LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Algoritma Dan Pemprograman 2B
Kelas	: 1IA24
Praktikum Ke-	: 7
Tanggal	: Selasa, 23 mei 2022
Materi	: Review Materi
NPM	: 50422818
Nama	: Lutfi Robbani
Ketua asisten	: Gede Oke
Nama asisten	:
Paraf asisten	:
Jumlah Lembar	: 11 Lembar

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2022

LISTENING

```python
from flask import Flask, request
from werkzeug.utils import secure_filename
from flask import send_from_directory
import os

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = os.path.join(
    os.path.dirname(__file__), 'uploads')

if not os.path.exists(app.config['UPLOAD_FOLDER']):
    os.makedirs(app.config['UPLOAD_FOLDER'])


@app.route('/upload', methods=['POST'])
def upload_file():
    file = request.files['file']
    filename = secure_filename(file.filename)
    file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
    return 'File uploaded successfully'


@app.route('/uploads/<filename>', methods=['GET'])
def download_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)


if __name__ == '__main__':
    app.run(debug=True)
```

```python
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget,
QVBoxLayout,QPushButton, QLabel, QFileDialog, QMessageBox
from PyQt5.QtCore import Qt
import requests
import webbrowser


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

        self.setWindowTitle('File Uploader')
        self.setGeometry(100, 100, 300, 150)

        main_widget = QWidget(self)
        self.setCentralWidget(main_widget)

        layout = QVBoxLayout(main_widget)

        self.file_label = QLabel('No file selected', self)
        layout.addWidget(self.file_label)

        select_button = QPushButton('Select File', self)
        select_button.clicked.connect(self.select_file)
        layout.addWidget(select_button)

```

```
        upload_button = QPushButton('Upload', self)
        upload_button.clicked.connect(self.upload_file)
        layout.addWidget(upload_button)

        open_web_button = QPushButton('Open File in Web', self)
        open_web_button.clicked.connect(self.open_web_file)
        layout.addWidget(open_web_button)

        self.uploaded_file_path = None

        self.setStyleSheet('''
            QMainWindow {
                background-color: #f0f0f0;
            }
            QLabel {
                font-size: 14px;
            }
            QPushButton {
                font-size: 14px;
                background-color: #4CAF50;
                color: white;
                padding: 8px 16px;
                border: none;
                border-radius: 4px;
            }
            QPushButton:hover {
                background-color: #45a049;
            }
            QPushButton:pressed {
                background-color: #367d39;
            }
        ''')

    def select_file(self):
        file_dialog = QFileDialog(self)
        file_dialog.setFileMode(QFileDialog.ExistingFile)
        file_dialog.setNameFilter('PDF files (*.pdf)')
        if file_dialog.exec_():
            self.uploaded_file_path = file_dialog.selectedFiles()[0]
            self.file_label.setText(self.uploaded_file_path)

    def upload_file(self):
        if self.uploaded_file_path:
            try:
                files = {'file': open(self.uploaded_file_path, 'rb')}
                response = requests.post(
                    'http://127.0.0.1:5000/upload', files=files)
                if response.status_code == 200:
                    QMessageBox.information(
                        self, 'Success', 'File uploaded successfully')
                    self.file_label.setText('No file selected')
                else:
                    QMessageBox.critical(
                        self, 'Error', 'Failed to upload file')
            except requests.exceptions.RequestException:
                QMessageBox.critical(
```

```
                    self, 'Error', 'Failed to connect to the server')
            else:
                QMessageBox.warning(
                    self, 'Warning', 'Please select a file to upload')

    def open_web_file(self):
        if self.uploaded_file_path:
            file_name = self.uploaded_file_path.split('/')[-1]
            web_url = f"http://127.0.0.1:5000/uploads/{file_name}"
            webbrowser.open(web_url)
        else:
            QMessageBox.warning(self, 'Warning', 'Please upload a file first')


if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = FileUploaderWindow()
    window.show()
    sys.exit(app.exec())
```

LOGIKA

```
1   from flask import Flask, request
2   from werkzeug.utils import secure_filename
3   from flask import send_from_directory
4   import os
```

Pertama tama buat file server terlebih dahulu, lalu import modul Flask dan request dari kelas flask, modul secure_filename dari kelas werkzeug.utils, modul send_from_directory dari kelas flask, import modul os.

```
6   app = Flask(__name__)
7   app.config['UPLOAD_FOLDER'] = os.path.join(
8       os.path.dirname(__file__), 'uploads')
9
```

Selanjutnya buat objek app, yang nilai nya keyword Flask dengan parameter(__name__). Selanjutnya buat program app.config['UPLOAD_FOLDER'] = os.path.join yang berfungsi untuk menentukan folder unggahan file dengan menggabungkan direktori dari file skrip saat ini.

Kemudian masukan perintah os.path.dirname(__file__), ‘uploads’ ynag berfungsi untuk, folder di mana file-file yang diunggah akan disimpan dalam aplikasi Flask.

```
9
10  if not os.path.exists(app.config['UPLOAD_FOLDER']):
11      os.makedirs(app.config['UPLOAD_FOLDER'])
12
```

Selanjutnya buat perintah if not os.path.exists(app.config['UPLOAD_FOLDER']): os.makedirs(app.config['UPLOAD_FOLDER']), yang bertujuan untuk memeriksa keberadaan folder unggahan yang sudah ditentukan dalam konfigurasi app.config['UPLOAD_FOLDER']. Jika folder tersebut belum ada, maka kode akan membuat folder tersebut menggunakan os.makedirs().

```
13
14  @app.route('/upload', methods=['POST'])
15  def upload_file():
16      file = request.files['file']
17      filename = secure_filename(file.filename)
18      file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
19      return 'File uploaded successfully'
20
```

Kemudian buat perintah @app.route('/upload', methods=['POST']) agar decorator yang menentukan bahwa rute ‘/upload’ akan menerima permintaan POST. Selanjutnya Fungsi upload_file() unutuk menangani rute yang akan dieksekusi ketika permintaan POST diterima pada rute /upload.

Setelah itu buat perintah file = request.files['file'] yang digunakan untuk mengakses file yang diunggah dalam permintaan POST. 'file' dalam request.files['file'] adalah nama field dalam permintaan POST yang berisi file yang diunggah.

Kemudian buat perintah filename = secure_filename(file.filename) yang digunakan untuk memperoleh nama file yang aman dengan menghilangkan karakter-karakter yang tidak diinginkan atau berbahaya dari nama file yang diunggah.

Setalah itu buat perintah file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename)) yang digunakan untuk menyimpan file yang diunggah ke folder yang sudah ditentukan dalam konfigurasi

'UPLOAD_FOLDER'. Setelah program berhasil mengapload program akan mencetak file uploaded successfully.

```
@app.route('/uploads/<filename>', methods=['GET'])
def download_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)

```

Selanjutnya buat perintah @app.route('/uploads/<filename>', methods=['GET']) untuk decorator yang menentukan bahwa rute /uploads/<filename> akan menerima permintaan GET, dengan <filename> sebagai parameter yang akan digunakan untuk menentukan nama file yang akan diunduh.

Perintah selanjutnya download_file(filename) berfungsi penangan rute yang dieksekusi saat ada permintaan GET di rute /uploads/<filename>, di mana filename unutk nama file yang akan diunduh.

Fungsi ini menggunakan send_from_directory() untuk mengirim file dari folder yang telah ditentukan dalam app.config['UPLOAD_FOLDER']. Setelah file dikirimkan menggunakan send_from_directory(), file dengan nama yang sesuai akan menjadi respons yang dapat diunduh oleh user.

```

if __name__ == '__main__':
    app.run(debug=True)

```

Selanjutnya buat perintah ketika file Python dijalankan, program akan mengevaluasi kondisi if __name__ == '__main__' sebagai True, sehingga keyword app.run() akan dijalankan dan aplikasi akan dimulai pada alamat dan port yang ditentukan saat debug=True.

```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget,
QVBoxLayout,QPushButton, QLabel, QFileDialog, QMessageBox
from PyQt5.QtCore import Qt
import requests
import webbrowser
```

Selanjutnya import modul sys, qapplicaton, qwisget, qvboxlayout, QPushButton, QLabel, QFileDialog, QMessageBox dari kelas pyqt5.qtwidgets, import modul qt pula dari kelas pyqt5.qtcore, lalu import modul request dan webbrowser.

```
class FileUploaderWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle('File Uploader')
        self.setGeometry(100, 100, 300, 150)
```

Pertama buat kelas FileUploaderWindow yang merupakan turunan dari kelas QMainWindow. Dalam metode __init__(), pemanggilan super().__init__() berguna untuk memanggil konstruktor dari kelas induk (QMainWindow) dan melakukan inisialisasi awal. Judul jendela diatur dengan self.setWindowTitle('File Uploader'), lalu geometri jendela diatur dengan self.setGeometry(100, 100, 300, 150) yang menentukan posisi (x, y) dan ukuran (lebar, tinggi) jendela.

```
20          main_widget = QWidget(self)
21          self.setCentralWidget(main_widget)
22
23          layout = QVBoxLayout(main_widget)
24
25          self.file_label = QLabel('No file selected', self)
26          layout.addWidget(self.file_label)
27
```

Kemudian buat sebuah objek main_widget sebagai instance dari kelas QWidget dengan self sebagai parent. Kemudian, main_widget diatur sebagai central widget untuk objek FileUploaderWindow menggunakan self.setCentralWidget(main_widget). Selanjutnya, sebuah objek layout dibuat sebagai instance dari kelas QVBoxLayout dengan main_widget sebagai parent.

Sebuah objek file_label juga dibuat sebagai instance dari kelas QLabel dengan teks awal 'No file selected' dan self sebagai parent. Terakhir, file_label ditambahkan ke dalam layout menggunakan layout.addWidget(self.file_label).

```
28          select_button = QPushButton('Select File', self)
29          select_button.clicked.connect(self.select_file)
30          layout.addWidget(select_button)
31
32          upload_button = QPushButton('Upload', self)
33          upload_button.clicked.connect(self.upload_file)
34          layout.addWidget(upload_button)
35
```

Kemudian buat select_button dan upload_button. select_button berguna sebagai sebuah tombol dengan teks 'Select File' dan self sebagai parent. Sinyal clicked dari select_button dihubungkan ke method select_file menggunakan select_button.clicked.connect(self.select_file). Selanjutnya, select_button ditambahkan ke dalam layout menggunakan layout.addWidget(select_button).

upload_button juga dibuat sebagai objek tombol dengan teks 'Upload' dan self sebagai parent. Sinyal clicked dari upload_button dihubungkan ke metode upload_file menggunakan upload_button.clicked.connect(self.upload_file). Kemudian, upload_button ditambahkan ke dalam layout menggunakan layout.addWidget(upload_button).

```
35
36          open_web_button = QPushButton('Open File in Web', self)
37          open_web_button.clicked.connect(self.open_web_file)
38          layout.addWidget(open_web_button)
39
40          self.uploaded_file_path = None
```

Setelah itu buat objek open_web_button sebagai instance dari kelas QPushButton dengan teks 'Open File in Web' dan self sebagai parent. Sinyal clicked dari open_web_button dihubungkan ke method open_web_file menggunakan open_web_button.clicked.connect(self.open_web_file). Selanjutnya, open_web_button ditambahkan ke dalam layout menggunakan layout.addWidget(open_web_button).

Dengan begitu sebuah tombol baru (open_web_button) dibuat dengan teks 'Open File in Web' dan dihubungkan dengan method open_web_file. Selanjutnya buat atribut uploaded_file_path dibuat dan diatur

dengan nilai awal None. Atribut ini dapat digunakan untuk menyimpan jalur file yang diunggah dalam objek.

```
        self.setStyleSheet('''
            QMainWindow {
                background-color: #f0f0f0;
            }
            QLabel {
                font-size: 14px;
            }
            QPushButton {
                font-size: 14px;
                background-color: #4CAF50;
                color: white;
                padding: 8px 16px;
                border: none;
                border-radius: 4px;
            }
            QPushButton:hover {
                background-color: #45a049;
            }
            QPushButton:pressed {
                background-color: #367d39;
            }
        ''')
```

Setelah itu panggil method setStyleSheet() pada objek self (FileUploaderWindow) dengan argumen berupa string CSS yang berisi aturan gaya untuk mengatur, Elemen QMainWindow: Mengatur warna latar belakang menjadi #f0f0f0. Elemen QLabel: Mengatur ukuran font menjadi 14px.

Elemen QPushButton: Mengatur ukuran font menjadi 14px, warna latar belakang menjadi #4CAF50, warna teks menjadi putih, padding sebesar 8px pada bagian atas dan bawah serta 16px pada bagian kiri dan kanan, tidak ada border, dan border-radius sebesar 4px.

Ketika tombol QPushButton dihover: Mengatur warna latar belakang menjadi #45a049. Ketika tombol QPushButton ditekan: Mengatur warna latar belakang menjadi #367d39.

```
    def select_file(self):
        file_dialog = QFileDialog(self)
        file_dialog.setFileMode(QFileDialog.ExistingFile)
        file_dialog.setNameFilter('PDF files (*.pdf)')
        if file_dialog.exec_():
            self.uploaded_file_path = file_dialog.selectedFiles()[0]
            self.file_label.setText(self.uploaded_file_path)

```

Selanjutnya buat objek file_dialog sebagai instance dari kelas QFileDialog dengan self sebagai parent. Mengatur mode file dialog menjadi QFileDialog.ExistingFile. Mengatur filter nama file menjadi "PDF files (*.pdf)".

Jika file dialog dieksekusi dan user memilih file, maka jalur file yang dipilih akan disimpan dalam atribut self.uploaded_file_path. Dan Mengubah teks pada label self.file_label dengan jalur file yang dipilih.

Jadi kode di atas, saat tombol "Select File" diklik, jendela dialog file akan muncul. Pengguna dapat memilih file PDF, dan jalur file yang dipilih akan disimpan dalam atribut self.uploaded_file_path. Selain itu, teks pada label self.file_label akan diperbarui dengan jalur file yang dipilih.

```
73      def upload_file(self):
74          if self.uploaded_file_path:
75              try:
76                  files = {'file': open(self.uploaded_file_path, 'rb')}
77                  response = requests.post(
78                      'http://127.0.0.1:5000/upload', files=files)
79                  if response.status_code == 200:
80                      QMessageBox.information(
81                          self, 'Success', 'File uploaded successfully')
82                      self.file_label.setText('No file selected')
83                  else:
84                      QMessageBox.critical(
85                          self, 'Error', 'Failed to upload file')
86              except requests.exceptions.RequestException:
87                  QMessageBox.critical(
88                      self, 'Error', 'Failed to connect to the server')
89          else:
90              QMessageBox.warning(
91                  self, 'Warning', 'Please select a file to upload')
92
```

Selanjutnya periksa apakah self.uploaded_file_path tidak kosong, yaitu apakah user telah memilih file untuk diunggah. Jika self.uploaded_file_path tidak kosong, maka dilakukan langkah-langkah berikut:

Membuka file yang akan diunggah dalam mode baca biner menggunakan open(self.uploaded_file_path, 'rb'). Membuat objek files yang berisi file yang akan diunggah, dengan kunci 'file' dan nilai berupa file yang telah dibuka.

Mengirim permintaan POST ke URL 'http://127.0.0.1:5000/upload' menggunakan requests.post() dan mengirimkan files sebagai data.

Memeriksa status kode respons dari server. Jika status kode adalah 200, berarti file berhasil diunggah. Dalam hal ini, ditampilkan pesan informasi dengan judul "Success" dan pesan "File uploaded successfully" menggunakan QMessageBox.information(). Selain itu, teks pada label self.file_label diatur kembali menjadi "No file selected".

Jika status kode tidak sama dengan 200, berarti terjadi kesalahan dalam mengunggah file. Dalam hal ini, ditampilkan pesan kesalahan dengan judul "Error" dan pesan "Failed to upload file" menggunakan QMessageBox.critical().

Jika terjadi kesalahan dalam koneksi ke server saat mengunggah file, ditampilkan pesan kesalahan dengan judul "Error" dan pesan "Failed to connect to the server" menggunakan QMessageBox.critical().

Jika self.uploaded_file_path kosong, yaitu pengguna belum memilih file untuk diunggah, ditampilkan pesan peringatan dengan judul "Warning" dan pesan "Please select a file to upload" menggunakan QMessageBox.warning().

```
92
93      def open_web_file(self):
94          if self.uploaded_file_path:
95              file_name = self.uploaded_file_path.split('/')[-1]
96              web_url = f"http://127.0.0.1:5000/uploads/{file_name}"
97              webbrowser.open(web_url)
98          else:
99              QMessageBox.warning(self, 'Warning', 'Please upload a file first')
100
```

Selanjutnya periksa self.uploaded_file_path tidak kosong, maka dilakukan langkah-langkah berikut:

Memisahkan nama file dari jalur lengkap self.uploaded_file_path menggunakan metode split('/') dengan argumen '/' sebagai pemisah. Ini akan menghasilkan sebuah daftar (list) yang berisi semua komponen jalur, kemudian program mengambil elemen terakhir dengan menggunakan indeks -1. Hal ini dilakukan untuk mendapatkan nama file saja.

Kemudian Membentuk URL web dengan menggabungkan http://127.0.0.1:5000/uploads/ (alamat basis) dan file_name (nama file) menggunakan f-string. Lalu Membuka URL web menggunakan webbrowser.open(web_url). Ini akan membuka browser web dan mengarahkannya ke URL yang telah dibentuk.

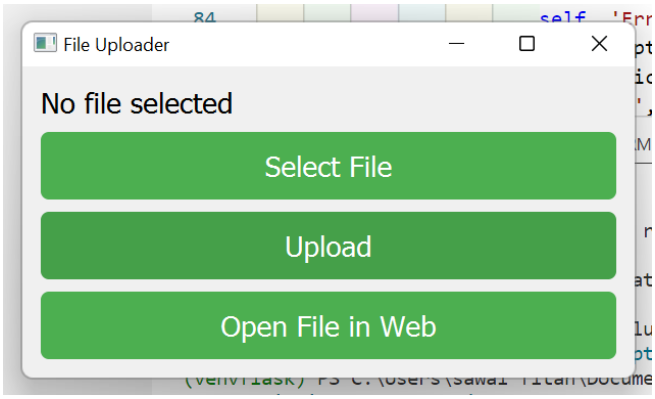
Jika self.uploaded_file_path kosong, yaitu pengguna belum memilih file untuk diunggah, ditampilkan pesan peringatan menggunakan QMessageBox.warning() dengan judul "Warning" dan pesan "Please upload a file first".

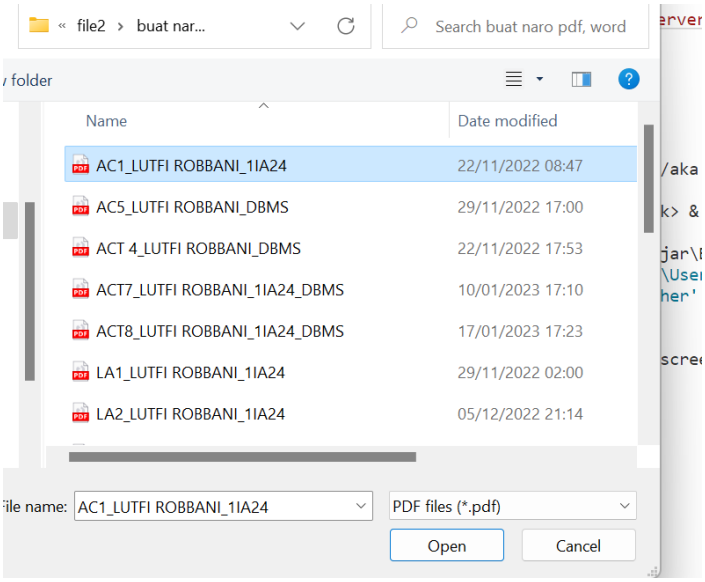
```
101
102 if __name__ == '__main__':
103     app = QApplication(sys.argv)
104     window = FileUploaderWindow()
105     window.show()
106     sys.exit(app.exec())
```

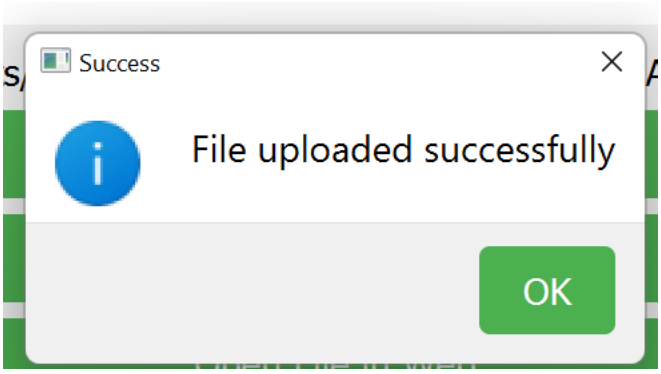
Lalu cek apakah skrip ini dijalankan sebagai skrip utama menggunakan if __name__ == '__main__':. buat instance QApplication dengan app = QApplication(sys.argv). buat instance FileUploaderWindow dengan window = FileUploaderWindow(). Menampilkan jendela dengan window.show(). Memulai eksekusi aplikasi dengan app.exec().Menghentikan eksekusi aplikasi dan keluar dari program dengan sys.exit(app.exec()).

Dengan demikian, kode tersebut akan menjalankan aplikasi File Uploader dengan menggunakan Qt dan menampilkan jendela FileUploaderWindow saat program dijalankan.

OUTPUT


File Uploader
No file selected
Select File
Upload
Open File in Web


file2 › buat nar...
Search buat naro pdf, word
folder
Name
Date modified
AC1_LUTFI ROBBANI_1IA24 22/11/2022 08:47
AC5_LUTFI ROBBANI_DBMS 29/11/2022 17:00
ACT 4_LUTFI ROBBANI_DBMS 22/11/2022 17:53
ACT7_LUTFI ROBBANI_1IA24_DBMS 10/01/2023 17:10
ACT8_LUTFI ROBBANI_1IA24_DBMS 17/01/2023 17:23
LA1_LUTFI ROBBANI_1IA24 29/11/2022 02:00
LA2_LUTFI ROBBANI_1IA24 05/12/2022 21:14
File name: AC1_LUTFI ROBBANI_1IA24
PDF files (*.pdf)
Open
Cancel


Success
File uploaded successfully
OK