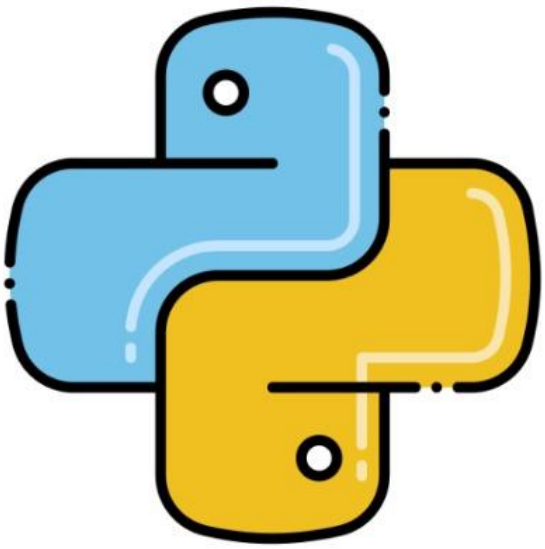
LAPORAN PRAKTIKUM PBO

APLIKASI PENGHITUNGAN KALKULATOR SEDRHANA MENGGUNAKAN METODE PYQT & QTDESAINER

Dosen Pengampu : Freddy Wicaksono,M.Kom



Disusun oleh:

Rafi Zaidan Rabbani : 220511074

M Farhan Saino : 220511035

M Yusuf Kurnia : 220511122

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH CIREBON
Jl. Fatahilah, Watubelah,Kec. Sumber
Kabupaten Cirebon, Jawa Barat 45611

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga dapat menyelesaikan tugas makalah yang berjudul kalkulator sederhana ini. Bertujuan untuk memenuhi tugas Bapak Freddy Wicaksono, M,Kom pada Mata kuliah Pemrograman Berorientasi Objek. Selain itu, makalah ini juga bertujuan untuk menambah wawasan tentang kalkulator sederhana

Terima kasih kepada Bapak Freddy Wicaksono, M.Kom, selaku Dosen Mata kuliah Pemrograman Berorientasi Objek yang telah memberikan tugas ini sehingga dapat menambah pengetahuan dan wawasan sesuai dengan bidang studi yang kami tekuni. juga terima kasih kepada semua pihak yang telah membagi sebagian pengetahuannya sehingga saya dapat menyelesaikan makalah ini.

Sangat disadari, makalah ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun akan selalu dinantikan demi kesempurnaan makalah ini.

Cirebon, 13 November 2023

DAFTAR ISI

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam era teknologi modern, aplikasi perangkat lunak telah menjadi bagian integral dari kehidupan sehari-hari. Salah satu bentuk aplikasi yang umum digunakan adalah kalkulator. Kalkulator digunakan secara luas baik dalam kegiatan sehari-hari maupun dalam konteks pekerjaan yang lebih teknis.

Dengan perkembangan teknologi, pengembangan aplikasi kalkulator semakin beragam. Salah satu pendekatan yang populer adalah menggunakan framework PyQt, yang memanfaatkan kekuatan bahasa pemrograman Python dan toolkit Qt. PyQt menyediakan alat yang kuat untuk pembuatan antarmuka grafis pengguna (GUI) dengan efisiensi tinggi.

Pembuatan aplikasi kalkulator dengan PyQt tidak hanya memungkinkan pengguna untuk melakukan operasi matematika dasar dengan mudah, tetapi juga memberikan pengalaman pengguna yang lebih interaktif dan menarik. Latar belakang ini menjadi dasar penting untuk mengembangkan aplikasi kalkulator berbasis PyQt, yang tidak hanya dapat memenuhi kebutuhan praktis pengguna, tetapi juga memberikan tampilan yang estetis dan mudah digunakan.

Melalui proyek ini, diharapkan dapat memberikan pemahaman mendalam tentang pengembangan aplikasi GUI menggunakan PyQt dan memberikan kontribusi pada peningkatan keterampilan pengembangan perangkat lunak bagi para pengembang dan pembelajar pemrograman.

1.2. Tujuan

Tujuan dari praktikum ini adalah untuk memberikan pemahaman dan keterampilan kepada peserta praktikum dalam pengembangan aplikasi perhitungan kalkulator dengan menggunakan PyQt dan desain QT.Dengan mencapai tujuan praktikum ini, diharapkan peserta dapat mengembangkan keterampilan dalam mengaplikasikan konsep Pemrograman Berorientasi Objek dan menerapkannya dalam pengembangan aplikasi GUI menggunakan PyQt dan desain QT.

1.3. Ruang Lingkup

Ruang lingkup proyek ini mencakup beberapa aspek yang akan dikaji dan diimplementasikan selama pengembangan aplikasi kalkulator berbasis PyQt.Ruang lingkup ini memberikan kerangka kerja yang jelas untuk mencapai tujuan proyek dan memastikan bahwa pengembangan aplikasi kalkulator berjalan sesuai rencana dan memenuhi standar kualitas yang diinginkan.

1.4. Manfaat

- Kemudahan Penghitungan:
  Aplikasi kalkulator sederhana menyediakan alat yang cepat dan mudah untuk melakukan perhitungan matematika dasar, seperti penjumlahan, pengurangan, perkalian, dan pembagian. Pengguna dapat dengan cepat mendapatkan hasil tanpa perlu membuka kalkulator fisik atau melakukan perhitungan manual.
- Kecepatan dan Efisiensi:
  Dengan menggunakan aplikasi kalkulator, pengguna dapat menghemat waktu dan usaha dalam melakukan perhitungan sehari-hari. Hasil perhitungan dapat diperoleh secara instan, meningkatkan efisiensi dalam pemecahan masalah matematika.
- Pencegahan Kesalahan
  Aplikasi kalkulator dapat membantu mencegah kesalahan perhitungan manusia yang mungkin terjadi saat melakukan perhitungan manual. Hal ini dapat meningkatkan akurasi hasil perhitungan, terutama untuk perhitungan yang melibatkan angka yang besar atau kompleks.
- Pemecahan Masalah Sehari-hari:
  Aplikasi kalkulator dapat membantu pengguna dalam memecahkan masalah sehari-hari yang melibatkan perhitungan matematika, seperti menghitung biaya belanja, konversi unit, atau perhitungan waktu.

BAB II

DASAR TEORI

2.1. Pengenalan PyQt

PyQt adalah satu set binding (disebut juga plug-in atau library) Python untuk kerangka aplikasi Qt yang dibuat oleh perusahaan Qt dan berjalan di semua platform yang didukung oleh Qt termasuk Windows, macOS, Linux, iOS, dan Android. PyQt6 mendukung Qt v6, PyQt5 mendukung Qt v5 dan PyQt4 mendukung Qt v4.

PyQt5 adalah satu set lengkap binding Python untuk Qt v5. Dalam versi ini terdapat lebih dari 35 modul ekstensi dan memungkinkan Python untuk digunakan sebagai bahasa pengembangan aplikasi alternatif untuk C++ di semua platform yang didukung termasuk iOS dan Android[2]

2.2 Keunggulan PyQt

- Kemampuan Cross-Platform: PyQt mendukung pengembangan aplikasi yang dapat dijalankan secara lintas platform, artinya dapat berfungsi baik di sistem operasi Windows, Linux, maupun macOS.
- Antarmuka Grafis yang Menarik: Dengan Qt Designer, pengembang dapat merancang antarmuka pengguna dengan mudah dan membuat aplikasi yang estetis dan ramah pengguna.
- Kemampuan Multithreading: PyQt mendukung pengembangan aplikasi yang memanfaatkan multithreading, memungkinkan penanganan tugas-tugas yang kompleks dan berat secara efisien.
- Komunitas dan Dokumentasi yang Kuat: PyQt memiliki komunitas yang aktif dan dokumentasi yang baik, memudahkan pengembang untuk memahami dan mengatasi berbagai tantangan.

2.3 Pengenalan Qt Designer

Qt Designer adalah desainer GUI seret dan lepas lintas platform, yang dapat digunakan untuk membangun UI untuk PyQt dan PySide. Ini adalah alat yang hebat untuk menyederhanakan proses membangun antarmuka untuk aplikasi Anda.

Meskipun Qt Designer didistribusikan oleh Qt sebagai bagian dari IDE terintegrasi Qt Creator, sebagian besar IDE tersebut tidak berguna atau membantu pengembangan Python -- terutama dirancang untuk pengembang Python. Jika Anda hanya menginginkan aplikasi Desainer, saat ini tidak tersedia dari Qt.

Pada Python sendiri sebenarnya terdapat beberapa API yang dapat digunakan untuk membuat GUI, seperti wxPython, TKInter, PyGtk, ataupun PyQt. Untuk list lengkap API yang dapat digunakan untuk membuat GUI pada Python dapat dilihat disini.

QtDesigner sendiri adalah tools yang dapat digunakan untuk mendesain dan mem-build Graphical User Interfaces menggunakan QtComponents. QtDesigner dapat digunakan tidak hanya untuk membuat aplikasi desktop, tapi juga aplikasi mobile.

2.4. Keunggulan Qt Designer

- WYSIWYG Editing (What You See Is What You Get): Qt Designer memungkinkan pengembang untuk melihat tampilan akhir antarmuka pengguna sebagaimana adanya selama proses perancangan, memudahkan penyesuaian dan perbaikan desain.
- Dukungan untuk Layout Management: Qt Designer menyediakan fitur layout management yang mempermudah pengaturan posisi dan tata letak elemen-elemen antarmuka, sehingga menciptakan desain yang responsif.
- 3-Integrasi dengan Kode Python: Antarmuka yang dirancang di Qt Designer dapat diintegrasikan langsung dengan kode Python, memungkinkan implementasi fungsionalitas tanpa harus menulis desain GUI secara manual.
- Dukungan untuk Pembuatan Dialog dan Widget: Qt Designer tidak hanya digunakan untuk merancang tampilan utama aplikasi, tetapi juga untuk membuat dialog dan widget tambahan dengan mudah.

2.5. Konsep Dasar Kalkulator

Konsep dasar kalkulator mencakup operasi matematika dasar, input dan output melalui antarmuka pengguna, serta fitur tambahan untuk meningkatkan fungsionalitas. Implementasi konsep ini akan menjadi dasar untuk merancang dan mengembangkan aplikasi kalkulator dengan menggunakan PyQt.

BAB III

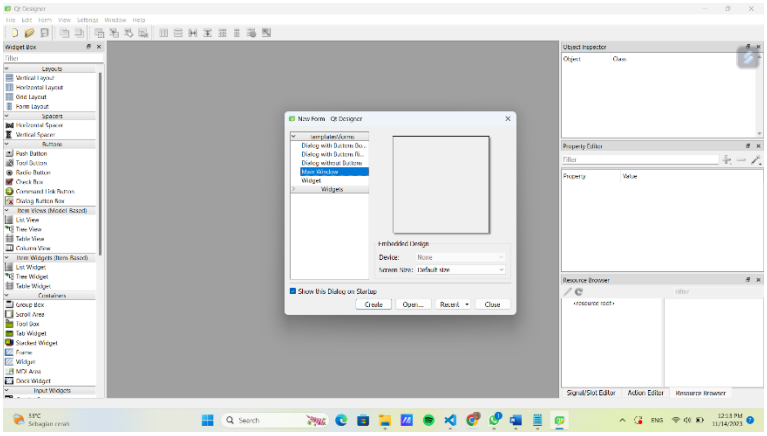
PEMBAHASAN

3.1. Instalasi PyQt5

- Untuk menginstal PyQt di Windows,kita bisa menggunakan paket installer dari website qt-project.
- Untuk Python versi 3.3 atau versi terbaru,kita bisa menginstall semua package pyqt dengan cata mencentang pada box features saat kita menginstall python.
- Untuk python versi 3.6 keatas bisa menggunakan perintah dibawah ini: *"pip install pyqt5"*

3.2. Instalasi Qt Designer

- Langkah pertama bukalah situs https://build-system.fman.io/qt-designer-download untuk mendownload Qt Designer
- Selanjutnya jika sudah terdownload, buka aplikasi isntalasi Qt designer nya lalu install sesuai dengan perintah yang di berikan
- Setelah proses instalasi selesai, kemudian bukalah aplikasi Qt Designer yang sudah terinstall lalu akan muncul halaman windows sebagai berikut



- Aplikasi Qt Designer sudah terinstall anda bisa langsung menggunakan Qt Designer sesuai yang anda inginkan

3.3.Antarmuka Aplikasi Kalkulator

Sebuah antarmuka kalkulator sederhana biasanya terdiri dari tombol-tombol angka, operator matematika, dan beberapa fungsi tambahan. Berikut adalah penjelasan untuk antarmuka kalkulator sederhana:

- **Layar Kalkulator(QLineEdit):**

  Tempat di mana angka dan hasil perhitungan ditampilkan.

  Menampilkan angka yang dimasukkan oleh pengguna dan hasil perhitungan.

- **Tombol Angka (PushButton 0-9):**

  Digunakan untuk memasukkan angka ke dalam kalkulator.

  Setiap tombol mewakili satu digit angka.

- **Tombol Operator Matematika(PushButton *,+,-,/):**

  *+, -, , /: Tombol ini digunakan untuk melakukan operasi penjumlahan, pengurangan, perkalian, dan pembagian.

  Biasanya disusun dengan jelas untuk memudahkan pengguna memilih operasi yang diinginkan.

- **Tombol Hapus (PushButton C):**

  C (Clear): Menghapus seluruh masukan dan perhitungan.

- **Tombol Samakan (PushButton =):**

  Digunakan untuk mengeksekusi perhitungan dan menampilkan hasilnya di layar.

3.4.Konversi File Ui to Py

Jika anda sudah selesain mendesain Qt Designer sesuai keinginan anda,
selanjutnya adalah meng-convert file dari UI ke Py:

- Pertama cari terlebih dahulu tempat anda menyimpan file UI yang anda simpan dari Qt Designer
- Selanjutnya buka Windows Power Shell, jika sudah terbuka silahkan arahkan direktori ke tempat anda menyimpan file UI dari Qt Designer
- Selanjutnya masukan kode : pyuic5 -x “.ui -o “.py, silahkan ubah tanda “”(petik) menjadi nama file.UI yang anda simpan dari Qt Desginer
- Jika sudah file Py akan muncul, hasil dari convert UI ke Py

3.5.Perancangan Kode

- **Struktur Proyek**

Berikut ini adalah hasil dari konversi UI menjadi source code Py

```
from PyQt5.QtCore import (QCoreApplication, QDate, QDataStream, QMetaObject,)
from PyQt5.QtGui import (QBrush, QColor, QConicalGradient, QCursor, QFont )
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout



class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(269, 385)
        MainWindow.setMinimumSize(QtCore.QSize(0, 0))
        MainWindow.setMaximumSize(QtCore.QSize(269, 385))
        MainWindow.setStyleSheet("background-color: rgb(85, 170, 255);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout.setObjectName("verticalLayout")
        self.txtResult = QtWidgets.QLineEdit(self.centralwidget)
        self.txtResult.setMinimumSize(QtCore.QSize(0, 41))
        self.txtResult.setMaximumSize(QtCore.QSize(251, 41))
        font = QtGui.QFont()
        font.setFamily("Muli SemiBold")
        font.setPointSize(20)
        font.setBold(True)
        font.setWeight(75)
        self.txtResult.setFont(font)
        self.txtResult.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.txtResult.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
        self.txtResult.setObjectName("txtResult")
        self.verticalLayout.addWidget(self.txtResult)
        self.gb_angka = QtWidgets.QGroupBox(self.centralwidget)
        self.gb_angka.setMinimumSize(QtCore.QSize(251, 279))
        self.gb_angka.setMaximumSize(QtCore.QSize(251, 279))
        self.gb_angka.setObjectName("gb_angka")
        self.btn1 = QtWidgets.QPushButton(self.gb_angka)
        self.btn1.setGeometry(QtCore.QRect(10, 20, 51, 51))
font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn1.setFont(font)
        self.btn1.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn1.setText("1")
        self.btn1.setObjectName("btn1")
        self.btn2 = QtWidgets.QPushButton(self.gb_angka)
        self.btn2.setGeometry(QtCore.QRect(70, 20, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn2.setFont(font)
        self.btn2.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn2.setText("2")
        self.btn2.setObjectName("btn2")
        self.btn3 = QtWidgets.QPushButton(self.gb_angka)
        self.btn3.setGeometry(QtCore.QRect(130, 20, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn3.setFont(font)
```

```
        self.btn3.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn3.setText("3")
        self.btn3.setObjectName("btn3")
        self.btnMin = QtWidgets.QPushButton(self.gb_angka)
        self.btnMin.setGeometry(QtCore.QRect(190, 20, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btnMin.setFont(font)
        self.btnMin.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btnMin.setText("-")
        self.btnMin.setObjectName("btnMin")
        self.btn6 = QtWidgets.QPushButton(self.gb_angka)
        self.btn6.setGeometry(QtCore.QRect(130, 80, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn6.setFont(font)
        self.btn6.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn6.setText("6")
        self.btn6.setObjectName("btn6")
        self.btnDivide = QtWidgets.QPushButton(self.gb_angka)
        self.btnDivide.setGeometry(QtCore.QRect(190, 80, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btnDivide.setFont(font)
        self.btnDivide.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btnDivide.setText("/")
        self.btnDivide.setObjectName("btnDivide")
        self.btn4 = QtWidgets.QPushButton(self.gb_angka)
        self.btn4.setGeometry(QtCore.QRect(10, 80, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn4.setFont(font)
        self.btn4.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn4.setText("4")
        self.btn4.setObjectName("btn4")
        self.btn5 = QtWidgets.QPushButton(self.gb_angka)
        self.btn5.setGeometry(QtCore.QRect(70, 80, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn5.setFont(font)
        self.btn5.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn5.setText("5")
        self.btn5.setObjectName("btn5")
        self.btn9 = QtWidgets.QPushButton(self.gb_angka)
        self.btn9.setGeometry(QtCore.QRect(130, 140, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn9.setFont(font)
        self.btn9.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn9.setText("9")
        self.btn9.setObjectName("btn9")
        self.btn7 = QtWidgets.QPushButton(self.gb_angka)
        self.btn7.setGeometry(QtCore.QRect(10, 140, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn7.setFont(font)
```

```
        self.btn7.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn7.setText("7")
        self.btn7.setObjectName("btn7")
        self.btnPlus = QtWidgets.QPushButton(self.gb_angka)
        self.btnPlus.setGeometry(QtCore.QRect(190, 140, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btnPlus.setFont(font)
        self.btnPlus.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btnPlus.setText("+")
        self.btnPlus.setObjectName("btnPlus")
        self.btn8 = QtWidgets.QPushButton(self.gb_angka)
        self.btn8.setGeometry(QtCore.QRect(70, 140, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn8.setFont(font)
        self.btn8.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn8.setText("8")
        self.btn8.setObjectName("btn8")
        self.btnClear = QtWidgets.QPushButton(self.gb_angka)
        self.btnClear.setGeometry(QtCore.QRect(10, 200, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btnClear.setFont(font)
        self.btnClear.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btnClear.setText("c")
        self.btnClear.setObjectName("btnClear")
        self.btn0 = QtWidgets.QPushButton(self.gb_angka)
        self.btn0.setGeometry(QtCore.QRect(70, 200, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btn0.setFont(font)
        self.btn0.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btn0.setText("0")
        self.btn0.setObjectName("btn0")
        self.btnKali = QtWidgets.QPushButton(self.gb_angka)
        self.btnKali.setGeometry(QtCore.QRect(130, 200, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btnKali.setFont(font)
        self.btnKali.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btnKali.setText("*")
        self.btnKali.setObjectName("btnKali")
        self.btnHasil = QtWidgets.QPushButton(self.gb_angka)
        self.btnHasil.setGeometry(QtCore.QRect(190, 200, 51, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.btnHasil.setFont(font)
        self.btnHasil.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.btnHasil.setText("=")
        self.btnHasil.setObjectName("btnHasil")
        self.verticalLayout.addWidget(self.gb_angka)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 269, 21))
        self.menubar.setObjectName("menubar")
        self.menufile = QtWidgets.QMenu(self.menubar)
        self.menufile.setObjectName("menufile")
```

```
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionQuit = QtWidgets.QAction(MainWindow)
        self.actionQuit.setObjectName("actionQuit")
        self.menufile.addAction(self.actionQuit)
        self.menubar.addAction(self.menufile.menuAction())

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Simple Calculator"))
        self.gb_angka.setTitle(_translate("MainWindow", "Kalkulator"))
        self.menufile.setTitle(_translate("MainWindow", "file"))
        self.actionQuit.setText(_translate("MainWindow", "Quit"))
        self.actionQuit.setShortcut(_translate("MainWindow", "Ctrl+Q"))
```

3.6.Perancangan Kode Inisialisasi Apikasi

Berikut ini adalah source code dari fungsi program desain aplikasi yang sudah kita buat menggunakan Qt Designer :

```
    class Calculator(QMainWindow, Ui_MainWindow):
def __init__(self):
    super(Calculator, self).__init__()

    self.setupUi(self)

    self.result = "0"
    self.operator = ""

    #Quit Menu Option
    self.actionQuit.triggered.connect(self.close)

    # Button Presses
    self.btn0.clicked.connect(self.nolPressed)
    self.btn1.clicked.connect(self.satuPressed)
    self.btn2.clicked.connect(self.duaPressed)
    self.btn3.clicked.connect(self.tigaPressed)
    self.btn4.clicked.connect(self.empatPressed)
    self.btn5.clicked.connect(self.limaPressed)
    self.btn6.clicked.connect(self.enamPressed)
    self.btn7.clicked.connect(self.tujuhPressed)
    self.btn8.clicked.connect(self.delapanPressed)
    self.btn9.clicked.connect(self.sembilanPressed)

    #clear Button
    self.btnClear.clicked.connect(self.clear)

    #Operation Button
    self.btnPlus.clicked.connect(self.plus)
    self.btnMin.clicked.connect(self.minus)
    self.btnDivide.clicked.connect(self.bagi)
    self.btnKali.clicked.connect(self.kali)
```

```
        #Hasil Button
        self.btnHasil.clicked.connect(self.calculate)
    def nolPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}0")
        else:
            self.txtResult.setText("0")

    def satuPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}1")
        else:
            self.txtResult.setText("1")

    def duaPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}2")
        else:
            self.txtResult.setText("2")

    def tigaPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}3")
        else:
            self.txtResult.setText("3")

    def empatPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}4")
        else:
            self.txtResult.setText("4")

    def limaPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}5")
        else:
            self.txtResult.setText("5")

    def enamPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}6")
        else:
            self.txtResult.setText("6")

    def tujuhPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}7")
```

```
        else:
            self.txtResult.setText("7")

    def delapanPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}8")
        else:
            self.txtResult.setText("8")

    def sembilanPressed(self):
        val = self.txtResult.text()
        if not val == "0":
            self.txtResult.setText(f"{val}9")
        else:
            self.txtResult.setText("9")

    def clear(self):
        self.txtResult.setText("0")
```

3.7.Perancangan Kode Penanganan Operasi Matematika

Berikut ini adalah kode fungsi penanganan operasi matematika dari aplikasi kalkulator sederhana

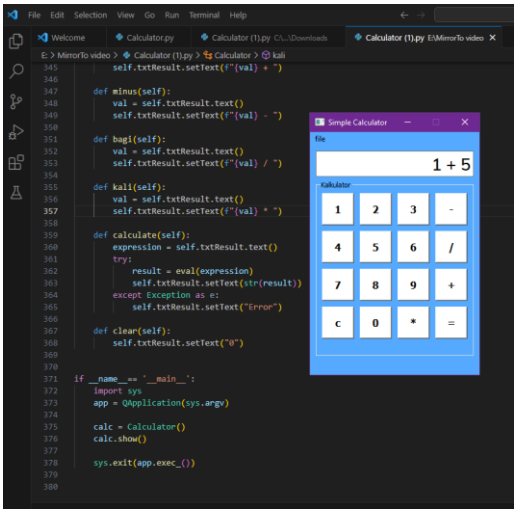
```
    def plus(self):
        val = self.txtResult.text()
        self.txtResult.setText(f"{val} + ")

    def minus(self):
        val = self.txtResult.text()
        self.txtResult.setText(f"{val} - ")

    def bagi(self):
        val = self.txtResult.text()
        self.txtResult.setText(f"{val} / ")

    def kali(self):
        val = self.txtResult.text()
        self.txtResult.setText(f"{val} * ")

    def calculate(self):
        expression = self.txtResult.text()
        try:
            result = eval(expression)
            self.txtResult.setText(str(result))
        except Exception as e:
            self.txtResult.setText("Error")
    def clear(self):
        self.txtResult.setText("0")
if __name__== '__main__':
    import sys
    app = QApplication(sys.argv)
```

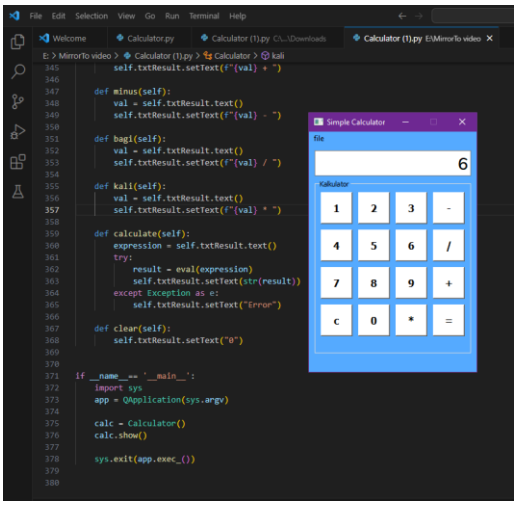
```
    calc = Calculator()
    calc.show()

    sys.exit(app.exec_())
```

3.8.Pengujian Fungsionalitas

A. Pengujian Penambahan

Masukan dua angka positif dan tekan tombol “+”, gambar di sebelah kiri merupakan operasi dari penjumlahan sedangkan gambar di sebelah kanan merupakan hasil dari operasi penjumlahan.

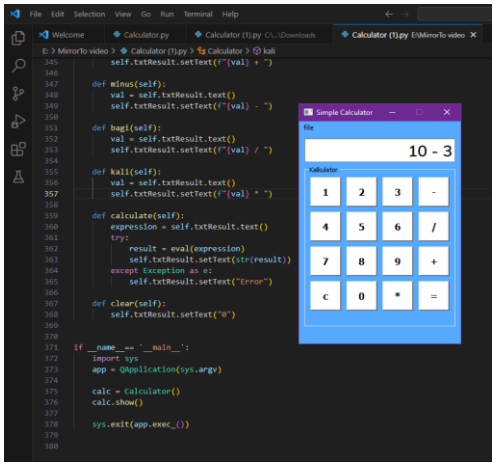


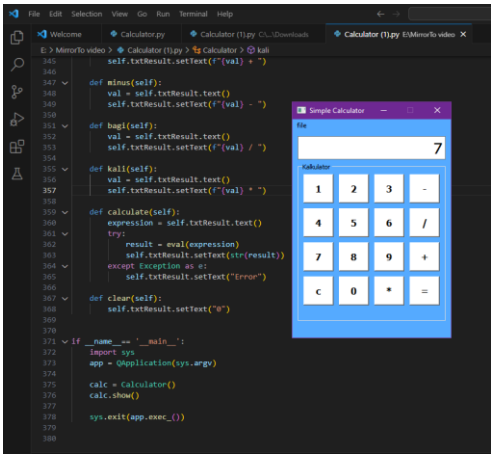


disini ketika tombol sama dengan di tekan akan menggantikan tampilan operasi yang digunakan dan menampilkan hasil dari operasi yang di gunakan

B. Pengujian Pengurangan

Masukan dua angka dan tekan tombol “-“,gambar di sebelah kiri merupakan operasi dari pengurangan sedangkan gambar di sebelah kanan merupakan hasil dari operasi pengurangan.

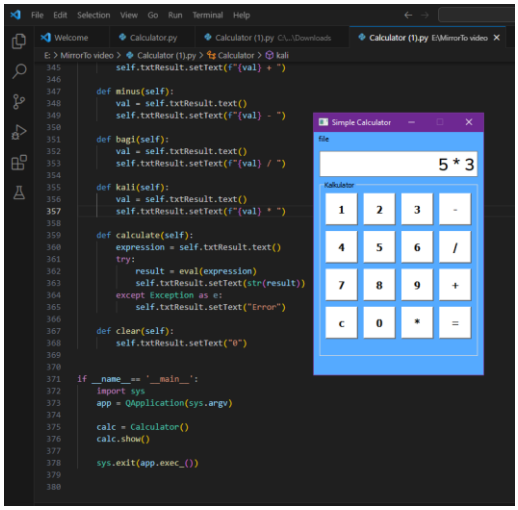


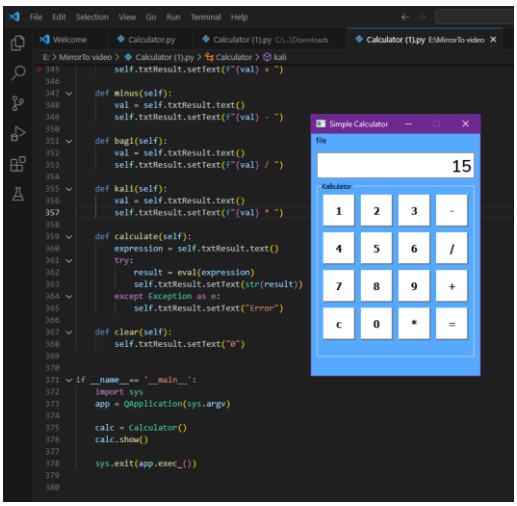


disini ketika tombol sama dengan di tekan akan menggantikan tampilan operasi yang digunakan dan menampilkan hasil dari operasi yang di gunakan

C. Pengujian Perkalian

Input dua angka lalu tekan tombol “*”,gambar di sebelah kiri merupakan operasi dari perkalian sedangkan gambar di sebelah kanan merupakan hasil dari operasi perkalian

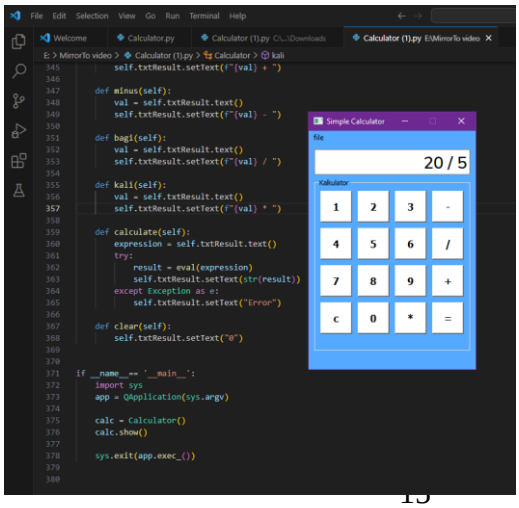


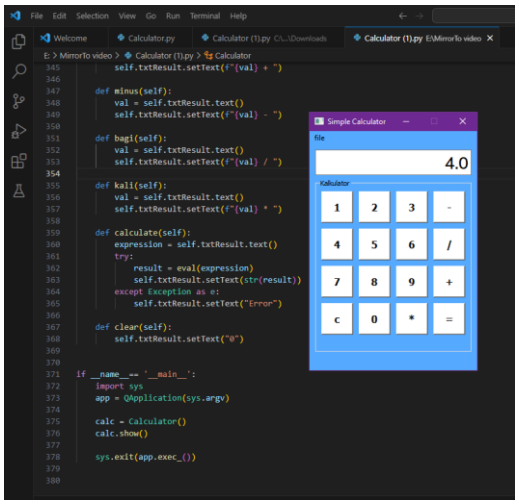


disini ketika tombol sama dengan di tekan akan menggantikan tampilan operasi yang digunakan dan menampilkan hasil dari operasi yang di gunakan

D. Pengujian Pembagian

Input dua angka (pastikan angka tidak nol) dan tekan tombol “/”,gambar di sebelah kiri merupakan operasi dari pembagian sedangkan gambar di sebelah kanan merupakan hasil dari operasi pembagian





disini ketika tombol sama dengan di tekan akan menggantikan tampilan operasi yang digunakan dan menampilkan hasil dari operasi yang di gunakan

BAB IV
PENUTUP

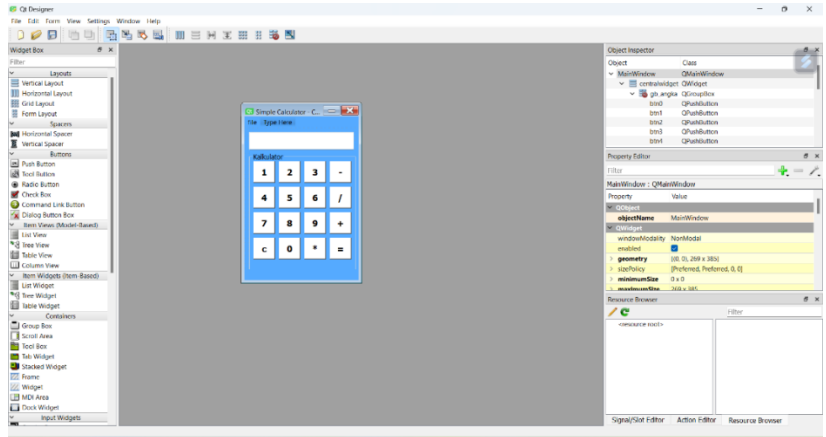
4.1. Kesimpulan

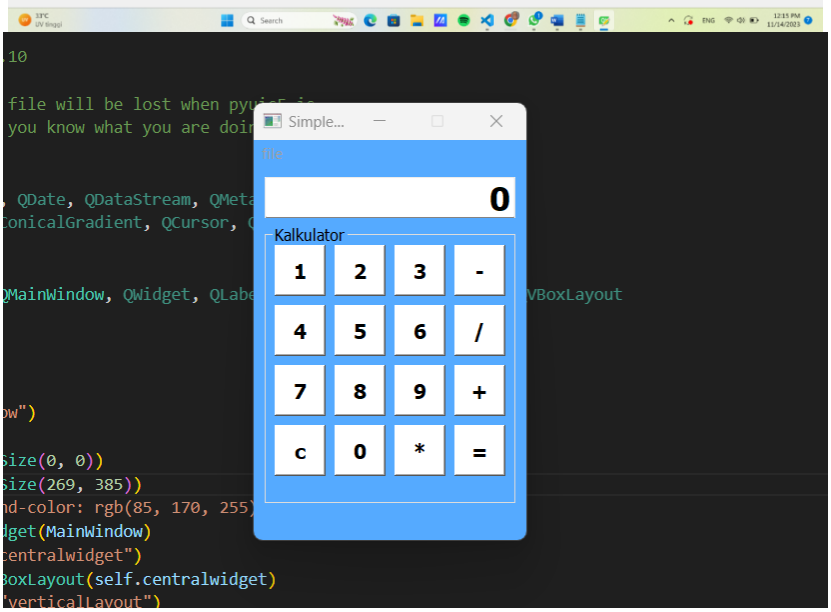
Dalam proyek pembuatan aplikasi penghitungan kalkulator menggunakan PyQt dan Qt Designer, kami berhasil merancang dan mengimplementasikan antarmuka pengguna yang intuitif dan fungsional. Melalui penggunaan Qt Designer, kami dapat dengan mudah menyusun elemen-elemen antarmuka, seperti tombol dan input, sementara PyQt memungkinkan kami untuk menghubungkan elemen-elemen tersebut dengan fungsi-fungsi penghitungan.

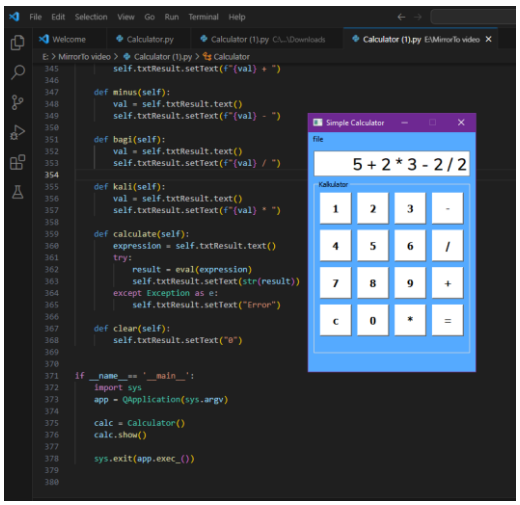
Aplikasi kalkulator yang dihasilkan mampu menangani operasi aritmatika dasar dengan baik. Fungsi-fungsi penghitungan telah diimplementasikan dengan benar, dan kesalahan input dapat ditangani dengan baik untuk meningkatkan pengalaman pengguna. Proses pengujian telah membuktikan bahwa aplikasi ini dapat beroperasi dengan stabil dan memberikan hasil yang sesuai dengan harapan.

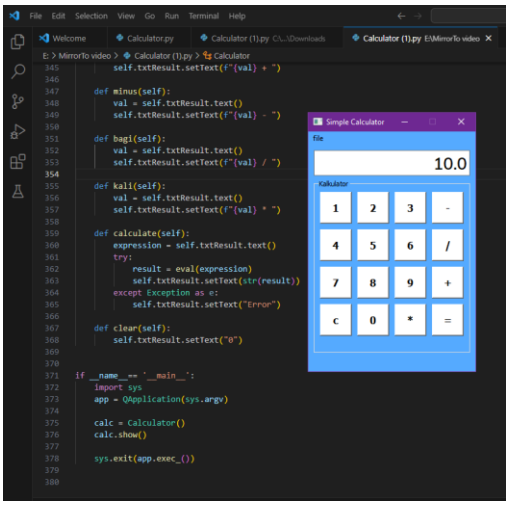
Selama pembuatan aplikasi, kami menghadapi beberapa tantangan, terutama dalam menangani logika penghitungan dan menyusun tata letak antarmuka. Namun, melalui riset dan eksperimen, kami berhasil mengatasi kendala-kendala tersebut dan menghasilkan produk yang memenuhi standar kualitas yang diinginkan.

LAMPIRAN




Simple...
file
0
Kalkulator
1 2 3 -
4 5 6 /
7 8 9 +
c 0 * =


Simple Calculator
5+2*3-2/2
Kalkulator
1 2 3 -
4 5 6 /
7 8 9 +
c 0 * =


Simple Calculator
10.0
Kalkulator
1 2 3 -
4 5 6 /
7 8 9 +
c 0 * =

DAFTAR PUSTAKA

https://www.pythonguis.com/installation/install-qt-designer-standalone/

https://masgalih.medium.com/pemanfaatan-pyqt-untuk-membangun-aplikasi-berbasis-gui-9c742b916998

https://prasetiautamacv.wordpress.com/2016/08/08/gui-menggunakan-python-dan-qtdesigner/