.text += "Ruko Starla Jati Asih\n"
        self.text += "(047)\n"
        self.text += ("-"*length) + "\n"
        Notes = "Kode Nota:     {} ".format(self.itemsCounterParser)
        Notes.center(14)
        self.text += (Notes)
        ###
        dataString    =    f"\n931242                    {str(datetime.now().strftime('%Y-%m-%d
%H:%M:%S'))}\n"
        dataString.center(length)
```

```
        dataString.replace(" ", "-")
        ###
        self.text += dataString
        self.text += ("-"*length) + "\n"
        self.text.center(length)

        total = 0
        self.text += "\n"
        self.text += "\n"
        for orderKey in self.order:
            text0 = "{}".format(self.item[orderKey].name)
            text0 = self.leftStringFormater(text0, 15)
            text1 = "{}".format("x"+str(self.order[orderKey]))
            text1 = self.rightStringFormater(text1, 8)
            text2 = "{}".format("Rp. "+str(self.item[orderKey].price))
            text2 = self.rightStringFormater(text2, 13)
            self.text += (text0+text1+text2+"\n")
            total += (self.order[orderKey] * self.item[orderKey].price)

        self.text += (("\n"+"-"*length) + "\n")
        self.text += ("TOTAL : Rp. {}\n".format(str(total)))
        self.text += ("-"*length) + "\n"

        thanks = '''
BERIKAN NOTA INI KEPADA KASIR
UNTUK MELAKUKAN PEMESANAN
THANK YOU
PLEASE BUY AGAIN
'''
        thanks.center(length)
        self.text += thanks

    def showD(self):
        self.itemsCounterPrinted +=1
        self.label = QLabel(self.text)
```

```
        self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.label.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }""")
        self.button = QPushButton("Cetak")
        self.button.clicked.connect(self.callF)
        VBox = QVBoxLayout()
        VBox.addWidget(self.label)
        VBox.addWidget(self.button)
        self.setLayout(VBox)

    def callF(self):
        filename, _ = QFileDialog.getSaveFileName(self, 'Save File')
        if filename:
            with open(filename, 'w') as f:
                f.write(self.text)
                ###print("Saved to {}".format(filename))

class ItemNodes():
    def __init__(self, name, amnt = 0, price = 0, img = "", information = ""):
        self.name = name
        self.amnt = amnt
        self.price = price
        self.img = img
        self.information = information

    ### def check(self):
    ###print("ItemNode Check: {} {} {} {}".format(self.amnt, self.price, self.img,
self.information))

class ItemsCore():
    def __init__(self):
        self.itemNodesHolder = {}
```

```
        self.orderedItems = {}
        self.filpath = (str(__file__).replace(Path(__file__).name, "")+"src\\").replace("\\", "/")+"items.conf"
        ## self.itemNodesHolder["dav"] = ItemNodes(12, 12000, "C:/Users/KillerKing/Downloads/Screenshot-2022-09-13-183142.webp", "HI THERE!")
        ## self.itemNodesHolder["pena"] = ItemNodes(15, 600, )

    def checkAval(self, key, amnt):
        return amnt < self.itemNodesHolder[key].amnt

    def get(self):
        return dict(self.itemNodesHolder)

    def getOrder(self):
        return dict(self.orderedItems)

    def incItem(self, key):
        key = key.lower()
        ###print(self.orderedItems)
        if key not in self.orderedItems:
            self.orderedItems[key] = 1
            ###print(self.orderedItems[key])
            return True
        elif self.orderedItems[key] < self.itemNodesHolder[key].amnt:
            self.orderedItems[key] += 1
            ###print(self.orderedItems[key])
            return True
        else:
            return False
    def decItem(self, key):
        key = key.lower()
        if key not in self.orderedItems:
            return False
        self.orderedItems[key] -= 1
        ###print(self.orderedItems[key])
```

```
        if self.orderedItems[key] < 1:
            del self.orderedItems[key]
        return True

    def saveCurrentItems(self):
        strs = ""
        for key in self.itemNodesHolder:
            strs += (key + " [!=-=Holder=-=!] " + self.itemNodesHolder[key].name +" [!=-
=Holder=-=!]              "+str(self.itemNodesHolder[key].amnt)+"              [!=-=Holder=-=!]
"+str(self.itemNodesHolder[key].price)+"                                    [!=-=Holder=-=!]
"+self.itemNodesHolder[key].img+"                                           [!=-=Holder=-=!]
"+self.itemNodesHolder[key].information+"\n")
        ###print(strs)
        try:
            with open(self.filpath, 'w') as f:
                f.write(strs)
                ###print('Saved to {}'.format(filpath))
                return True
        except(FileNotFoundError):
            ###print("ERROR!, {}".format(filpath))
            return

    def loadItems(self):
        ##print(filpath)
        with open(self.filpath, 'r') as f:
            data = f.read()
            ##print(data)
            data = data.split("\n")
            for rdata in data:
                ##print(rdata)
                rdata = rdata.split(" [!=-=Holder=-=!] ")
                ##print(rdata)
                try:
                    self.itemNodesHolder[rdata[0]] = ItemNodes(rdata[1], int(rdata[2]), int(rdata[3]),
rdata[4], rdata[5])
```

```
            except(IndexError):
                pass
            ###print(rdata)
            ###print('Loaded from {}'.format(filpath))

    def addItem(self, key, name, amnt, price, img, information):

        ###print("addI, : | {} | {} | {} | {} | {} |".format(key, amnt, price, img, information))
        ###print(key, amnt, price, img, information)
        pureimg = img.strip().split("/")
        pureimg = pureimg[-1]
        if not os.path.isfile(self.filpath+pureimg):
            shutil.copy(img, self.filpath+pureimg)
            ##print("Image Moved Successfully")
        self.itemNodesHolder[key.lower()] = ItemNodes(name, amnt, price, pureimg,
information)
        ##print (self.itemNodesHolder[key])
        self.saveCurrentItems()
        return True

    def delItem(self, key):
        try:
            if os.path.isfile(self.filpath+self.itemNodesHolder[key].img):
                os.remove(self.filpath+self.itemNodesHolder[key].img)
            del self.itemNodesHolder[key]
            self.saveCurrentItems()
            return True
        except(KeyError):
             return False

    def resetOrder(self):
        self.orderedItems = {}

class ErrorPopUp(QDialog):
    def __init__(self):
```

```
        super().__init__()
        self.setWindowTitle('Error!')
        self.setGeometry(100,100,300,200)

    def error(self, error):
        VBox = QVBoxLayout()
        label0 = QLabel(error)
        label0.setAlignment(Qt.AlignmentFlag.AlignCenter)
        VBox.addWidget(label0)
        self.setLayout(VBox)

class itemBoxDetail(QWidget):
    def __init__(self, name, itemAmnt, price):
        super().__init__()
        self.type = ""
        self.name = QLabel(str(name)+", ")
        self.amntBox = QLabel("   x"+str(itemAmnt)+",")
        self.priceBox = QLabel(str(price))
        HBox = QHBoxLayout()
        HBox.addWidget(self.name)
        HBox.addWidget(self.amntBox)
        HBox.addWidget(self.priceBox)
        self.setLayout(HBox)

class itemDetail(QWidget):
    def __init__(self):
        super().__init__()
        self.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

    def updateItem(self, name, itemAmnt, price):
```

```
        self.type = ""
        self.nameIT = QLabel(str(name)+", ")
        self.amntBox = QLabel(str(itemAmnt+","))
        self.priceBox = QLabel(str(price))
        HBox = QHBoxLayout()
        HBox.addWidget(self.nameIT)
        HBox.addWidget(self.amntBox)
        HBox.addWidget(self.priceBox)
        HBox.layout()

    def updateItem(self, information):
        super().__init__()
        scrolls = Scrolly()
        self.label = QLabel(information)
        self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.label.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")
        scrolls.add_widget
        widget = QWidget()
        HBox = QHBoxLayout(self.label)
        widget.setLayout(HBox)
        scrolls.add_widget(widget)
        HBox = QHBoxLayout()
        HBox.addWidget(scrolls)
        self.layout(HBox)

class Scrolly(QScrollArea):
    def __init__(self):
        super().__init__()
        self.i = 0
        self.widget = QWidget()
```

```
        self.widget.setLayout(QVBoxLayout())
        self.setVerticalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOn)
        self.setHorizontalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOff)
        self.setWidgetResizable(True)
        self.setWidget(self.widget)

    def add_widget(self, widget):
        self.widget.layout().addWidget(widget)
        self.update()
        self.i += 1

class ImagedItemNode(QWidget):
    def __init__(self, key, amnt, price, img, info, IC, types = "nonSub"):
        super().__init__()
        self.amnt = amnt
        self.img = (str(__file__).replace(Path(__file__).name, "")+"src/") + img
        self.subWidget = QWidget()
        self.key = key
        self.price = price
        self.info = info
        self.IC = IC
        self.types = types

        # Key #
        self.label = QLabel(key)
        self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.label.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

        # Counter #
        self.counter = 0
```

```
        self.counterBox = QLabel("Dipesan: {}".format(str(self.counter)))
        self.counterBox.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.counterBox.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

        # Price #pixmap
        self.priceBox = QLabel("Rp. " + str(self.price))
        self.priceBox.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.priceBox.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

        # Icon #
        self.imgLabel = QLabel()
        self.pixmap = QPixmap('{}'.format(self.img))
        self.scaled_pixmap = self.pixmap.scaled(150, 150)
        self.imgLabel.setPixmap(self.scaled_pixmap)

    #Detail Box
        self.DetailBox = QPushButton("Detail")
        self.DetailBox.clicked.connect(self.updateDetailed)

    #Info Box
        self.infoBox = QWidget()
        self.scrolly = Scrolly()
        self.infos = None
        if self.types == "nonSub":
            self.infos = QLabel(str(info))
```

```
        #Scroll Box
            self.infoV = QVBoxLayout()
            self.infoV.addWidget(self.infos)
            infoWidget = QWidget()
            infoWidget.setLayout(self.infoV)
            self.scrolly.add_widget(infoWidget)
            self.infoV2 = QVBoxLayout()
            self.infoV2.addWidget(self.scrolly)
            self.infoBox.setLayout(self.infoV2)
            self.infoBox.setHidden(True)

        #Vertical for adding utility
        self.counterBody = QVBoxLayout()
        self.amntBox = QLabel("Tersedia: {}".format(str(self.amnt)))
        self.amntBox.setStyleSheet("""
    QLabel {
        border: 1px solid black;
        border-radius: 0px;
    }
""")

        self.buttonplus = QPushButton("+")
        self.buttonplus.clicked.connect(self.incBI)
        self.buttonnegav = QPushButton("-")
        self.buttonnegav.clicked.connect(self.decBI)
        self.buttonnegav.setEnabled(False)
        self.counterBody.addWidget(self.amntBox)
        self.counterBody.addWidget(self.counterBox)
        self.counterBody.addWidget(self.buttonplus)
        self.counterBody.addWidget(self.buttonnegav)

        #Horizontal for Img + adding utility
        self.bodyBox = QVBoxLayout()
        self.bodyBox.addWidget(self.imgLabel)
```

```
        #Body Column
        self.HbodyBox = QHBoxLayout()
        self.HbodyBox.addLayout(self.bodyBox)
        self.HbodyBox.addLayout(self.counterBody)

        # Name Column
        self.HBox1 = QHBoxLayout()
        self.HBox1.addWidget(self.DetailBox)

        self.VBox1 = QVBoxLayout()
        self.VBox1.addWidget(self.label)
        self.VBox1.addLayout(self.HbodyBox)
        self.VBox1.addWidget(self.priceBox)
        self.VBox1.addWidget(self.infoBox)
        self.VBox1.addLayout(self.HBox1)
        self.setLayout(self.VBox1)

    def updateDetailed(self):
        ###print(self.info)
        if self.infoBox.isHidden():
            self.infoBox.setHidden(False)
            self.infoBox.update()
        else:
            self.infoBox.setHidden(True)
            self.infoBox.update()
        self.update()

    def incDecUpdate(self):
        self.buttonplus.setEnabled(self.checkAval())
        self.buttonnegav.setEnabled(self.counter > 0)
        self.update()

    def checkAval(self):
        return self.counter < self.amnt
```

```
    def incBI(self):
        self.IC.incItem(self.key)
        self.counter += 1
        self.counterBox.setText("Dipesan: {}".format(str(self.counter)))
        self.incDecUpdate()

    def decBI(self):
        self.IC.decItem(self.key)
        self.counter -= 1
        self.counterBox.setText("Dipesan: {}".format(str(self.counter)))
        self.incDecUpdate()

class LoginWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        ### ESSENTIALS

        self.user = {"user":"user"}  #Default User Username & Password: user:user
        self.admin = {"admin":"admin"} #Default Admin Username & Password: admin:admin
        self.error = ""
        self.filePath = ""
        self.tempImgFile = ""
        self.loginType = "user"
        self.IC = ItemsCore()
        self.scrolls = Scrolly()
        self.curr_dir = (str(__file__).replace(Path(__file__).name, "")).replace("\\", "/")
        self.cfg_dir = self.curr_dir + "/Essentials/cfg.main"
        self.checkoutCount = 0

        ### ESSENTIALS

        self.iconLink = self.curr_dir.replace("/", "\\")+"src\Icon.png"
        self.setWindowIcon(QIcon(self.iconLink))
        self.setGeometry(500,250,500,200)
```

```
        self.MainMenu()

    def scrollWidget(self):
        self.scrolls = Scrolly()
        data = self.IC.get()
        ###print(data)
        for key in data:
            amnt = int(data[key].amnt)
            price = int(data[key].price)
            img = data[key].img
            info = data[key].information
            ###print(amnt, price, img, info)
            self.scrolls.add_widget(ImagedItemNode(data[key].name, amnt, price, img, info,
self.IC, "nonSub"))
        return self.scrolls

    def clearScrollWidget(self):
        self.scrollWidget().update()
        self.scrolls.update()

    def incItem(self,key):
        self.IC.incItem(key)

    def decItem(self,key):
        self.IC.decItem(key)

    def getUser(self):
        return self.user

    def getAdmin(self):
        return self.admin

    def addUser(self, type, user, pswd):
        if type == "user":
            self.user[user] = pswd
```

```
            return True
        elif type == "admin":
            self.admin[user] = pswd
            return True
        else:
            return False

    def delUser(self, user):
        try:
            del self.user[user]
            return True
        except(KeyError):
             return False

    def update_password(self):
        if self.cfg_dir:
            try:
                with open(self.cfg_dir, 'r') as f:
                    self.admin["admin"] = f.read().strip()
                    ###print("Password Admin: " + self.admin["admin"])
            except(FileNotFoundError):
                self.default()

    def change_password(self, pswd):
        with open(self.cfg_dir, 'w') as f:
                f.write(pswd)

    def default(self):
        with open(self.cfg_dir, 'w') as f:
            f.write("admin")

    def matchPswd(self, type, user, pswd):
        ###print(type, user, pswd)
        if type == "user" and pswd == self.user[user]:
            return "user"
```

```
        elif type == "admin" and pswd == self.admin[user]:
            return "admin"
        else:
            return False

    def findUser(self, user, pswd):
        ###print(user,pswd)
        if user in self.user:
            return self.matchPswd("user", user, pswd)
        elif user in self.admin:
            return self.matchPswd("admin", user, pswd)
        else:
            return False

    def errorPop(self):
        error = ErrorPopUp()
        error.error(self.error)
        error.exec()

    def checkout(self):
        item = self.IC.get()
        order = self.IC.getOrder()
        order = printText(item, order, self.checkoutCount)
        order.exec()
        self.checkoutCount += 1

    def MainMenu(self):
        self.setWindowTitle('Main Menu')
        self.IC.resetOrder()
        VBox = QVBoxLayout()
        self.setStyleSheet("QMainWindow {background-color: Lavender;}")
        self.applabel = QLabel("\nSTARLA x STORE\n")
        self.applabel.setStyleSheet("""
        QLabel {
            color: lightcoral;
```

```
        font-size: 34px;
        font: bold italic large "Times New Roman";
    }
    """)

    self.applabel.setAlignment(Qt.AlignmentFlag.AlignCenter)
    VBox.addWidget(self.applabel)

    self.labelIcon = QLabel()
    ###print(self.iconLink)
    self.labelpixmap = QPixmap('{}'.format(self.iconLink))
    self.scaled_labelpixmap = self.labelpixmap.scaled(250, 200)
    self.labelIcon.setPixmap(self.scaled_labelpixmap)
    self.labelIcon.setAlignment(Qt.AlignmentFlag.AlignCenter)
    VBox.addWidget(self.labelIcon)
    fill = QLabel()
    masuk = QPushButton("Mulai")
    masuk.clicked.connect(lambda: self.login("user", "user"))
    admin = QPushButton("Admin")
    admin.clicked.connect(self.AdminLogin)
    VBox.addWidget(fill)
    VBox.addWidget(masuk)
    VBox.addWidget(admin)
    widg = QWidget()
    widg.setLayout(VBox)
    self.setCentralWidget(widg)

def AdminLogin(self):
    self.setWindowTitle('Login Window')
    self.update_password()
    VBox = QVBoxLayout()
    username_label = QLabel ('username:', self)
    username_label.setStyleSheet("""
    QLabel {
        color: bold black;
```

```
        }""")
        username_input = QLineEdit(self)
        HBox0 = QHBoxLayout()
        HBox0.addWidget(username_label)
        HBox0.addWidget(username_input)
        VBox.addLayout(HBox0)

        #Create Password and its input boxes
        password_label = QLabel('password:', self)
        password_label.setStyleSheet("""
        QLabel {
            color: bold black;
            }""")
        password_input = QLineEdit(self)
        password_input.setEchoMode(QLineEdit.EchoMode.Password)
        HBox1 = QHBoxLayout()
        HBox1.addWidget(password_label)
        HBox1.addWidget(password_input)
        VBox.addLayout(HBox1)

        kembali = QPushButton("Kembali")
        kembali.clicked.connect(self.MainMenu)
        login_button = QPushButton('Login', self)
        login_button.clicked.connect(lambda:                    self.login(username_input.text(),
password_input.text()))
        HBox3 = QHBoxLayout()
        HBox3.addWidget(kembali)
        HBox3.addWidget(login_button)
        VBox.addLayout(HBox3)
        tempWidget = QWidget(self)
        tempWidget.setLayout(VBox)
        self.setCentralWidget(tempWidget)

    def login(self, user, pswd):
        ###print(user, pswd)
```

```
        stats = self.findUser(user, pswd)
        ###print(stats)
        if stats and stats == "user":
            self.UserMenu()
        elif stats and stats == "admin":
            self.adminMenu()
        else:
            error = ErrorPopUp()
            error.error("User atau Password Salah!")
            error.exec()

    def UpdateDisplay(self, Update):
            self.centralWidget(Update)

    def UserMenu(self):
        self.setWindowTitle('User Menu')
        self.loadLogic()
        self.scrollWidget().update()
        self.VBox1 = QVBoxLayout()
        self.HBox1 = QHBoxLayout()

        out = QPushButton("Log Out")
        out.clicked.connect(self.MainMenu)
        self.VBox1.addWidget(self.scrollWidget())
        refreshB = QPushButton("Refresh")
        refreshB.clicked.connect(self.updateScroll)
        self.label = QPushButton("Finalkan Pembelian")
        self.HBox1.addWidget(out)
        self.HBox1.addWidget(self.label)
        self.HBox1.addWidget(refreshB)
        self.label.clicked.connect(self.checkout)
        self.VBox1.addLayout(self.HBox1)
        widget = QWidget()
        widget.setLayout(self.VBox1)
        self.setCentralWidget(widget)
```

```
def saveLogic(self):
    ##self.filepath, _ = QFileDialog.getSaveFileName(self, 'Save File')
    ###print(self.filepath)
    ##filtxt = str(self.filepath)
    self.IC.saveCurrentItems()
    QGuiApplication.processEvents()

def loadLogic(self):
    ##self.filepath, _ = QFileDialog.getOpenFileName(self, 'Load File')
    ###print(self.filepath)
    ##filtxt = str(self.filepath)
    self.IC.loadItems()
    ##data = self.IC.get()
    ###print("This is Load Logic: {}".format(data))
    self.updateScroll()
    self.update()
    QGuiApplication.processEvents()

def updateScroll(self):
    self.clearScrollWidget()
    self.update()

def adminMenu(self):
    self.setWindowTitle('Admin Menu')
    self.loadLogic()
    self.updateScroll()
    VBox0 = QVBoxLayout()
    self.loc = QLineEdit()
    self.loc.setText(self.tempImgFile)
    refreshB = QPushButton("Refresh")
    refreshB.clicked.connect(self.updateScroll)
    showUserB = QPushButton("Ganti Password Admin")
    showUserB.clicked.connect(self.changePasswordMenu)
    tambahBarang = QPushButton("Tambah Barang")
    tambahBarang.clicked.connect(self.tambahBarang)
```

```
        ubahBarang = QPushButton("Ubah Barang")
        ubahBarang.clicked.connect(self.tambahBarang)
        hapusBarang = QPushButton("Hapus Barang")
        hapusBarang.clicked.connect(self.hapusBarang)
        buttonK = QPushButton("LogOut")
        buttonK.clicked.connect(self.MainMenu)
        VBox0.addWidget(showUserB)
        VBox0.addWidget(tambahBarang)
        VBox0.addWidget(ubahBarang)
        VBox0.addWidget(hapusBarang)
        VBox0.addWidget(buttonK)
        self.VBox1 = QVBoxLayout()
        widg = self.scrollWidget()
        self.VBox1.addWidget(widg)
        self.label = QPushButton("Finalkan Pembelian")
        self.label.clicked.connect(self.checkout)
        HBox1 = QHBoxLayout()
        HBox1.addWidget(self.label)
        HBox1.addWidget(refreshB)
        self.VBox1.addLayout(HBox1)
        HBox0 = QHBoxLayout()
        HBox0.addLayout(VBox0)
        HBox0.addLayout(self.VBox1)
        widget = QWidget()
        widget.setLayout(HBox0)
        self.setCentralWidget(widget)

    def changePasswordMenu(self):
        self.update_password()
        lab1 = QLabel("Masukkan Password Sebelumnya: ")
        lab2 = QLabel("Masukkan Password Baru:       ")
        lab3 = QLabel("Masukkan Ulang Password Baru: ")

        lineIn1 = QLineEdit()
        lineIn2 = QLineEdit()
```

```
        lineIn3 = QLineEdit()

        submit = QPushButton("Selesai")
        submit.clicked.connect(lambda: self.changeProcess(lineIn1.text(), lineIn2.text(),
lineIn3.text()))
        back = QPushButton("Kembali")
        back.clicked.connect(self.adminMenu)

        H1 = QHBoxLayout()
        H1.addWidget(lab1)
        H1.addWidget(lineIn1)

        H2 = QHBoxLayout()
        H2.addWidget(lab2)
        H2.addWidget(lineIn2)

        H3 = QHBoxLayout()
        H3.addWidget(lab3)
        H3.addWidget(lineIn3)

        H4 = QHBoxLayout()
        H4.addWidget(back)
        H4.addWidget(submit)

        Vbox = QVBoxLayout()
        Vbox.addLayout(H1)
        Vbox.addLayout(H2)
        Vbox.addLayout(H3)
        Vbox.addLayout(H4)

        widg = QWidget()
        widg.setLayout(Vbox)
        self.setCentralWidget(widg)

    def changeProcess(self, curr, new, rnew):
```

```
        if curr == self.admin["admin"]:
            if new == rnew:
                self.change_password(new)
                self.error = "Sukses!"
                self.errorPop()
                self.adminMenu()
                return
            self.error = "Sandi Baru Tidak Sama!"
            self.errorPop()
            self.adminMenu()
            return
        self.error = "Sandi Salah!"
        self.errorPop()
        self.adminMenu()

    def tambahBarang(self):
        kembali = QPushButton("Kembali")
        kembali.clicked.connect(self.adminMenu)
        input_N = QLabel("Nama Barang: ")
        input_NB = QLineEdit()
        name_box = QHBoxLayout()
        name_box.addWidget(input_N)
        name_box.addWidget(input_NB)
        amnt_N = QLabel("Jumlah Barang: ")
        amnt_NB = QLineEdit()
        amnt_box = QHBoxLayout()
        amnt_box.addWidget(amnt_N)
        amnt_box.addWidget(amnt_NB)
        prc_N = QLabel("Harga Satuan: ")
        prc_NB = QLineEdit()
        price_box = QHBoxLayout()
        price_box.addWidget(prc_N)
        price_box.addWidget(prc_NB)
        img_N = QLabel("Pilih Lokasi Gambar: ")
        img_NB = QPushButton("Pilih")
```

```
        img_NB.clicked.connect(self.openImgFile)
        img_box = QHBoxLayout()
        img_box.addWidget(img_N)
        img_box.addWidget(img_NB)
        info_N = QLabel("Informasi Barang: ")
        info_NB = QLineEdit()
        info_box = QHBoxLayout()
        info_box.addWidget(info_N)
        info_box.addWidget(info_NB)
        VBox0 = QVBoxLayout()
        VBox0.addLayout(name_box)
        VBox0.addLayout(amnt_box)
        VBox0.addLayout(price_box)
        VBox0.addLayout(img_box)
        VBox0.addWidget(self.loc)
        VBox0.addLayout(info_box)
        konfirmasi = QPushButton("Konfirmasi")
        konfirmasi.clicked.connect(lambda:      self.inputted(input_NB.text(),      amnt_NB.text(),
prc_NB.text(), self.loc.text(), info_NB.text()))
        HBox = QHBoxLayout()
        HBox.addWidget(kembali)
        HBox.addWidget(konfirmasi)
        VBox0.addLayout(HBox)
        widg = QWidget()
        widg.setLayout(VBox0)
        self.setCentralWidget(widg)

    def inputted(self, nama, amnt, prc, img, info):
        if "\\" in img:
            img = img.replace("\\", "/")
        self.IC.addItem(nama.lower(), nama, int(amnt), int(prc), img, info)
        self.tempImgFile = ""
        self.loc.setText(self.tempImgFile)
        self.adminMenu()
```

```
def avalNamaBarang(self):
    data = self.IC.get()
    VBox0 = QVBoxLayout()
    scrollable = Scrolly()
    for key in data:
        scrollable.add_widget(itemBoxDetail(key, data[key].amnt, data[key].price))
    VBox0.addWidget(scrollable)
    return VBox0

def hapusBarang(self):
    kembali = QPushButton("Kembali")
    kembali.clicked.connect(self.adminMenu)
    HBox = QHBoxLayout()
    VBox = QVBoxLayout()
    x = self.avalNamaBarang()
    VBox.addLayout(x)
    HBox0 = QHBoxLayout()
    name = QLabel("Nama Barang: ")
    inputN = QLineEdit()
    HBox0.addWidget(name)
    HBox0.addWidget(inputN)
    VBox.addLayout(HBox0)
    konfirmasi = QPushButton("Konfirmasi")
    konfirmasi.clicked.connect(lambda: self.delBarang(inputN.text()))
    HBox.addWidget(kembali)
    HBox.addWidget(konfirmasi)
    VBox.addLayout(HBox)
    widget = QWidget()
    widget.setLayout(VBox)
    self.setCentralWidget(widget)

def delBarang(self, name):
    self.IC.delItem(name)
    self.hapusBarang()
    self.updateScroll()
```

```
    def openImgFile(self):
        filname, _ = QFileDialog.getOpenFileName(self, 'Load File')
        self.tempImgFile = filname
        self.loc.setText(self.tempImgFile)

if __name__ == '__main__':
    app = QApplication([])
    login_window = LoginWindow()
    login_window.show()
    app.exec()
```

Penjelasan sourcecode:

Kode diatas merupakan Fasilitas pemesanan makanan menggunakan Bahasa python untuk membantu seseorang memesan makanan dari rumah tanpa membuang waktu di restoran. Pada kode from PyQt6.QtWidgets import QScrollArea, QDialog, QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QFileDialog from PyQt6.QtGui import QPixmap, QGuiApplication, QIcon from PyQt6.QtCore import Qt Dengan menggunakan framework PyQt6 memungkinkan penggunaan Qt dalam bahasa Python, sehingga memudahkan pengembangan aplikasi desktop yang kaya fitur dan dapat dijalankan di berbagai sistem operasi karena PyQt6 menyediakan berbagai fitur untuk membangun aplikasi desktop seperti pembuatan jendela, pengaturan tata letak, penggunaan widget seperti tombol, kotak teks, dan table.

Selanjutnya terdapat beberapa kelas pada class printText(QDialog): berisi sebuah jendela dialog yang digunakan dalam aplikasi untuk mencetak nota. Ketika objek dari kelas ini dibuat, konstruktor __init__ akan mengatur beberapa atribut seperti item, order, dan dict. Kemudian, metode PrintText digunakan untuk menghasilkan teks nota yang akan dicetak. Metode ini mengatur teks dengan informasi toko, kode nota, tanggal, daftar pesanan, dan total harga. Setelah teks dibuat, metode showD dipanggil untuk menampilkan jendela dialog dengan teks nota yang telah dibuat. Di jendela dialog tersebut, terdapat tombol "Cetak" yang, ketika diklik, akan memicu metode callF. Metode callF akan meminta pengguna untuk memilih lokasi penyimpanan dan menyimpan teks nota ke dalam file dengan lokasi tersebut.

Pada class ItemNodes(): digunakan untuk membuat objek yang merepresentasikan item dalam aplikasi. Konstruktor __init__ menerima beberapa parameter seperti name, amnt, price, img, dan information yang digunakan untuk menginisialisasi atribut-atribut pada objek.

Atribut name menyimpan nama item, amnt menyimpan jumlah item, price menyimpan harga item, img menyimpan gambar item, dan information menyimpan informasi tambahan tentang item. Kelas ini juga memiliki metode check yang di-comment (dikomen) pada kode. Metode digunakan untuk memeriksa nilai-nilai atribut pada objek.

Pada class ItemsCore(): merupakan kelas inti yang digunakan dalam manajemen item dalam aplikasi. Konstruktor __init__ menginisialisasi atribut-atribut seperti itemNodesHolder yang merupakan kamus untuk menyimpan objek ItemNodes dengan kunci sebagai nama item, orderedItems yang merupakan kamus untuk menyimpan item-item yang dipesan, dan filepath yang merupakan jalur file untuk menyimpan data item. Kelas ini memiliki berbagai metode, seperti checkAval untuk memeriksa ketersediaan item, get untuk mengambil daftar item, getOrder untuk mengambil item yang dipesan, incItem untuk menambahkan jumlah item yang dipesan, decItem untuk mengurangi jumlah item yang dipesan, saveCurrentItems untuk menyimpan data item ke file, loadItems untuk memuat data item dari file, addItem untuk menambahkan item baru, delItem untuk menghapus item, dan resetOrder untuk mengatur ulang pesanan. Kelas ini digunakan untuk mengelola dan menyediakan fungsi-fungsi utama terkait item dalam aplikasi.

Pada class ErrorPopUp(QDialog): digunakan untuk menampilkan pesan kesalahan dalam bentuk pop-up pada aplikasi. Konstruktor __init__ mengatur judul jendela pop-up dan mengatur geometri pop-up. Metode error digunakan untuk menampilkan pesan kesalahan yang diterima sebagai argumen. Metode ini membuat tata letak vertikal (VBox) dan menambahkan label dengan pesan kesalahan yang ditengahkan. Pop-up ini digunakan untuk memberi tahu pengguna tentang kesalahan yang terjadi dalam aplikasi.

Pada class itemBoxDetail(QWidget): yang merupakan widget yang digunakan untuk menampilkan detail item dalam kotak item. Konstruktor __init__ menerima argumen name (nama item), itemAmnt (jumlah item), dan price (harga item). Kelas ini merupakan turunan dari kelas QWidget. Variabel self.type digunakan untuk menyimpan tipe item, namun tidak digunakan dalam kode yang diberikan. Dalam konstruktor, label-label self.name, self.amntBox, dan self.priceBox dibuat untuk menampilkan nama item, jumlah item, dan harga item secara berurutan dan digunakan untuk menampilkan detail item atau daftar belanja.

Pada class itemDetail(QWidget): yang merupakan turunan dari kelas QWidget digunakan untuk menampilkan detail item dalam sebuah widget. Terdapat dua metode updateItem. Metode pertama digunakan untuk mengupdate detail item dengan argumen name, itemAmnt, dan price. Dalam metode ini, label-label self.nameIT, self.amntBox, dan self.priceBox diperbarui dan ditambahkan ke dalam tata letak horizontal (HBox). Metode kedua digunakan untuk mengupdate informasi item dengan argumen information. Pada

metode ini, sebuah objek Scrolly dibuat, label self.label diberi teks information, dan ditambahkan ke dalam tata letak horizontal (HBox). Kelas itemDetail digunakan untuk menampilkan detail item dalam bentuk label-label yang berisi nama, jumlah, dan harga item, serta dalam bentuk label dengan teks informasi.

Pada class Scrolly(QScrollArea): yang merupakan turunan dari kelas QScrollArea yang digunakan untuk membuat area scrollable di dalam widget. Pada konstruktor __init__, sebuah widget self.widget dibuat dan diatur sebagai konten dari QScrollArea. Metode add_widget digunakan untuk menambahkan widget ke dalam tata letak vertikal (QVBoxLayout) pada self.widget.

Pada class ImagedItemNode(QWidget): Yang merupakan turunan dari kelas QWidget yang digunakan untuk menampilkan detail item dalam sebuah widget dengan gambar. Pada konstruktor __init__, berbagai elemen seperti label, gambar, tombol, dan informasi item didefinisikan dan ditambahkan ke dalam tata letak vertikal (QVBoxLayout). Metode updateDetailed digunakan untuk mengatur keadaan tampilan dari self.infoBox. Metode incDecUpdate digunakan untuk mengupdate tombol plus dan minus berdasarkan ketersediaan jumlah item. Metode checkAval digunakan untuk memeriksa ketersediaan jumlah item. Metode incBI dan decBI digunakan untuk menambah atau mengurangi jumlah item yang dipesan, serta mengupdate tampilan sesuai perubahan tersebut.

Pada class LoginWindow(QMainWindow): yang merupakan kelas turunan dari QMainWindow yang digunakan untuk mengatur tampilan dan logika jendela login dalam sebuah aplikasi. Pada konstruktor __init__, beberapa atribut dan objek penting diinisialisasi, seperti pengguna default (user dan admin), error message, path file, objek ItemsCore, dan lain-lain. Terdapat pula metode-metode seperti scrollWidget untuk membuat daftar item yang dapat digulir, clearScrollWidget untuk membersihkan daftar item yang sedang ditampilkan, incItem dan decItem untuk menambah dan mengurangi jumlah item dalam keranjang, serta beberapa metode lain yaitu getUser(), getAdmin(), addUser(), delUser(), update_password(), change_password(), default(), matchPswd(), findUser(), errorPop(), checkout(),login(), UpdateDisplay(), UserMenu(), saveLogic(), loadLogic(), updateScroll(), adminMenu(), changePasswordMenu(), changeProcess(), tambahBarang(), inputted(), avalNamaBarang(), hapusBarang(), Metode MainMenu untuk menampilkan menu utama, AdminLogin untuk menampilkan jendela login admin, login untuk memverifikasi login pengguna atau admin dan beberapa metode lainnya yang digunakan untuk mengatur tampilan dan logika aplikasi manajemen pengguna, pengaturan password, pengecekan login, dan tampilan menu pengguna dan admin.

Yang terakhir jika if __name__ == '__main__': maka akan ditampilkan login_window.show() untuk menampilkan jendela login dan app.exec() untuk untuk memulai event loop, yang akan menjalankan aplikasi dan menangani interaksi pengguna. Event loop ini akan berjalan sampai aplikasi ditutup, dan setelah itu eksekusi program akan selesai.

KESIMPULAN DAN SARAN

A. Kesimpulan

Python adalah bahasa pemrograman tingkat tinggi yang mudah dipelajari dan dipahami, serta memiliki sintaksis yang sederhana dan mudah dibaca. Python banyak digunakan dalam berbagai bidang, seperti data science, machine learning, web development, dan sebagainya.

Fleksibilitas Python menjadi faktor kunci dalam memilih bahasa pemrograman. Python mendukung paradigma pemrograman berorientasi objek, memungkinkan pengembang untuk mengelola informasi seperti daftar menu, detail pesanan, dan informasi pelanggan dalam aplikasi pemesanan makanan. Selain itu, Python juga mendukung pemrograman fungsional, memungkinkan pengembang untuk menulis kode yang mudah dipahami dan dikelola. Salah satu kekuatan Python terletak pada ketersediaan modul dan pustaka yang melimpah. Dalam pembuatan sistem aplikasi pemesanan makanan, pengembang dapat menggunakan modul seperti PyQt6 untuk membangun antarmuka pengguna yang responsif dan interaktif, serta mengelola basis data pesanan dan pelanggan.

Skalabilitas juga menjadi keunggulan Python dalam mengembangkan aplikasi pemesanan makanan. Dengan dukungan untuk pemrograman paralel dan pemrosesan asinkron, Python memungkinkan aplikasi menangani beban kerja yang berat dengan respons yang cepat. Dalam pengembangan, pendekatan desain dan pengembangan yang tepat akan memastikan aplikasi pemesanan makanan yang dibangun dengan Python mampu mengatasi peningkatan lalu lintas dengan lancar. Sehingga Python adalah pilihan yang ideal untuk mengembangkan sistem aplikasi pemesanan makanan. Dengan fleksibilitas, dukungan modul dan pustaka yang melimpah, kemampuan integrasi yang mudah, serta skalabilitas yang baik.

B. Saran

Melakukan perencanaan yang baik dan analisis mendalam tentang kebutuhan sistem manajemen pemesanan makanan. Identifikasi fitur utama yang diperlukan, seperti pengelolaan menu, Hal ini akan membantu dalam merancang arsitektur dan struktur basis data yang efisien. Selanjutnya pemilihan framework Python yang sesuai untuk membangun sistem aplikasi pemesanan makanan, seperti PyQt6 yang memungkinkan pengembang menggunakan Qt dalam bahasa Python, sehingga memudahkan pengembangan aplikasi desktop yang kaya fitur dan dapat dijalankan di berbagai sistem operasi. PyQt6 menyediakan berbagai fitur untuk membangun aplikasi desktop seperti pembuatan jendela, pengaturan tata letak, penggunaan widget seperti tombol, kotak teks, dan table.

DAFTAR PUSTAKA

Huda, A. (2020). DASAR-DASAR PEMROGRAMAN BERBASIS PYTHON. In A. Huda, *DASAR-DASAR PEMROGRAMAN BERBASIS PYTHON* (p. 117). Padang: UNP Press.

Josikie. (2021, Juli 26). Sejarah Lahirnya Bahasa Pemprograman Python. Retrieved 2 Juni 2023, from https://josikie.com/sejarah-lahirnya-bahasa-pemrograman-python/

Khoirudin. (2019). Algoritma dan Struktur Data Dengan Python 3. In Khoirudin., *Algoritma dan Struktur Data Dengan Python 3* (p. 75). Semarang: Universitas Semarang Press.

Kurniawan, T. (2018). Input dan output pada bahasa pemograman python. Retrieved 4 Juni 2023, from https://www.researchgate.net/profile/Tedi-Kurniawan-2/publication/338385483_INPUT_DAN_OUTPUT_PADA_BAHASA_PEMROGRAMAN_PYTHON/links/5e10643392851c8364b029c3/INPUT-DAN-OUTPUT-PADA-BAHASA-PEMROGRAMAN-PYTHON.pdf. *Jurnal Dasar Pemograman Python.*

Muhammad Romzi, Budi Kurniawan. (2020). Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma. *Jurnal Teknik Informatika Mahakarya*, 37.

Septian, R. F. (2013). Belajar Pemrograman Python Dasar. In R. F. Septian, *Belajar Pemrograman Python Dasar* (p. 108). Jawa Barat: POSS – UPI..

Trisno, I. B. (2016). BELAJAR PEMROGRAMAN SULIT ? COBA PYTHON. In I. B. Trisno, *BELAJAR PEMROGRAMAN SULIT ? COBA PYTHON* (p. 71). Surabaya: Ubhara Manajemen Press Surabaya.

Zein, A. (2018). Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka OPENCV dan DLIB PYTHON. *Jurnal Penelitian dan Pengkajian Sains dan Teknologi*.

KEMENTRIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
UNIVERSITAS BENGKULU
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA
Jl. Wr. Supratman Kandang Limun, Bengkulu
Bengkulu 38371 A Telp: (0736) 344087, 22105 - 227

LEMBAR ASISTENSI
UJIAN AKHIR SEMESTER PEMROGRAMAN BERORIENTASI OBJEK

Nama Mahasiswa : 1. Natasya Salsabilla (G1A022023)
2. Alif Nurhidayat (G1A022073)
3. Saniyyah Zhafirah (G1A022081)

Asisten Dosen : 1. Randy Julian S (G1A019066)

Dosen Pengampu : 1. Arie Vatresia, S.T,. M.TI, Ph.D
2. Mochammad Yusa, S. Kom., M. Kom.

Ujian Akhir Semester	Catatan dan Tanda Tangan
Ujian Akhir Semester	