**LAPORAN TUGAS BESAR**

**PROYEK STRUKTUR DATA DAN ALGORITMA**



**DISUSUN OLEH :**

1. Bavio Robio R (G1A023002)

2. Azzahra Faranisa (G1A023010)

3. Carissa Nabilah Putri R (G1A023026)

4. M Tariq Pratama Buhar (G1A023028)

5. Fidia Dewi Wulandari BB (G1A023040)

**Nama Asisten Dosen :**

1. Davi Sulaiman

(G1A022001)

(G1A022002)

(G1A022011)

(G1A022021)

(G1A022024)

(G1A022027)

(G1A022040)

(G1A022060)

(G1A022073)

(G1A022086)

1. Attiya Dianti Fadli
2. Abdi Agung Kurniawan
3. Sophina Shafa Salsabila
4. Evelyn Eunike Aritonang
5. Diodo Arrahman
6. Sinta Ezra Wati Gulo
7. Wahyu Ozorah Manurung
8. Alif Nurhidayat
9. Ahmad Radesta

**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**

**2024**

# **KATA PENGANTAR**

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan Laporan tugas besar pada mata kuliah Praktikum Struktur Data dan Algoritma guna memenuhi tugas mata kuliah tersebut. Mata kuliah ini merupakan salah satu mata kuliah yang wajib ditempuh di Program Studi Teknik Informatika Universitas Dayanu Ikhsanuddin.

Kami mengucapkan terima kasih kepada asisten dosen mata kuliah Praktikum Struktur Data dan Algoritma yang telah banyak memberikan bimbingan dan pengarahan sehingga kami dapat menyelesaikan laporan ini. Ucapan terima kasih yang sama juga kami ucapkan kepada semua pihak yang telah membantu dalam menyelesaikan laporan ini.

Tujuan laporan ini adalah memberikan informasi kepada pembaca mengenai Pembuatan Sistem Informasi Playbook menggunakan Linked List serta memberikan bahasan mengenai sistem yang kami buat. Kami menyadari bahwa dalam laporan ini masih terdapat banyak kekurangan, sehinga kami juga mengharapkan adanya komentar dan masukkan dari pembaca maupun pemberi nilai. Meskipun demikian, mudah-mudahan laporan ini dapat bermanfaat.

Bengkulu, 9 Mei 2024

Penyusun

**DAFTAR ISI**

[KATA PENGANTAR 1](#_Toc166184514)

[DAFTAR GAMBAR 3](#_Toc166184515)

[BAB I 4](#_Toc166184516)

[PENDAHULUAN 4](#_Toc166184517)

[Latar Belakang 4](#_Toc166184518)

[Rumusan Masalah 5](#_Toc166184519)

[Tujuan dan Manfaat 6](#_Toc166184520)

[BAB II 7](#_Toc166184521)

[LANDASAN TEORI 7](#_Toc166184522)

[C++ 7](#_Toc166184523)

[Linked List 7](#_Toc166184524)

[Sistem Informasi Menggunakan Linked List 8](#_Toc166184525)

[Qt Creator 9](#_Toc166184526)

[BAB III 10](#_Toc166184527)

[PEMBAHASAN 10](#_Toc166184528)

[Persiapan Pembuatan Project 10](#_Toc166184529)

[Pembuatan GUI 11](#_Toc166184530)

[Proses Koding 14](#_Toc166184531)

[Output 47](#_Toc166184532)

[BAB IV 50](#_Toc166184533)

[KESIMPULAN DAN SARAN 50](#_Toc166184534)

[Kesimpulan 50](#_Toc166184535)

[Saran 50](#_Toc166184536)

[DAFTAR PUSTAKA 51](#_Toc166184537)

# DAFTAR GAMBAR

# **BAB I**

# **PENDAHULUAN**

## **Latar Belakang**

Sistem Informasi (SI) merupakan fondasi utama dalam menjembatani aktivitas manusia dengan pemanfaatan teknologi. John F. Nash menjelaskan bahwa pengertian sistem informasi adalah kombinasi dari manusia, fasilitas ataupun teknologi, media, prosedur serta pengendalian yang memiliki maksud untuk menata jaringan komunikasi yang penting, proses maupun transaksi tertentu secara rutin, membantu manajemen serta pemakai intern maupun ekstern dan menyediakan dasar dari pengambilan keputusan yang tepat.

Tujuan utama dari SI adalah memastikan aksesibilitas yang baik bagi pengguna, mencapai tujuan organisasi, perusahaan, atau entitas secara cepat dengan didukung oleh data yang terpercaya, mengontrol aliran informasi dan menerima umpan balik, serta meningkatkan kapasitas untuk mengelola SI dengan penilaian yang kritis dan logis terhadap kualitas dan pengalaman.

SI bukan hanya menjadi alat untuk mengelola informasi, tetapi juga menjadi salah satu aset yang paling penting dalam mengoptimalkan kinerja organisasi modern. Dalam era di mana informasi menjadi salah satu komoditas terbesar, penggunaan SI yang efektif dan efisien dapat memberikan keunggulan kompetitif yang signifikan bagi suatu entitas.

Linked List adalah salah satu struktur data linier yang penting dalam dunia pemrograman komputer. Konsep dasar dari Linked List adalah kumpulan node yang saling terhubung, di mana setiap node menyimpan dua item: nilai data dan pointer ke simpul berikutnya.

Dalam penggunaannya, Linked List memungkinkan traversal yang efisien dari node ke node lainnya. Hal ini dikarenakan setiap node memiliki referensi ke node berikutnya dalam urutan. Kelebihan utama Linked List terletak pada kemampuannya untuk menyediakan struktur data dinamis yang memudahkan implementasi stack, queue, dan operasi penyisipan serta penghapusan data.

Meskipun memiliki sejumlah keunggulan, Linked List juga memiliki beberapa kelemahan yang perlu dipertimbangkan. Pertama, penggunaan memori yang lebih besar dibandingkan dengan struktur data lainnya, seperti array. Karena setiap node memerlukan alokasi memori tersendiri. Kedua, traversal dalam Linked List cenderung lebih lambat dibandingkan dengan akses langsung ke elemen dalam array, karena perlu melalui satu per satu dari node awal hingga node yang dituju. Terakhir, Linked List tidak mendukung akses acak, yang berarti untuk mengakses elemen di indeks tertentu, harus dilakukan traversal dari awal hingga elemen yang diinginkan. Dalam konteks pengembangan sistem informasi playbook, penggunaan Linked List dapat menjadi salah satu pilihan yang relevan dalam membangun struktur data untuk manajemen buku. Berikut adalah pembahasan mengenai penerapan Linked List dalam latar belakang sistem informasi playbook:

Kelebihan utama penggunaan Linked List dalam sistem informasi playbook adalah kemampuannya untuk menyediakan struktur data dinamis. Hal ini memungkinkan penambahan dan penghapusan buku dengan efisien, tanpa perlu mengalami pembatasan ukuran yang mungkin terjadi pada struktur data statis seperti array.

Selain itu, Linked List juga mendukung operasi pencarian dan traversal dari satu buku ke buku lainnya dengan cara yang efisien. Meskipun traversal dalam Linked List mungkin cenderung lebih lambat daripada akses langsung ke elemen dalam array, namun hal ini tidak menjadi masalah besar dalam konteks manajemen buku, karena biasanya pencarian dilakukan berdasarkan kriteria tertentu seperti judul atau pengarang.

## **Rumusan Masalah**

1. Bagaimana cara membangun Sistem Informasi Playbook dengan Bahasa C++ menggunakan penyimpanan data Linked List?

## **Tujuan dan Manfaat**

1. Membangun sistem informasi yang efisien dan efektif dalam manajemen buku, yang memanfaatkan kelebihan struktur data Linked List dalam menyediakan fleksibilitas operasional.
2. Mengimplementasikan konsep dasar Linked List dalam pemrograman, sehingga dapat memahami dan menguasai konsep struktur data tersebut.
3. Menyediakan platform yang memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete) terhadap data buku secara mudah dan efisien.
4. Memperluas pemahaman tentang pemrograman C++ melalui proyek praktis dalam pengembangan sistem informasi.

# **BAB II**

# **LANDASAN TEORI**

## **C++**

C++ adalah bahasa pemrograman komputer yang telah menjadi salah satu bahasa paling populer di dunia. Diciptakan oleh Bjarne Stroustrup, C++ adalah pengembangan dari bahasa C yang dikembangkan di Bell Labs. Keunggulan utama dari C++ adalah fleksibilitas dan kemampuannya dalam pengembangan perangkat lunak yang efisien dan cepat.

Salah satu alasan utama popularitas C++ adalah kemampuannya dalam berbagai aplikasi, termasuk pengembangan permainan, virtual reality, simulasi waktu nyata, dan trading frekuensi tinggi. Dalam konteks ini, efisiensi dan kecepatan menjadi kunci, dan C++ menawarkan kinerja yang sangat baik untuk aplikasi-aplikasi ini.

C++ juga dikenal sebagai bahasa pemrograman berorientasi objek. Konsep ini memungkinkan pengembang untuk mengorganisasi kode mereka dalam bentuk objek yang terkait satu sama lain, memudahkan dalam pengembangan dan pemeliharaan kode yang kompleks. Selain itu, paradigma berorientasi objek memberikan fleksibilitas yang lebih besar dalam desain dan pengembangan perangkat lunak, sehingga memungkinkan untuk pengembangan yang lebih modular dan dapat diperluas.

Selain itu, C++ memiliki kemampuan yang sangat dekat dengan perangkat keras, sehingga sangat efektif dalam pengembangan aplikasi yang memerlukan performa tinggi. Ini membuatnya menjadi pilihan yang populer untuk pengembangan sistem operasi, perangkat lunak embedded, dan aplikasi yang memanfaatkan sumber daya perangkat keras secara langsung.

## **Linked List**

Dalam bahasa pemrograman C++, Linked List (LL) adalah salah satu struktur data yang penting dan sering digunakan. Linked List adalah kumpulan elemen yang terhubung secara linear menggunakan pointer. Setiap elemen dalam Linked List disebut sebagai "node" atau "simpul." Setiap node memiliki dua bagian utama: data yang menyimpan informasi yang ingin disimpan (seperti nilai buku dalam sistem informasi playbook) dan pointer yang menunjuk ke node berikutnya dalam urutan. Salah satu keunggulan utama Linked List adalah kemampuannya dalam mengelola memori secara efisien. Struktur data ini memungkinkan alokasi memori yang dinamis, artinya memori dialokasikan saat diperlukan dan direalisasikan saat elemen baru ditambahkan ke Linked List. Ini berbeda dengan array, di mana memori harus dialokasikan sebelumnya dengan ukuran yang sudah ditentukan.

Selain itu, Linked List juga memungkinkan operasi tambah, kurangi, atau menghapus elemen secara dinamis. Misalnya, untuk menambahkan buku baru ke dalam sistem informasi playbook, hanya perlu membuat node baru dan mengatur pointer dengan benar. Begitu juga, untuk menghapus atau memperbarui informasi buku, cukup mengarahkan pointer sehingga node tertentu diabaikan atau informasinya diperbarui.

## **Sistem Informasi Menggunakan Linked List**

Dalam pembuatan sistem informasi sederhana, penggunaan Linked List sebagai struktur data penyimpanan dapat memberikan sejumlah fungsi yang penting. Pertama, Linked List memungkinkan fleksibilitas dalam penyimpanan data, memungkinkan sistem untuk menangani entitas yang beragam tanpa membatasi ukuran atau struktur data yang telah ditetapkan sebelumnya.

Dalam konteks sistem informasi, ini berarti pengelolaan data yang dinamis, di mana entitas seperti buku atau objek lainnya dapat ditambahkan, dihapus, atau diubah tanpa kesulitan. Selain itu, operasi penyisipan dan penghapusan data dalam Linked List relatif efisien, memungkinkan sistem untuk merespons perubahan data dengan cepat dan responsif.

Fungsi traversal yang efisien dari Linked List juga memberikan kemampuan untuk mencari dan mengakses data dengan cepat, sehingga memungkinkan pengguna untuk dengan mudah menemukan informasi yang mereka butuhkan.

## **Qt Creator**

Qt Creator adalah sebuah Integrated Development Environment (IDE) yang dikembangkan oleh The Qt Company untuk memfasilitasi pengembangan aplikasi berbasis Qt. Qt Creator dirancang khusus untuk mempercepat dan menyederhanakan proses pengembangan perangkat lunak, terutama aplikasi yang menggunakan Qt Framework.

Salah satu fitur utama dari Qt Creator adalah editor kode yang kuat dan responsif. Editor ini dilengkapi dengan fitur-fitur seperti penyorotan sintaks, autocompletion, dan integrasi dengan sistem manajemen versi, yang memungkinkan pengembang untuk menulis dan mengedit kode dengan mudah dan efisien.

Selain itu, Qt Creator juga dilengkapi dengan berbagai alat bantu pengembangan seperti debugger, profiler, dan analisis kode, yang memungkinkan pengembang untuk melakukan debugging, pengujian, dan pengoptimalkan aplikasi mereka dengan lebih baik.

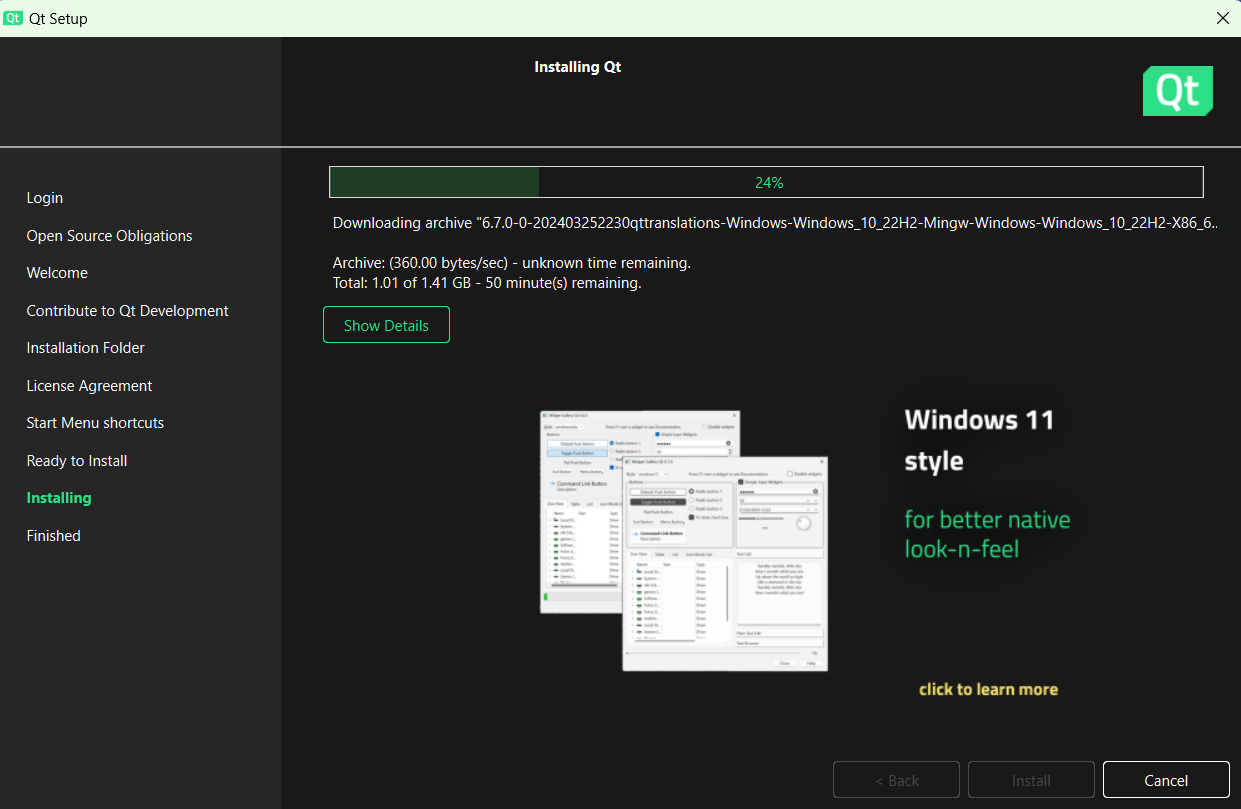
Manajemen proyek yang efisien juga merupakan salah satu keunggulan Qt Creator. IDE ini menyediakan alat untuk membuat, mengelola, dan mengatur proyek-proyek Qt dengan baik. Pengembang dapat dengan mudah menambahkan file-file sumber, mengonfigurasi pengaturan proyek, dan mengatur dependensi dengan cepat dan mudah.

Qt Creator juga mendukung berbagai bahasa pemrograman, termasuk C++, Java, dan Python, serta berbagai teknologi Qt seperti Qt Quick, Qt QML, Hal ini memungkinkan pengembang untuk membuat aplikasi yang berjalan di berbagai platform, termasuk desktop (Windows, macOS, Linux) dan perangkat mobile. Dengan semua fitur dan alat bantu yang disediakannya, Qt Creator memungkinkan pengembang untuk mengembangkan aplikasi Qt dengan cepat, efisien, dan dengan kualitas yang tinggi.

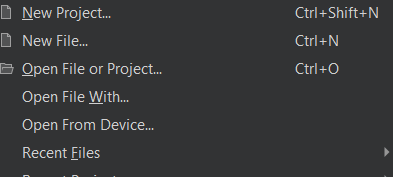
# **BAB III**

# **PEMBAHASAN**

## **Persiapan Pembuatan Project**

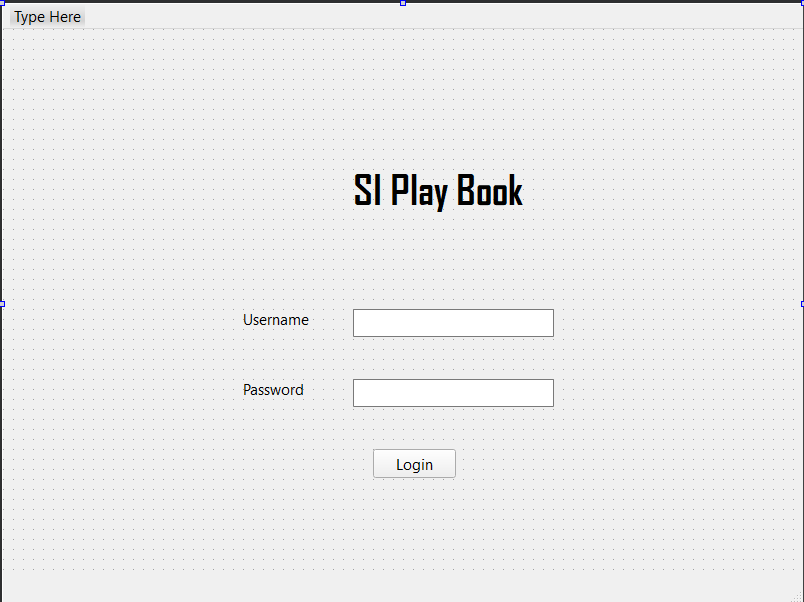
1. Langkah pertama dalam membangun sistem informasi playbook menggunakan Qt yaitu persiapan. Pada tahap ini, dilakukan penginstalan software Qt Creator. Penyusun menggunakan Qt setup online installer untuk menginstall Qt Creator for desktop development.

Gambar 1. Proses Install Qt Creator

1. Berikutnya, setelah semua komponen Qt Creator sudah terinstall, penyusun melakukan create project untuk membuat project baru bernama SI Playbook

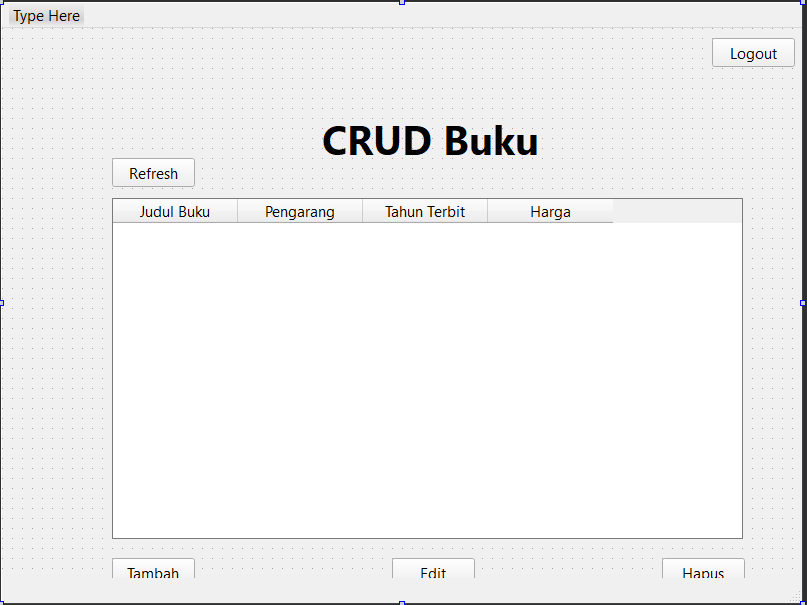
Gambar 2. Create Project

## **Pembuatan GUI**

1. Setelah project berhasil dibuat, langkah pertama yang dilakukan yaitu menentukan halaman atau GUI apa saja yang akan dibangun. Dalam SI playbook ini, penyusun menentukan bahwa akan ada 3 halaman untuk SI ini yaitu halaman mainwindow (login), Dashboard admin, dan Dashboard User.
2. Tampilan pertama yang dibuat yaitu mainwindow (login). Pada tampilan ini, dibuat sebuah label berjudul SI Playboook untuk memberikan judul aplikasi. Halaman ini juga diisi dengan label dan textfield untuk menginput username dan password. Tak lupa pula tombol login untuk melakukan validasi user.

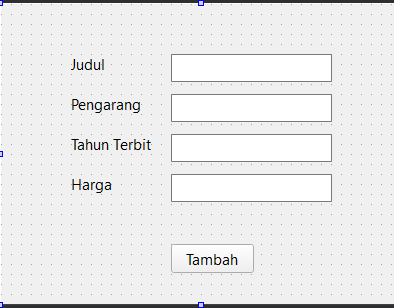
Gambar 3. GUI Login

1. Tampilan berikutnya yang dibuat yaitu admindashboard. Tampilan ini dibuat sebagai tampilan utama bagi admin dimana dia nantinya dapat melakukan CRUD data buku pada sistem. Komponen – komponen pada tampilan ini yaitu Label untuk judul, tablewidget untuk menampilkan data, dan beberapa button untuk melakukan fungsi Refresh, Tambah, Edit, Hapus, dan Logout.



Gambar 4. GUI Dashboard Admin

1. Tampilan berikutnya yaitu tampilan dialog dari tambah buku. Tampilan ini berfungsi sebagai window yang menyediakan textfield untuk input dari admin untuk melakukan proses penambahan buku.



Gambar 5. GUI Tambah Buku

1. Tampilan berikutnya yaitu tampilan dialog edit buku. Tampilan ini berfungsi sebagai window dialog yang menyediakan textfield untuk melakukan proses pengeditan data.



Gambar 6. GUI Edit Buku

1. Tampilan berikutnya yang dibuat yaitu userdashboard. Tampilan ini dibuat sebagai tampilan utama bagi user dimana dia nantinya dapat melihat data buku, melihat detail buku, dan melakukan pencarian buku terdapat label, tablewidget, textfield untuk pencarian, dan tombol lihat detail serta tombol logout.

Gambar 7. GUI Dashboard User

## **Proses Koding**

1. Proses selanjutnya setelah pembuatan GUI adalah proses koding. Karena class pertama yang diakses oleh user adalah mainwindow, maka proses koding pertama kali dilakukan di bagian mainwindow.h dan mainwindow.cpp.

Source Code mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "admindashboard.h"

#include "userdashboard.h"

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindow;

}

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_btn\_login\_clicked();

private:

Ui::MainWindow \*ui;

struct User {

QString username;

QString password;

};

User users[4];

admindashboard \*adminDashboard = nullptr;

UserDashboard \*userDashboard = nullptr;

};

#endif

Penjelasan Kode:

class MainWindow : public QmainWindow. Ini mendeklarasikan kelas MainWindow yang merupakan turunan dari QMainWindow. Ini adalah jendela utama dalam aplikasi.

private slots:

void on\_btn\_login\_clicked();

Ini mendeklarasikan slot private on\_btn\_login\_clicked(), yang akan dipanggil ketika tombol login (btn\_login) ditekan.

private:

Ui::MainWindow \*ui;

struct User {

QString username;

QString password;

};

User users[4];

admindashboard \*adminDashboard = nullptr;

UserDashboard \*userDashboard = nullptr;

* Ui::MainWindow \*ui;: Pointer ke objek UI yang dibuat oleh Qt Designer.
* struct User: Struktur data yang menyimpan username dan password.
* User users[4];: Array yang menyimpan data pengguna.
* admindashboard \*adminDashboard = nullptr;: Pointer ke objek dashboard admin.
* UserDashboard \*userDashboard = nullptr;: Pointer ke objek dashboard pengguna.

Source Code mainwindow.cpp:

#include "mainwindow.h"

#include "./ui\_mainwindow.h"

#include "admindashboard.h"

#include "userdashboard.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

users[0].username = "user";

users[0].password = "user";

users[1].username = "user1";

users[1].password = "user1";

users[2].username = "user2";

users[2].password = "user2";

users[3].username = "admin";

users[3].password = "admin";

connect(ui->btn\_login, &QPushButton::clicked, this, &MainWindow::on\_btn\_login\_clicked);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_btn\_login\_clicked()

{

QString username = ui->tf\_username->text();

QString password = ui->tf\_password->text();

bool loginSuccess = false;

for (const User &user : users) {

if (user.username == username && user.password == password) {

if (username == "admin") {

this->close();

qDebug() << "Login sebagai admin berhasil.";

if (!adminDashboard) {

adminDashboard = new admindashboard(this);

}

adminDashboard->show();

} else {

this->close();

qDebug() << "Login sebagai user berhasil.";

if (!userDashboard) {

userDashboard = new UserDashboard(this);

}

userDashboard->show();

}

loginSuccess = true;

break;

}

}

if (!loginSuccess) {

qDebug() << "Login gagal. Username atau password tidak valid.";

}

}

Penjelasan Kode:

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

Ini adalah konstruktor MainWindow yang menginisialisasi UI dan data pengguna.

users[0].username = "user";

users[0].password = "user";

Data pengguna diinisialisasi di sini dengan nama pengguna dan kata sandi yang sudah ditetapkan.

connect(ui->btn\_login, &QPushButton::clicked, this, &MainWindow::on\_btn\_login\_clicked);

Tombol login (btn\_login) dihubungkan dengan slot on\_btn\_login\_clicked() agar dipanggil saat tombol ditekan.

MainWindow::~MainWindow()

{

delete ui;

}

Destruktor untuk menghapus objek antarmuka pengguna (ui) saat MainWindow dihancurkan.

void MainWindow::on\_btn\_login\_clicked()

Ini adalah slot yang dipanggil ketika tombol login ditekan. Ini memproses informasi masukan pengguna untuk melakukan otentikasi.

1. Setelah membuat kode untuk fungsi login, selanjutnya adalah pembubatan kode untuk objek Buku menggunakan linked list. Disini class yang digunakan untuk mengatur buku yaitu bukumanager.h dan bukumanager.cpp  
     
   Source Code bukumanager.h:

#ifndef BUKUMANAGER\_H

#define BUKUMANAGER\_H

#include <QObject>

#include <QTableWidget>

#include <QDebug>

struct Buku {

QString judul;

QString pengarang;

QString tahunTerbit;

QString harga;

Buku\* next;

};

class BukuManager : public QObject

{

Q\_OBJECT

public:

explicit BukuManager(QObject \*parent = nullptr);

~BukuManager();

void tambahBuku(const QString& judul, const QString& pengarang, const QString& tahunTerbit, const QString& harga);

void tampilkanDataBuku(QTableWidget\* tableWidget);

void tampilkanDataBuku();

void hapusBuku(const QString& judul);

void tampilkanDataBukuUser(QTableWidget\* tableWidget);

void tambahBeberapaBuku();

void perbaruiBuku(const QString& judulLama, const QString& judulBaru, const QString& pengarangBaru, const QString& tahunTerbitBaru, const QString& hargaBaru);

void pencarianBukuUser(const QString& kataKunci, QTableWidget\* tableWidget);

private:

Buku\* head;

};

#endif // BUKUMANAGER\_H

Penjelasan Kode:

struct Buku {

QString judul;

QString pengarang;

QString tahunTerbit;

QString harga;

Buku\* next;

};

Ini adalah struktur data untuk menyimpan informasi tentang sebuah buku, termasuk judul, pengarang, tahun terbit, harga, dan pointer ke buku berikutnya dalam daftar.

class BukuManager : public QObject

{

Kelas BukuManager adalah kelas yang bertanggung jawab untuk manajemen data buku. Metode-metode yang dideklarasikan di sini mencakup penambahan buku, tampilan data buku, penghapusan buku, dan lainnya.

explicit BukuManager(QObject \*parent = nullptr);

~BukuManager();

Konstruktor digunakan untuk inisialisasi, sedangkan destruktor untuk membersihkan sumber daya yang dialokasikan.

* tambahBuku: Menambahkan buku baru ke dalam sistem.
* tampilkanDataBuku: Menampilkan data buku ke dalam QTableWidget.
* hapusBuku: Menghapus buku dari sistem berdasarkan judulnya.
* tampilkanDataBukuUser: Menampilkan data buku untuk pengguna.

Source Code bukumanager.cpp:

#include "bukumanager.h"

BukuManager::BukuManager(QObject \*parent) : QObject(parent), head(nullptr) {

tambahBeberapaBuku();

}

BukuManager::~BukuManager()

{

Buku\* current = head;

while (current) {

Buku\* next = current->next;

delete current;

current = next;

}

}

void BukuManager::tambahBeberapaBuku() {

tambahBuku("Harry Potter", "J.K. Rowling", "1997", "100000");

tambahBuku("To Kill a Mockingbird", "Harper Lee", "1960", "95000");

tambahBuku("The Great Gatsby", "F. Scott Fitzgerald", "1925", "90000");

tambahBuku("1984", "George Orwell", "1949", "85000");

tambahBuku("The Catcher in the Rye", "J.D. Salinger", "1951", "80000");

}

void BukuManager::tambahBuku(const QString& judul, const QString& pengarang, const QString& tahunTerbit, const QString& harga)

{

Buku\* newBuku = new Buku;

newBuku->judul = judul;

newBuku->pengarang = pengarang;

newBuku->tahunTerbit = tahunTerbit;

newBuku->harga = harga;

newBuku->next = nullptr;

if (!head) {

head = newBuku;

} else {

Buku\* current = head;

while (current->next) {

current = current->next;

}

current->next = newBuku;

}

}

void BukuManager::tampilkanDataBuku(QTableWidget\* tableWidget)

{

tableWidget->clearContents();

tableWidget->setRowCount(0);

Buku\* current = head;

int row = 0;

while (current) {

tableWidget->insertRow(row);

tableWidget->setItem(row, 0, new QTableWidgetItem(current->judul));

tableWidget->setItem(row, 1, new QTableWidgetItem(current->pengarang));

tableWidget->setItem(row, 2, new QTableWidgetItem(current->tahunTerbit));

tableWidget->setItem(row, 3, new QTableWidgetItem(current->harga));

current = current->next;

++row;

}

}

void BukuManager::tampilkanDataBuku()

{

Buku\* current = head;

while (current) {

qDebug() << "Judul:" << current->judul << ", Pengarang:" << current->pengarang << ", Tahun Terbit:" << current->tahunTerbit << ", Harga:" << current->harga;

current = current->next;

}

}

void BukuManager::perbaruiBuku(const QString& judulLama, const QString& judulBaru, const QString& pengarangBaru, const QString& tahunTerbitBaru, const QString& hargaBaru)

{

Buku\* current = head;

while (current) {

if (current->judul == judulLama) {

current->judul = judulBaru;

current->pengarang = pengarangBaru;

current->tahunTerbit = tahunTerbitBaru;

current->harga = hargaBaru;

break;

}

current = current->next;

}

}

void BukuManager::hapusBuku(const QString& judul)

{

if (!head) {

qDebug() << "Tidak ada buku dalam daftar.";

return;

}

if (head->judul == judul) {

Buku\* temp = head;

head = head->next;

delete temp;

qDebug() << "Buku dengan judul" << judul << "telah dihapus.";

return;

}

Buku\* current = head;

while (current->next) {

if (current->next->judul == judul) {

Buku\* temp = current->next;

current->next = current->next->next;

delete temp;

qDebug() << "Buku dengan judul" << judul << "telah dihapus.";

return;

}

current = current->next;

}

qDebug() << "Buku dengan judul" << judul << "tidak ditemukan dalam daftar.";

}

void BukuManager::tampilkanDataBukuUser(QTableWidget\* tableWidget)

{

tableWidget->clearContents();

tableWidget->setRowCount(0);

Buku\* current = head;

int row = 0;

while (current) {

tableWidget->insertRow(row);

tableWidget->setItem(row, 0, new QTableWidgetItem(current->judul));

tableWidget->setItem(row, 1, new QTableWidgetItem(current->harga));

current = current->next;

++row;

}

}

void BukuManager::pencarianBukuUser(const QString& kataKunci, QTableWidget\* tableWidget)

{

for (int row = 0; row < tableWidget->rowCount(); ++row) {

for (int col = 0; col < tableWidget->columnCount(); ++col) {

QTableWidgetItem\* item = tableWidget->item(row, col);

if (item && item->text().contains(kataKunci, Qt::CaseInsensitive)) {

tableWidget->setRowHidden(row, false);

break;

} else {

tableWidget->setRowHidden(row, true);

}

}

}

}

Penjelasan Kode:

BukuManager::BukuManager(QObject \*parent) : QObject(parent), head(nullptr) {

tambahBeberapaBuku();

}

Konstruktor ini digunakan untuk membuat objek BukuManager dan menginisialisasi pointer head menjadi nullptr. Selain itu, ini juga memanggil fungsi tambahBeberapaBuku() untuk menambahkan beberapa buku ke dalam sistem.

BukuManager::~BukuManager()

{

Buku\* current = head;

while (current) {

Buku\* next = current->next;

delete current;

current = next;

}

}

Destruktor ini digunakan untuk membersihkan memori yang dialokasikan dinamis. Ini menghapus semua node dalam linked list buku.

void BukuManager::tambahBeberapaBuku() {

Fungsi ini digunakan untuk menambahkan beberapa buku ke dalam sistem. Di sini, buku-buku ditambahkan dengan menggunakan fungsi tambahBuku.

void BukuManager::tambahBuku(const QString& judul, const QString& pengarang, const QString& tahunTerbit, const QString& harga)

{

Fungsi ini digunakan untuk menambahkan buku baru ke dalam sistem. Ini mengalokasikan memori untuk buku baru, mengatur nilai-nilai buku, dan menambahkannya ke linked list buku.

void BukuManager::tampilkanDataBuku(QTableWidget\* tableWidget)

Fungsi ini digunakan untuk menampilkan data buku ke dalam QTableWidget. Ini mengisi tabel dengan judul buku, pengarang, tahun terbit, dan harga dari setiap buku.

void BukuManager::perbaruiBuku(const QString& judulLama, const QString& judulBaru, const QString& pengarangBaru, const QString& tahunTerbitBaru, const QString& hargaBaru)

Fungsi ini digunakan untuk memperbarui informasi buku berdasarkan judul lama. Jika buku dengan judul yang sesuai ditemukan, informasi buku tersebut diperbarui dengan informasi baru yang diberikan.

void BukuManager::hapusBuku(const QString& judul)

Fungsi ini digunakan untuk menghapus buku dari sistem berdasarkan judul. Jika buku dengan judul yang sesuai ditemukan, buku tersebut dihapus dari linked list.

void BukuManager::tampilkanDataBukuUser(QTableWidget\* tableWidget)

Fungsi ini mirip dengan tampilkanDataBuku, tetapi hanya menampilkan judul dan harga buku ke dalam QTableWidget. Ini digunakan untuk menampilkan data buku dalam antarmuka pengguna.

void BukuManager::pencarianBukuUser(const QString& kataKunci, QTableWidget\* tableWidget)

{

Fungsi ini digunakan untuk melakukan pencarian buku berdasarkan kata kunci dan menampilkan hasil pencarian dalam QTableWidget. Ini membantu pengguna mencari buku berdasarkan judul atau kata kunci tertentu.

1. Proses koding berikutnya yaitu proses koding untuk dashboard admin. Dimana disini akan dibuat kode untuk menerapkan fungsi CRUD bagi admin untuk objek buku yang telah dibuat sebelumnya pada bukumanager.  
     
   Source Code admindashboard.h:

#ifndef ADMINDASHBOARD\_H

#define ADMINDASHBOARD\_H

#include <QMainWindow>

#include <QTableWidget>

#include "bukumanager.h"

namespace Ui {

class admindashboard;

}

class admindashboard : public QMainWindow

{

Q\_OBJECT

public:

explicit admindashboard(QWidget \*parent = nullptr);

~admindashboard();

private slots:

void on\_btn\_tambah\_clicked();

void refreshTableWidget();

void on\_btn\_edit\_clicked();

void on\_btn\_hapus\_clicked();

void on\_btn\_logout\_clicked();

private:

void tampilkanDataBuku();

Ui::admindashboard \*ui;

BukuManager \*bukuManager;

};

#endif

Penjelasan Kode:

#include <QMainWindow>

#include <QTableWidget>

#include "bukumanager.h"

File header ini memerlukan beberapa file header dari Qt untuk kelas QMainWindow dan QTableWidget. Selain itu, juga memerlukan header file bukumanager.h karena akan menggunakan kelas BukuManager.

Kelas admindashboard adalah turunan dari QMainWindow, sehingga merupakan jendela utama aplikasi.

Mendefinisikan slot untuk menangani sinyal yang dihasilkan oleh tombol-tombol dalam antarmuka pengguna.

Memiliki beberapa fungsi anggota private untuk melakukan operasi internal, seperti menampilkan data buku.

Source Code admindashboard.cpp:

#include "admindashboard.h"

#include "ui\_admindashboard.h"

#include "bukumanager.h"

#include "tambahbuku.h"

#include "QMessageBox"

#include "editbuku.h"

#include "mainwindow.h"

admindashboard::admindashboard(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::admindashboard),

bukuManager(new BukuManager)

{

ui->setupUi(this);

tampilkanDataBuku();

}

admindashboard::~admindashboard()

{

delete bukuManager;

delete ui;

}

void admindashboard::on\_btn\_tambah\_clicked()

{

tambahbuku \*dialog = new tambahbuku(bukuManager, this);

dialog->exec();

connect(dialog, &tambahbuku::dialogClosed, this, &admindashboard::refreshTableWidget);

}

void admindashboard::refreshTableWidget()

{

tampilkanDataBuku();

}

void admindashboard::tampilkanDataBuku()

{

bukuManager->tampilkanDataBuku(ui->tableWidget);

}

void admindashboard::on\_btn\_edit\_clicked()

{

QModelIndexList selectedIndexes = ui->tableWidget->selectionModel()->selectedRows();

if(selectedIndexes.isEmpty()) {

QMessageBox::warning(this, "Peringatan", "Harap pilih data terlebih dahulu.");

return;

}

int selectedRow = selectedIndexes.at(0).row();

QTableWidgetItem\* judulItem = ui->tableWidget->item(selectedRow, 0);

QTableWidgetItem\* pengarangItem = ui->tableWidget->item(selectedRow, 1);

QTableWidgetItem\* tahunTerbitItem = ui->tableWidget->item(selectedRow, 2);

QTableWidgetItem\* hargaItem = ui->tableWidget->item(selectedRow, 3);

QString judul = judulItem->text();

QString pengarang = pengarangItem->text();

QString tahunTerbit = tahunTerbitItem->text();

QString harga = hargaItem->text();

editbuku \*dialog = new editbuku(bukuManager, judul, this);

dialog->setJudul(judul);

dialog->setPengarang(pengarang);

dialog->setTahunTerbit(tahunTerbit);

dialog->setHarga(harga);

dialog->exec();

}

void admindashboard::on\_btn\_hapus\_clicked()

{

QModelIndexList selectedIndexes = ui->tableWidget->selectionModel()->selectedRows();

if(selectedIndexes.isEmpty()) {

QMessageBox::warning(this, "Peringatan", "Harap pilih data terlebih dahulu.");

return;

}

int choice = QMessageBox::question(this, "Konfirmasi Hapus", "Anda yakin ingin menghapus buku?", QMessageBox::Yes | QMessageBox::No);

if(choice == QMessageBox::No)

return;

int selectedRow = selectedIndexes.at(0).row();

QTableWidgetItem\* judulItem = ui->tableWidget->item(selectedRow, 0);

QString judul = judulItem->text();

bukuManager->hapusBuku(judul);

ui->tableWidget->removeRow(selectedRow);

QMessageBox::information(this, "Hapus Buku", "Buku berhasil dihapus.");

qDebug() << "Isi linked list setelah penghapusan:";

bukuManager->tampilkanDataBuku();

}

void admindashboard::on\_btn\_logout\_clicked()

{

this->close();

this->close();

MainWindow \*mainWindow = new MainWindow();

mainWindow->show();

}

Penjelasan Kode:

#include "admindashboard.h"

#include "ui\_admindashboard.h"

#include "bukumanager.h"

#include "tambahbuku.h"

#include "QMessageBox"

#include "editbuku.h"

#include "mainwindow.h"

Mengimpor header yang diperlukan untuk kelas admindashboard, UI yang dihasilkan oleh Qt Designer, BukuManager, tambahbuku, QMessageBox, editbuku, dan mainwindow.

admindashboard::admindashboard(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::admindashboard),

bukuManager(new BukuManager)

{

ui->setupUi(this);

tampilkanDataBuku();

}

Konstruktor admindashboard menginisialisasi antarmuka pengguna, objek BukuManager, dan menampilkan data buku.

void admindashboard::on\_btn\_tambah\_clicked()

{

tambahbuku \*dialog = new tambahbuku(bukuManager, this);

dialog->exec();

connect(dialog, &tambahbuku::dialogClosed, this, &admindashboard::refreshTableWidget);

}

Slot ini dipicu ketika tombol tambah diklik. Ini membuat dialog tambah buku, menampilkannya, dan menghubungkan sinyal dialogClosed dari dialog dengan slot refreshTableWidget() untuk menyegarkan data buku setelah dialog ditutup.

void admindashboard::refreshTableWidget()

{

tampilkanDataBuku();

}

Slot ini merefresh tabel buku dengan memanggil fungsi tampilkanDataBuku().

void admindashboard::on\_btn\_edit\_clicked()

Slot ini dipicu ketika tombol edit diklik. Ini memungkinkan pengguna untuk mengedit entri buku yang dipilih.

void admindashboard::on\_btn\_hapus\_clicked()

Slot ini dipicu ketika tombol hapus diklik. Ini memungkinkan pengguna untuk menghapus entri buku yang dipilih.

void admindashboard::on\_btn\_logout\_clicked()

Slot ini dipicu ketika tombol logout diklik. Ini mengembalikan pengguna ke layar login.

1. Selanjutnya yaitu melakukan proses koding untuk salah satu fungsi CRUD yaitu Create atau tambah. Proses ini akan ditangani oleh class tambahbuku.h dan tambahbuku.cpp

Source Code tambahbuku.h:

#ifndef TAMBAHBUKU\_H

#define TAMBAHBUKU\_H

#include <QDialog>

#include "bukumanager.h"

namespace Ui {

class tambahbuku;

}

class tambahbuku : public QDialog

{

Q\_OBJECT

public:

explicit tambahbuku(BukuManager \*bukuManager, QWidget \*parent = nullptr);

~tambahbuku();

private slots:

void tambahBuku();

signals:

void dialogClosed();

private:

Ui::tambahbuku \*ui;

BukuManager \*bukuManager;

};

#endif // TAMBAHBUKU\_H

Penjelasan Kode:

private slots:

void tambahBuku();

Mendeklarasikan slot tambahBuku() yang akan dipanggil ketika tombol tambah di klik.

signals:

void dialogClosed();

Mendeklarasikan sinyal dialogClosed() yang akan dipancarkan ketika dialog ditutup.

private:

Ui::tambahbuku \*ui;

BukuManager \*bukuManager;

Mendeklarasikan pointer ke antarmuka pengguna ui dan pointer ke objek BukuManager untuk digunakan dalam kelas ini.

Source Code tambahbuku.cpp:

#include "tambahbuku.h"

#include "ui\_tambahbuku.h"

#include "QMessageBox"

#include <QDebug>

#include "QDialogButtonBox"

tambahbuku::tambahbuku(BukuManager \*bukuManager, QWidget \*parent) :

QDialog(parent),

ui(new Ui::tambahbuku),

bukuManager(bukuManager)

{

ui->setupUi(this);

}

tambahbuku::~tambahbuku()

{

delete ui;

emit dialogClosed();

}

void tambahbuku::tambahBuku() {

if (ui->tf\_judul->text().isEmpty() || ui->tf\_pengarang->text().isEmpty() || ui->tf\_tahun->text().isEmpty() || ui->tf\_harga->text().isEmpty()) {

QMessageBox::warning(this, "Peringatan", "Mohon lengkapi semua kolom.");

return;

}

QString judul = ui->tf\_judul->text();

QString pengarang = ui->tf\_pengarang->text();

QString tahunTerbit = ui->tf\_tahun->text();

QString harga = ui->tf\_harga->text();

bukuManager->tambahBuku(judul, pengarang, tahunTerbit, harga);

QMessageBox::information(this, "Informasi", "Buku berhasil ditambahkan ke dalam daftar.");

qDebug() << "Daftar Buku Setelah Penambahan:";

bukuManager->tampilkanDataBuku();

this->close();

}

Penjelasan Kode:

* Membuat fungsi tambahBuku() yang dipanggil ketika tombol tambah di klik.
* Memeriksa apakah semua kolom telah diisi.
* Mengambil nilai dari kolom input.
* Menambahkan buku ke dalam daftar menggunakan objek BukuManager.
* Menampilkan pesan informasi setelah penambahan buku.
* Menutup dialog setelah penambahan buku selesai.

1. Proses koding berikutnya yaitu proses pembuatan kode untuk fungsi Edit. Fungsi edit diatur dalam class editbuku.h dan editbuku.cpp.

Source Code editbuku.h:

#ifndef EDITBUKU\_H

#define EDITBUKU\_H

#include <QDialog>

#include "bukumanager.h"

namespace Ui {

class editbuku;

}

class editbuku : public QDialog

{

Q\_OBJECT

public:

explicit editbuku(BukuManager \*bukuManager, const QString &judulLama, QWidget \*parent = nullptr);

~editbuku();

void setJudul(const QString &judul);

void setPengarang(const QString &pengarang);

void setTahunTerbit(const QString &tahunTerbit);

void setHarga(const QString &harga);

signals:

void dialogClosed();

private slots:

void editBuku();

private:

Ui::editbuku \*ui;

BukuManager \*bukuManager;

QString judulLama;

};

#endif // EDITBUKU\_H

Penjelasan Kode:

* Mendefinisikan kelas editbuku yang merupakan turunan dari QDialog.
* Menyediakan konstruktor dan destruktor.
* Mendeklarasikan metode setJudul(), setPengarang(), setTahunTerbit(), dan setHarga() untuk mengatur nilai elemen UI.
* Mendeklarasikan sinyal dialogClosed() untuk memberi tahu bahwa dialog telah ditutup.
* Mendeklarasikan slot editBuku() yang akan dipanggil saat tombol edit diklik.
* Menyimpan pointer ke antarmuka pengguna dan objek BukuManager.

Source Code editbuku.cpp:

#include "editbuku.h"

#include "ui\_editbuku.h"

#include <QMessageBox>

editbuku::editbuku(BukuManager \*bukuManager, const QString &judulLama, QWidget \*parent) :

QDialog(parent),

ui(new Ui::editbuku),

bukuManager(bukuManager),

judulLama(judulLama)

{

ui->setupUi(this);

connect(ui->btn\_edit, &QPushButton::clicked, this, &editbuku::editBuku);

}

editbuku::~editbuku()

{

delete ui;

}

void editbuku::setJudul(const QString &judul)

{

ui->tf\_judul->setText(judul);

}

void editbuku::setPengarang(const QString &pengarang)

{

ui->tf\_pengarang->setText(pengarang);

}

void editbuku::setTahunTerbit(const QString &tahunTerbit)

{

ui->tf\_tahun->setText(tahunTerbit);

}

void editbuku::setHarga(const QString &harga)

{

ui->tf\_harga->setText(harga);

}

void editbuku::editBuku()

{

QString judulBaru = ui->tf\_judul->text();

QString pengarangBaru = ui->tf\_pengarang->text();

QString tahunTerbitBaru = ui->tf\_tahun->text();

QString hargaBaru = ui->tf\_harga->text();

bukuManager->perbaruiBuku(judulLama, judulBaru, pengarangBaru, tahunTerbitBaru, hargaBaru);

QMessageBox::information(this, "Edit Buku", "Buku berhasil diperbarui.");

emit dialogClosed();

this->close();

}

Penjelasan Kode:

connect(ui->btn\_edit, &QPushButton::clicked, this, &editbuku::editBuku);

Menghubungkan sinyal dari tombol edit (btn\_edit) ke slot editBuku().

void editbuku::setJudul(const QString &judul)

{

ui->tf\_judul->setText(judul);

}

void editbuku::setPengarang(const QString &pengarang)

{

ui->tf\_pengarang->setText(pengarang);

}

void editbuku::setTahunTerbit(const QString &tahunTerbit)

{

ui->tf\_tahun->setText(tahunTerbit);

}

void editbuku::setHarga(const QString &harga)

{

ui->tf\_harga->setText(harga);

}

Metode-metode ini digunakan untuk mengatur nilai dari elemen-elemen UI seperti judul, pengarang, tahun terbit, dan harga.

void editbuku::editBuku()

* Metode ini dipanggil ketika tombol edit diklik.
* Mengambil nilai baru dari elemen-elemen UI.
* Memanggil metode perbaruiBuku() dari BukuManager untuk memperbarui buku.
* Mengirimkan sinyal dialogClosed() untuk memberi tahu bahwa dialog telah ditutup.
* Menutup dialog setelah proses edit selesai.

1. Berikutnya, setelah proses koding CRUD untuk admin selesai, kita mulai melakukan pembuatan kode untuk Dashboard User. Dimana user nantinya akan dapat melihat daftar buku secara singkat dan melakukan pencarian buku. Semua fungsi ini akan diatur dalam userdashboard.h dan userdashboard.cpp.

Source Code userdashboard.h:

#ifndef USERDASHBOARD\_H

#define USERDASHBOARD\_H

#include <QMainWindow>

#include "bukumanager.h"

#include "ui\_userdashboard.h"

namespace Ui {

class userdashboard;

}

class UserDashboard : public QMainWindow

{

Q\_OBJECT

public:

explicit UserDashboard(QWidget \*parent = nullptr);

~UserDashboard();

public slots:

void on\_btn\_cari\_clicked();

void on\_btn\_logout\_clicked();

private:

Ui::userdashboard \*ui;

BukuManager \*bukuManager;

void refreshTableWidget();

void tampilkanDataBuku();

};

#endif // USERDASHBOARD\_H

Penjelasan Kode:

void on\_btn\_cari\_clicked();

Slot ini akan dipanggil ketika tombol cari (btn\_cari) ditekan. Berfungsi untuk menangani pencarian buku berdasarkan kata kunci.

void on\_btn\_logout\_clicked();

Slot ini akan dipanggil ketika tombol logout (btn\_logout) ditekan. Berfungsi untuk menangani proses logout pengguna dari dashboard.

refreshTableWidget(): Metode untuk menyegarkan tabel buku di antarmuka pengguna.

tampilkanDataBuku(): Metode untuk menampilkan data buku pada tabel di antarmuka pengguna.

Source Code userdashboard.cpp:

#include "userdashboard.h"

#include "bukumanager.h"

#include <QTableWidgetItem>

#include <QMessageBox>

#include "mainwindow.h"

UserDashboard::UserDashboard(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::userdashboard),

bukuManager(new BukuManager())

{

ui->setupUi(this);

refreshTableWidget();

connect(ui->btn\_cari, &QPushButton::clicked, this, &UserDashboard::on\_btn\_cari\_clicked);

}

UserDashboard::~UserDashboard()

{

delete ui;

delete bukuManager;

}

void UserDashboard::refreshTableWidget()

{

bukuManager->tampilkanDataBukuUser(ui->tableWidget);

}

void UserDashboard::on\_btn\_cari\_clicked()

{

QString kataKunci = ui->tf\_cari->text();

bukuManager->pencarianBukuUser(kataKunci, ui->tableWidget);

}

void UserDashboard::on\_btn\_logout\_clicked()

{

this->close();

this->close();

MainWindow \*mainWindow = new MainWindow();

mainWindow->show();

this->close();

}

Penjelasan Kode:

void UserDashboard::refreshTableWidget()

Metode ini memperbarui tabel buku pada antarmuka pengguna dengan memanggil metode tampilkanDataBukuUser() dari objek bukuManager.

void UserDashboard::on\_btn\_cari\_clicked()

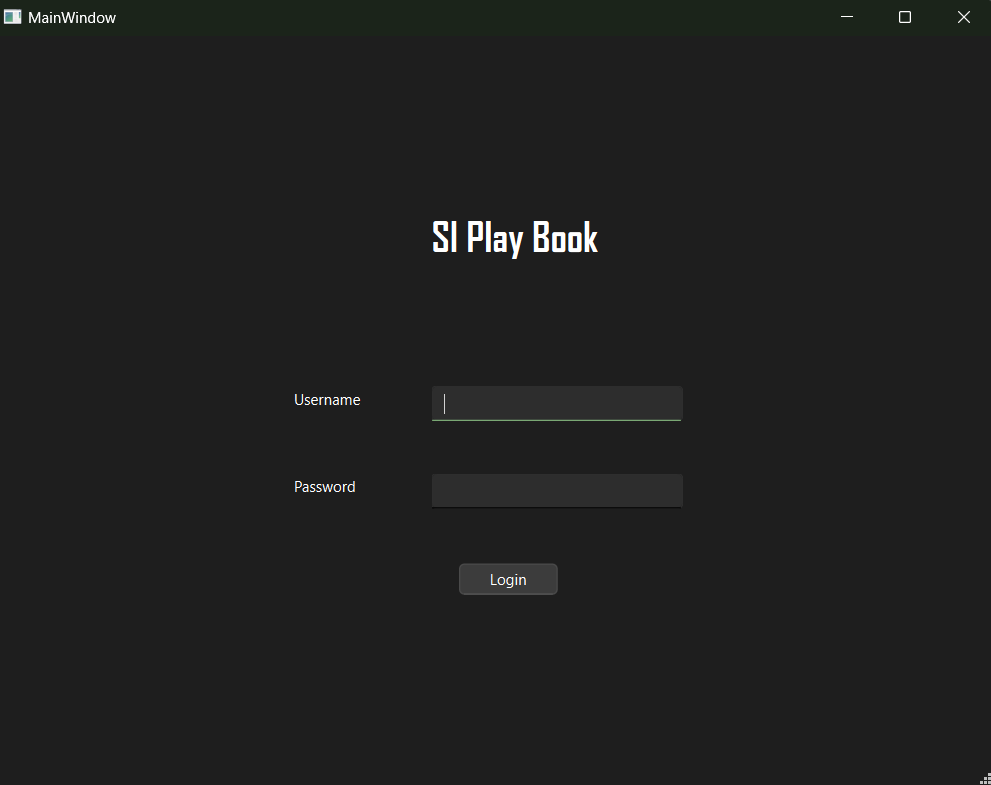
Slot ini dipanggil ketika tombol cari (btn\_cari) ditekan. Berfungsi untuk menangani pencarian buku berdasarkan kata kunci yang dimasukkan oleh pengguna.

void UserDashboard::on\_btn\_logout\_clicked()

Slot ini dipanggil ketika tombol logout (btn\_logout) ditekan. Berfungsi untuk menangani proses logout pengguna dari dashboard dan menampilkan jendela utama (MainWindow).

## **Output**

1. Tampilan Login

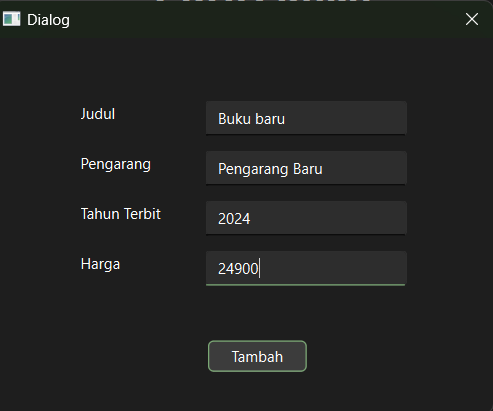


Gambar 8. Tampilan Login

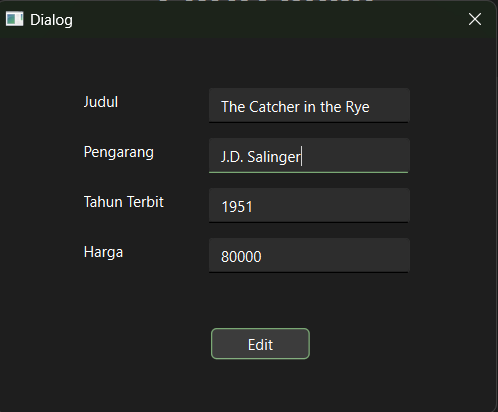
1. Tampilan Dashboard Admin



Gambar 9. Tampilan Dashboard Admin

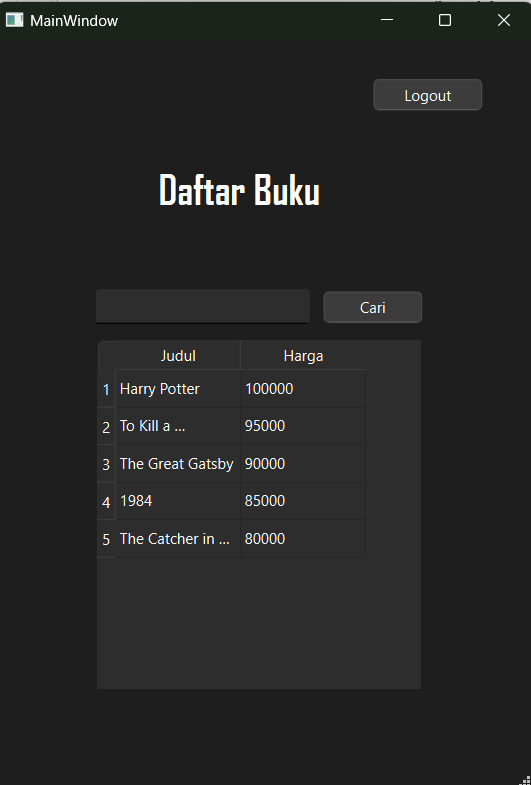
1. Tampilan Tambah Buku

Gambar 10. Tampilan Tambah Buku

1. Tampilan Edit Buku

Gambar 11. Tampilan Edit Buku

1. Tampilan Dashboard User



Gambar 12. Tampilan Dashboard User

# **BAB IV**

# **KESIMPULAN DAN SARAN**

## **Kesimpulan**

Dalam praktikum ini, kami berhasil mengimplementasikan berbagai fitur dalam aplikasi Sistem Informasi Playbook menggunakan Qt Framework. Kami menggambarkan proses pembuatan antarmuka pengguna (UI) untuk dashboard pengguna dan dashboard admin, serta implementasi fitur-fitur seperti penambahan, penghapusan, dan penyuntingan buku.

Selain itu, kami juga berhasil menangani fungsi pencarian buku berdasarkan kata kunci yang dimasukkan oleh pengguna.

Melalui praktikum ini, kami mendapatkan pemahaman yang lebih dalam tentang pengembangan perangkat lunak menggunakan Qt dan pentingnya desain antarmuka pengguna yang intuitif dalam meningkatkan pengalaman pengguna. Kami yakin bahwa praktikum ini membantu meningkatkan pemahaman kami tentang konsep-konsep dasar dalam pengembangan perangkat lunak dan memberikan landasan yang kuat untuk proyek-proyek masa depan

## **Saran**

Untuk saran dalam pengembangan aplikasi ini, kami merekomendasikan beberapa hal yang dapat diperhatikan di masa depan. Pertama, penting untuk mengoptimalkan kinerja aplikasi agar responsif dan efisien, terutama saat menghadapi volume data yang besar.

Selain itu, desain antarmuka pengguna perlu terus diperbaiki dengan memperhatikan prinsip-prinsip desain modern dan menjaga konsistensi antarmuka. Pengujian menyeluruh juga diperlukan untuk memastikan kestabilan dan kualitas aplikasi, serta pemeliharaan berkala untuk perbaikan bug dan peningkatan sesuai umpan balik pengguna.

Dengan menerapkan saran-saran tersebut, diharapkan aplikasi dapat menjadi lebih efektif dan berguna bagi pengguna perpustakaan.

# **DAFTAR PUSTAKA**

Algoritma .(2022). Apa itu C++? Algorit.ma. Diakses melalui https://algorit.ma/blog/apa-itu-c-2022/

Puspasari, E. K. (2022). Bahasa C++ adalah: Pengertian dan Manfaat. Academy Alterra. Diakses melalui <https://academy.alterra.id/blog/bahasa-c-adalah-pengertian-dan-manfaat%EF%BF%BC/>

Santoso, R. (2014). Single Linked List pada C++. Nblognlife. Diakses melalui <https://www.nblognlife.com/2014/12/single-linked-list-pada-c.html>

Qt . (n.d). QLinkedList Class. Qt Documentation. Diakses melalui <https://doc.qt.io/qt-6/qlinkedlist.html>

Qt . (n.d). Developing with Qt Creator. Qt Documentation. Diakses melalui https://doc.qt.io/qtcreator/creator-overview.html

Revou. (n.d). Bahasa C++. Revou.co. Diakses melalui <https://revou.co/kosakata/bahasa-cpp>