**TECHNICAL REPORT**

**PEMROGRAMAN DESKTOP**

**MODUL 5**



**Disusun Oleh :**

TGL. PRAKTIKUM : Jum’at, 11 Desember 2020

NAMA : Achmad Farid Alfa Waid

NRP : 190411100073

KELOMPOK : 2

DOSEN : Moh. Kautsar Sophan, S.Kom., M.MT.

TELAH DISETUJUI TANGGAL :

...........................................

ASISTEN PRAKTIKUM

Nadia Asri

(180411100063)

**LABORATORIUM MULTIMEDIA COMPUTING**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**BAB I**

**TUJUAN DAN DASAR TEORI**

1. **TUJUAN**

Pada praktikum kali ini bertujuan untuk memahami tentang cara membuat database, table, menambahkan data, menampilkan data, dan mencari data pada database.

1. **DASAR TEORI**

Database berfungsi untuk menyimpan dan mengelompokkan suatu data berdasarkan identifikasi data yang sudah dibuat, meminimalisir terjadinya duplikasi data pada saat penyimpanan data, memudahkan pengguna dalam proses (akses, edit, tambah, dan delete)

**BAB II**

**PEMBAHASAN**

1. **SOAL**
2. Buat aplikasi yang terkoneksi ke database
3. Siapkan fungsi untuk
4. Membuat database, membuat tabel
5. Memasukkan data ke dalam tabel dengan perintah SQL
6. Menampilkan isi tabel
7. Mencari data dalam tabel
8. **JAWABAN**
9. **modul 5.py**

import sys

from PyQt5 import \*

from PyQt5.QtWidgets import \*

from PyQt5.Qt import \*

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

from PyQt5 import QtSql

import sqlite3

class Film(QWidget):

#Membuat fungsi init untuk inisialisasi class Mahasiswa

def \_\_init\_\_(self):

#untuk mengembalikan semua atribut dan method yang ada

super().\_\_init\_\_()

#membuka database

self.OpenDatabase()

self.createTabel()

#memanggil fungsi Layout yang sudah dibuat agar ditampilkan hasilnya

self.Layout()

def OpenDatabase(self):

#Mendeklarasikan database

db = QtSql.QSqlDatabase.addDatabase('QSQLITE')

#Membuat nama database

db.setDatabaseName('test.db')

#Mengecek Database Apakah sudah terkoneksi atau belum

if db.open():

print('Berhasil membuka Database')

else:

print('Gagal membuka Database!')

def createTabel(self):

query = QtSql.QSqlQuery()

query.exec\_("create table listfilm(" "id integer primary key AUTOINCREMENT, "

"judul varchar(20), tahun\_rilis int(10), genre varchar(20), review varchar(100), rating varchar(20))")

print ("True")

def Layout(self):

#Membuat Grid Layout

grid = QGridLayout()

#Membuat label, Line Edit, dan Button yang akan dimasukkan ke dalam layout Grid

open\_database = QLabel("Buka Database")

grid.addWidget(open\_database,0,0)

open\_button = QPushButton("Open Database")

grid.addWidget(open\_button,0,1,1,2)

add\_data = QLabel("Tambah Data:")

grid.addWidget(add\_data,1,0,1,0)

add\_id = QLabel("ID:")

grid.addWidget(add\_id,2,0)

self.Id = QLineEdit(self)

grid.addWidget(self.Id,2,1,1,2)

add\_judul = QLabel("Judul:")

grid.addWidget(add\_judul,3,0)

self.Judul = QLineEdit(self)

grid.addWidget(self.Judul,3,1,1,2)

tahun\_rilis = QLabel("Tahun Rilis FIlm:")

grid.addWidget(tahun\_rilis,4,0)

self.Tahun = QLineEdit(self)

grid.addWidget(self.Tahun,4,1,1,2)

add\_genre = QLabel("Genre:")

grid.addWidget(add\_genre,5,0)

self.Genre = QLineEdit(self)

grid.addWidget(self.Genre,5,1,1,2)

add\_review = QLabel("Review:")

grid.addWidget(add\_review,6,0)

self.Review = QLineEdit(self)

grid.addWidget(self.Review,6,1,1,2)

add\_rating = QLabel("Rating:")

grid.addWidget(add\_rating,7,0)

self.Rating = QLineEdit(self)

grid.addWidget(self.Rating,7,1,1,2)

add\_button = QPushButton("Tambah Data")

grid.addWidget(add\_button,8,0,1,0)

search = QLabel("Cari Data:")

grid.addWidget(search,9,0)

self.Cari = QLineEdit(self)

grid.addWidget(self.Cari,9,1)

search\_button = QPushButton("Cari")

grid.addWidget(search\_button,9,2)

#Membuat widget table view yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid

self.tableview = QTableView(self)

self.tableview.setObjectName("Data")

grid.addWidget(self.tableview,10,0,1,0)

#Ketika button di klik akan memanggi fungsi masing - masing

open\_button.clicked.connect(self.tampilData)

add\_button.clicked.connect(self.tambahData)

search\_button.clicked.connect(self.filterData)

#Layout grid di jadikan layout utama

self.setLayout(grid)

#Fungsi untuk menampilankan data di dalam table

def tampilData(self):

#Membuat Model

model = QSqlQueryModel()

#Mendefinisikan sql

sql = "SELECT \* FROM listfilm"

#Mengeksekusi Model Query

model.setQuery(sql)

#Mengeset Data Model Ke table view

self.tableview.setModel(model)

#self.tableview.setWindowTitle(title)

return self.tableview

#Fungsi tambah Data ke dalam tabel listfilm

def tambahData(self):

#Mengambil Text inputan

Id = str(self.Id.text())

Judul = str(self.Judul.text())

Tahun = str(self.Tahun.text())

Genre = str(self.Genre.text())

Review = str(self.Review.text())

Rating = str(self.Rating.text())

#Mendefinisikan Query

query = QtSql.QSqlQuery()

#Menjalankan Perintah Sql

query.prepare("INSERT INTO listfilm VALUES ('" + Id + "','" + Judul + "', '" + Tahun + "', '" + Genre + "', '" + Review + "', '" + Rating + "')")

#Mengecek apakah query berjalan dengan baik

if query.exec\_():

self.Id.setText("")

self.Judul.setText("")

self.Tahun.setText("")

self.Genre.setText("")

self.Review.setText("")

self.Rating.setText("")

#Menampilkan Data

self.tampilData()

else:

#Apabila error akan menampilkan errornya ke dalam terminal

print("Insert Error: ", query.lastError().text())

#Fungsi Filter Data

def filterData(self):

#Membuat Model

model = QSqlQueryModel()

#Mengambil Inputan filter

filter\_search = str(self.Cari.text())

sql = "SELECT \* FROM listfilm WHERE id LIKE '%"+str(filter\_search)+"%' OR judul LIKE '%"+str(filter\_search)+"%' OR tahun\_rilis LIKE '%"+str(filter\_search)+"%' OR genre LIKE '%"+str(filter\_search)+"%'"

self.Cari.setText("")

#Mengeksekusi Model Query

model.setQuery(sql)

#Mengeset Data Model Ke table view

self.tableview.setModel(model)

#mengembalikan nilai table view

return self.tableview

if \_\_name\_\_ == '\_\_main\_\_':

#Inisisalisai pyqt

app = QApplication(sys.argv)

#mengatur style di window menjadi style fusion

app.setStyle("fusion")

#membuat variabel ex yang berisi class Film

ex = Film()

#Menentukan ukuran window dan title untuk menampilkan

ex.setGeometry(100,100,800,600)

#membuat judul window

ex.setWindowTitle("Database Sqlite in Pyqt5")

#menampilan isi dari variabel ex

ex.show()

#membuat system exit

sys.exit(app.exec\_())

1. a) Fungsi membuat database dan tabel

def OpenDatabase(self):

#Mendeklarasikan database

db = QtSql.QSqlDatabase.addDatabase('QSQLITE')

#Membuat nama database

db.setDatabaseName('test.db')

#Mengecek Database Apakah sudah terkoneksi atau belum

if db.open():

print('Berhasil membuka Database')

else:

print('Gagal membuka Database!')

def createTabel(self):

query = QtSql.QSqlQuery()

query.exec\_("create table listfilm(" "id integer primary key AUTOINCREMENT, "

"judul varchar(20), tahun\_rilis int(10), genre varchar(20), review varchar(100), rating varchar(20))")

print ("True")

b) Fungsi memasukkan data ke dalam tabel

#Fungsi tambah Data ke dalam tabel listfilm

def tambahData(self):

#Mengambil Text inputan

Id = str(self.Id.text())

Judul = str(self.Judul.text())

Tahun = str(self.Tahun.text())

Genre = str(self.Genre.text())

Review = str(self.Review.text())

Rating = str(self.Rating.text())

#Mendefinisikan Query

query = QtSql.QSqlQuery()

#Menjalankan Perintah Sql

query.prepare("INSERT INTO listfilm VALUES ('" + Id + "','" + Judul + "', '" + Tahun + "', '" + Genre + "', '" + Review + "', '" + Rating + "')")

#Mengecek apakah query berjalan dengan baik

if query.exec\_():

self.Id.setText("")

self.Judul.setText("")

self.Tahun.setText("")

self.Genre.setText("")

self.Review.setText("")

self.Rating.setText("")

#Menampilkan Data

self.tampilData()

else:

#Apabila error akan menampilkan errornya ke dalam terminal

print("Insert Error: ", query.lastError().text())

c) Fungsi menampilkan isi tabel

#Fungsi untuk menampilankan data di dalam table

def tampilData(self):

#Membuat Model

model = QSqlQueryModel()

#Mendefinisikan sql

sql = "SELECT \* FROM listfilm"

#Mengeksekusi Model Query

model.setQuery(sql)

#Mengeset Data Model Ke table view

self.tableview.setModel(model)

#self.tableview.setWindowTitle(title)

return self.tableview

1. Fungsi mencari data dalam tabel

#Fungsi Filter Data

def filterData(self):

#Membuat Model

model = QSqlQueryModel()

#Mengambil Inputan filter

filter\_search = str(self.Cari.text())

sql = "SELECT \* FROM listfilm WHERE id LIKE '%"+str(filter\_search)+"%' OR judul LIKE '%"+str(filter\_search)+"%' OR tahun\_rilis LIKE '%"+str(filter\_search)+"%' OR genre LIKE '%"+str(filter\_search)+"%'"

self.Cari.setText("")

#Mengeksekusi Model Query

model.setQuery(sql)

#Mengeset Data Model Ke table view

self.tableview.setModel(model)

#mengembalikan nilai table view

return self.tableview

1. **Penjelasan Kode Program**

* *class Film(QWidget):*

*#Membuat fungsi init untuk inisialisasi class Mahasiswa*

*def \_\_init\_\_(self):*

*#untuk mengembalikan semua atribut dan method yang ada*

*super().\_\_init\_\_()*

*#membuka database*

*self.OpenDatabase()*

*self.createTabel()*

*#memanggil fungsi Layout yang sudah dibuat agar ditampilkan hasilnya*

*self.Layout()*

Membuat sebuah class Film dan fungsi init untuk menampilan semua fungsi yang telah dibuat seperti fungsi OpenDatabase(), createTabel(), dan Layout()

* *def OpenDatabase(self):*

*#Mendeklarasikan database*

*db = QtSql.QSqlDatabase.addDatabase('QSQLITE')*

*#Membuat nama database*

*db.setDatabaseName('test.db')*

*#Mengecek Database Apakah sudah terkoneksi atau belum*

*if db.open():*

*print('Berhasil membuka Database')*

*else:*

*print('Gagal membuka Database!')*

Membuat fungsi OpenDatabase yang bersi variable db untuk menambahkan database ke dalam aplikasi QSQLITE, dan memberi nama database dengan syntax setDatabaseName. Dan membuat sebuah kondisi dimana jika database berhasi dibuka . jika gagal.

* *def createTabel(self):*

*query = QtSql.QSqlQuery()*

*query.exec\_("create table listfilm(" "id integer primary key AUTOINCREMENT, "*

*"judul varchar(20), tahun\_rilis int(10), genre varchar(20), review varchar(100), rating varchar(20))")*

*print ("True")*

Membuat fungsi createTabel yang bersi variable query untuk mendeklarasikan syntax QSqlQuery, dan menjalankan proses pembuatan table dengan syntax contoh di atas. Jika berhasil di terminal akan tampil tulisan True.

* *def Layout(self):*

*#Membuat Grid Layout*

*grid = QGridLayout()*

*#Membuat label, Line Edit, dan Button yang akan dimasukkan ke dalam layout Grid*

*open\_database = QLabel("Buka Database")*

*grid.addWidget(open\_database,0,0)*

*open\_button = QPushButton("Open Database")*

*grid.addWidget(open\_button,0,1,1,2)*

*add\_data = QLabel("Tambah Data:")*

*grid.addWidget(add\_data,1,0,1,0)*

*add\_id = QLabel("ID:")*

*grid.addWidget(add\_id,2,0)*

*self.Id = QLineEdit(self)*

*grid.addWidget(self.Id,2,1,1,2)*

*add\_judul = QLabel("Judul:")*

*grid.addWidget(add\_judul,3,0)*

*self.Judul = QLineEdit(self)*

*grid.addWidget(self.Judul,3,1,1,2)*

*tahun\_rilis = QLabel("Tahun Rilis FIlm:")*

*grid.addWidget(tahun\_rilis,4,0)*

*self.Tahun = QLineEdit(self)*

*grid.addWidget(self.Tahun,4,1,1,2)*

*add\_genre = QLabel("Genre:")*

*grid.addWidget(add\_genre,5,0)*

*self.Genre = QLineEdit(self)*

*grid.addWidget(self.Genre,5,1,1,2)*

*add\_review = QLabel("Review:")*

*grid.addWidget(add\_review,6,0)*

*self.Review = QLineEdit(self)*

*grid.addWidget(self.Review,6,1,1,2)*

*add\_rating = QLabel("Rating:")*

*grid.addWidget(add\_rating,7,0)*

*self.Rating = QLineEdit(self)*

*grid.addWidget(self.Rating,7,1,1,2)*

*add\_button = QPushButton("Tambah Data")*

*grid.addWidget(add\_button,8,0,1,0)*

*search = QLabel("Cari Data:")*

*grid.addWidget(search,9,0)*

*self.Cari = QLineEdit(self)*

*grid.addWidget(self.Cari,9,1)*

*search\_button = QPushButton("Cari")*

*grid.addWidget(search\_button,9,2)*

*#Membuat widget table view yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid*

*self.tableview = QTableView(self)*

*self.tableview.setObjectName("Data")*

*grid.addWidget(self.tableview,10,0,1,0)*

Membuat fungsi layout yang di dalamnya berisi beberapa widget. Dan membuat sebuah layout grid yang dimasukkan ke dalam variable grid. Membuat label, Line Edit, dan Button dengan identifikasi variable yang berbeda dan akan dimasukkan ke dalam layout Grid dengan syntax addWidget, dan mengatur posisi masing – masing widget tersebut agar rapi dan tidak bertumpukan. Dan yang terakhir Membuat widget table view yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid.

* *#Ketika button di klik akan memanggi fungsi masing - masing*

*open\_button.clicked.connect(self.tampilData)*

*add\_button.clicked.connect(self.tambahData)*

*search\_button.clicked.connect(self.filterData)*

*#Layout grid di jadikan layout utama*

*self.setLayout(grid)*

Membuat 3 signal dimana ketika masing – masing button di klik, akan memanggil fungsi yang sudah di set. Serta menjadikan layout grid menjadi layout utama pada window.

* #*Fungsi untuk menampilankan data di dalam table*

*def tampilData(self):*

*#Membuat Model*

*model = QSqlQueryModel()*

*#Mendefinisikan sql*

*sql = "SELECT \* FROM listfilm"*

*#Mengeksekusi Model Query*

*model.setQuery(sql)*

*#Mengeset Data Model Ke table view*

*self.tableview.setModel(model)*

*#self.tableview.setWindowTitle(title)*

*return self.tableview*

Membuat fungsitampilData untuk menampilkan data yang ada di dalam database. Membuat variable model yang didalamnya berisi syntax untuk menggunakan query, membuat variable sql dan berisi syntax untuk menampilkan data dala table list film. Setelah itu mengeksekusi model query dan mengembalikan nilai tampilan pada tableview.

* #*Fungsi tambah Data ke dalam tabel listfilm*

*def tambahData(self):*

*#Mengambil Text inputan*

*Id = str(self.Id.text())*

*Judul = str(self.Judul.text())*

*Tahun = str(self.Tahun.text())*

*Genre = str(self.Genre.text())*

*Review = str(self.Review.text())*

*Rating = str(self.Rating.text())*

*#Mendefinisikan Query*

*query = QtSql.QSqlQuery()*

*#Menjalankan Perintah Sql*

*query.prepare("INSERT INTO listfilm VALUES ('" + Id + "','" + Judul + "', '" + Tahun + "', '" + Genre + "', '" + Review + "', '" + Rating + "')")*

*#Mengecek apakah query berjalan dengan baik*

*if query.exec\_():*

*self.Id.setText("")*

*self.Judul.setText("")*

*self.Tahun.setText("")*

*self.Genre.setText("")*

*self.Review.setText("")*

*self.Rating.setText("")*

*#Menampilkan Data*

*self.tampilData()*

*else:*

*#Apabila error akan menampilkan errornya ke dalam terminal*

*print("Insert Error: ", query.lastError().text())*

Membuat fungs tambahData untuk menambahkan data hasil inputan dari line edit kedalam database. Dan membuat suatu kondisi dimana jika query berhasil dieksekusi value dalam line edit akan dikosongkan Kembali kemudian menamilkan data record terbaru kedalam tableview dengan fungsi tampilData. Jika gagal makan akan manampilak letak error ke dalam terminal.

* #*Fungsi Filter Data*

*def filterData(self):*

*#Membuat Model*

*model = QSqlQueryModel()*

*#Mengambil Inputan filter*

*filter\_search = str(self.Cari.text())*

*sql = "SELECT \* FROM listfilm WHERE id LIKE '%"+str(filter\_search)+"%' OR judul LIKE '%"+str(filter\_search)+"%' OR tahun\_rilis LIKE '%"+str(filter\_search)+"%' OR genre LIKE '%"+str(filter\_search)+"%'"*

*self.Cari.setText("")*

*#Mengeksekusi Model Query*

*model.setQuery(sql)*

*#Mengeset Data Model Ke table view*

*self.tableview.setModel(model)*

*#mengembalikan nilai table view*

*return self.tableview*

Membuat fungs filterData untuk mencari data record dengan cari menginputkan value ke dalam line edit, dimana value di dalam line edit akan diterima dan dimasukkan ke dalam variable filter\_search. Membuat variable sql yang berisi syntax query untuk mencari data yang diinputkan berdasarkan value dari filter\_search. Dan kemudian akan ditampilkan ke dalam tableview.

* *if \_\_name\_\_ == '\_\_main\_\_':*

*#Inisisalisai pyqt*

*app = QApplication(sys.argv)*

*#mengatur style di window menjadi style fusion*

*app.setStyle("fusion")*

*#membuat variabel ex yang berisi class Film*

*ex = Film()*

*#Menentukan ukuran window dan title untuk menampilkan*

*ex.setGeometry(100,100,800,600)*

*#membuat judul window*

*ex.setWindowTitle("Database Sqlite in Pyqt5")*

*#menampilan isi dari variabel ex*

*ex.show()*

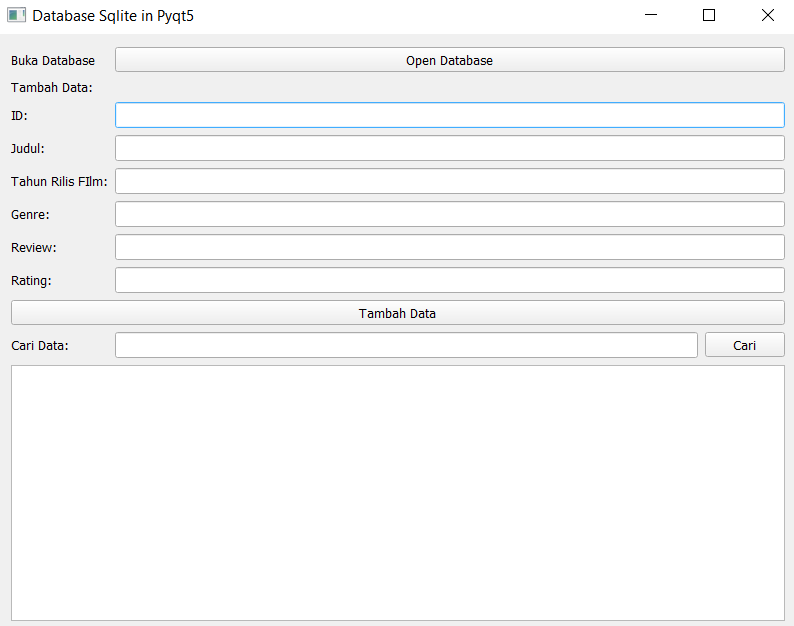
*#membuat system exit*

*sys.exit(app.exec\_())*

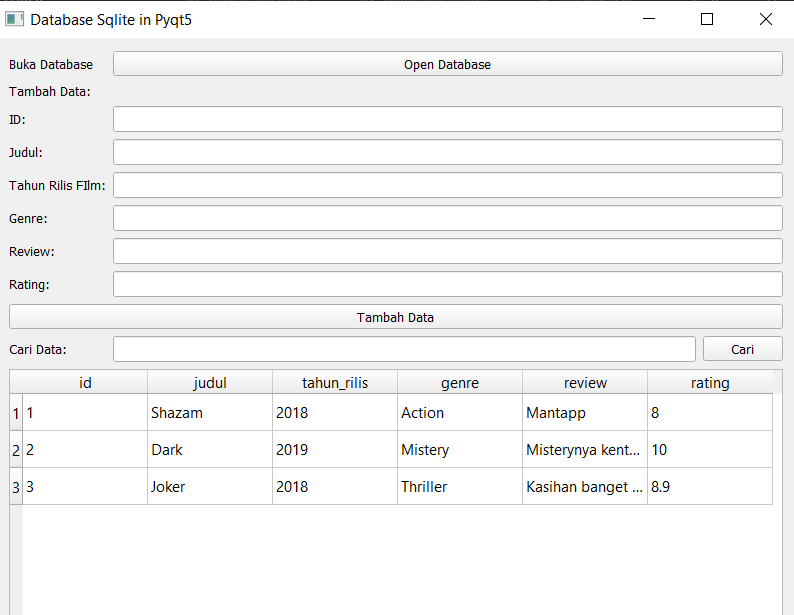
Mendeklarasikan QApplication di dalam variable app, memasukkan value yang ada di class Film ke dalam variable ex, dan memberikan style fusion ke dalam variable app. Lalu mengset ukuran geometry window dan memberikan title pada window dengan syntax setWindowTitle. Kemudian menampilkan variable tersebut dengan fungsi show(), dan membuat system exit.

1. **Hasil Running Program**

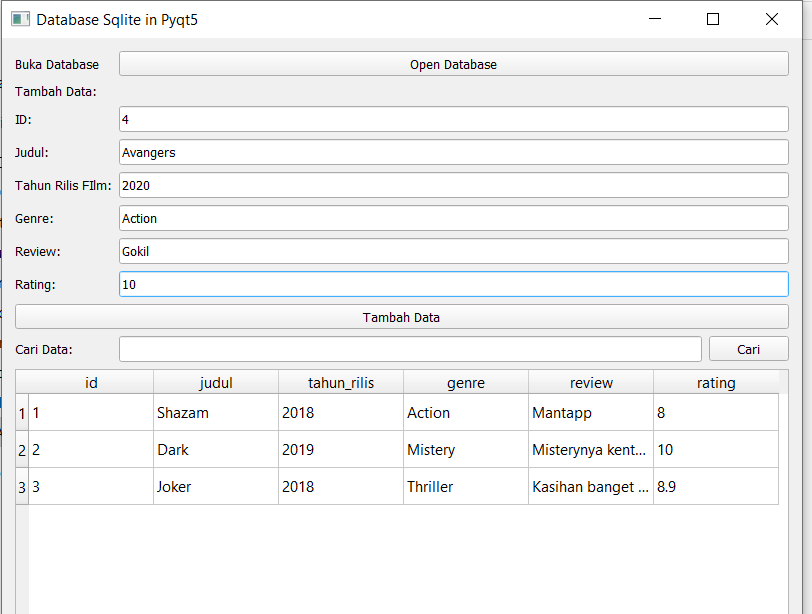
Tampilan awal



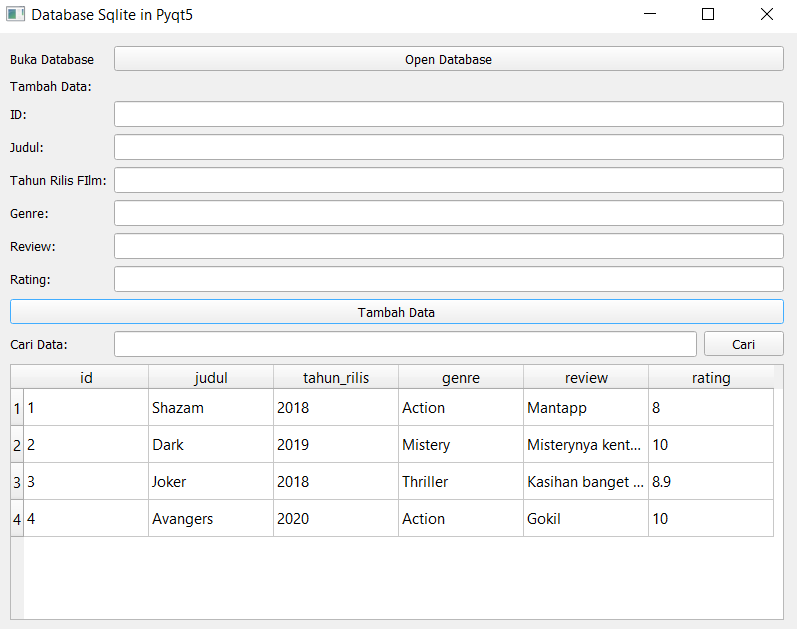
Untuk menampilkan data record pada database, yaitu dengan cari menekan button Open Database, maka pada widget tableview akan menampilkan data yang ada seperti gambar berikut.



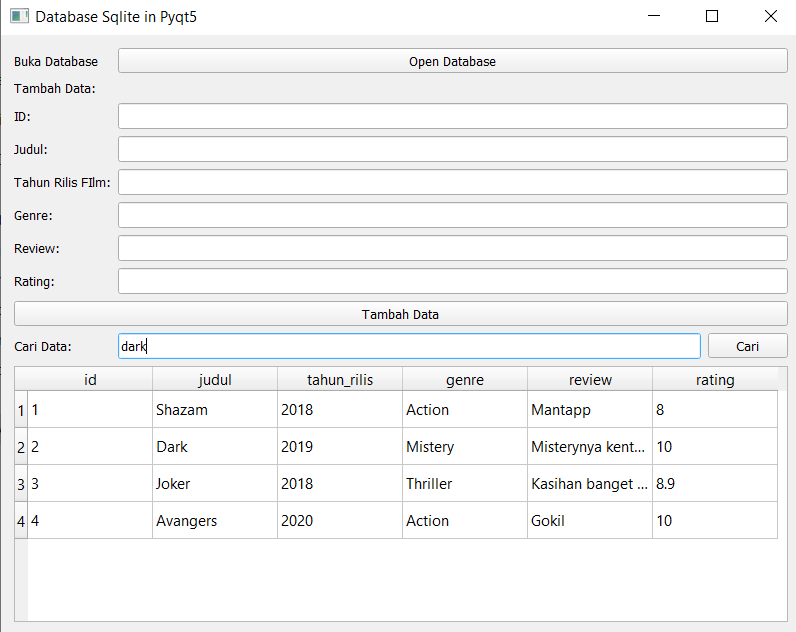
Untuk menambahkan data ke dalam database harus mengisi pada bebrapa line edit yang ada seperti berikut.



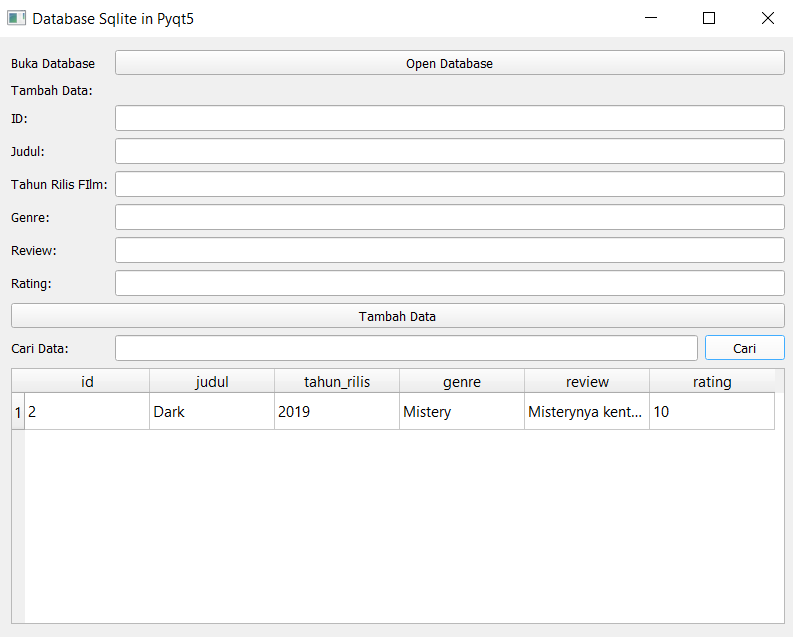
Kemudian menekan button Tambah Data untuk menambhkan data yang diinputkan di line edit ke dalam database. Dan akan menampilkan data baru seperti gambar berikut.



Untuk mencari data, kita haru menginputkan value yang akan dicari ke dalam line edit yang ada terlebih dahulu seperti gambar berikut.



Kemudian kita harus menekan button Cari agar value yang yang kita inputkan di line edit dapat diproses, dan akan menampilkan data seperti gambar berikut.



**BAB II**

**PENUTUP**

1. **Kesimpulan**
2. Kita bisa mendapatkan value dari inputan user menggunakan line edit.
3. Dengan adanya signal dan slot dapat membuat aplikasi yang di buat berfungsi lebih baik.
4. Database sangat membantu user untuk menyimpan beberapa data, dan data tersbut bisa diakses kapan saja
5. **Saran**

Banyak mencoba dan mengekplorasi widget yang lain agar lebih paham