

LAPORAN TUGAS BESAR
PROYEK STRUKTUR DATA DAN ALGORITMA



Disusun Oleh :

Kelompok 2

- | | |
|---------------------------|-----------|
| 1. Aggra Kurnia Idhan | G1A024003 |
| 2. Atikah Putri Utami | G1A024027 |
| 3. Agief Vemas Afrivanzah | G1A024037 |
| 4. Farhan Khairullah | G1A024043 |

Nama Asisten Dosen :

- | | |
|------------------------------|-----------|
| 1. Abdi Agung Kurniawan | G1A022011 |
| 2. Diodo Arrahman | G1A022027 |
| 3. Diosi Putri Arlita | G1A023012 |
| 4. Lio Kusnata | G1A023013 |
| 5. Sallaa Fikriyatul 'Arifah | G1A023015 |
| 6. Muhammad Yasser G.T. A. | G1A023030 |
| 7. Anis Syarifatul Mursyidah | G1A023036 |
| 8. Rayhan Muhammad Adha | G1A023051 |
| 9. Najwa Nabilah Wibisono | G1A023065 |
| 10. Dinda Krisnauli Pakpahan | G1A023076 |

Dosen Pengampu :

1. Arie Vatriesia, S.T., M.TI, Ph.D.
2. Kurnia Anggraini, S.T., M.T, Ph.D

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2025

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh,

Puji syukur kehadiran Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas besar proyek struktur data dan algoritma dengan judul "SI Play Book". Tugas besar ini kami buat dengan tujuan untuk mengimplementasikan struktur data dan algoritma yang telah kami pelajari selama perkuliahan ini, dan juga untuk membantu kami memperdalam pemahaman mengenai konsep-konsep tentang struktur data dan algoritma.

Proyek ini diharapkan dapat memberikan manfaat dan pemahaman yang lebih baik bagi kami maupun bagi pembaca, khususnya dalam hal penerapan struktur data dan algoritma dalam kehidupan sehari-hari. Dalam pembuatan proyek ini, kami menyadari bahwa masih banyak kekurangan dan kelemahan, namun kami berharap bahwa proyek ini dapat dijadikan sebagai bahan referensi dan inspirasi untuk pengembangan aplikasi yang lebih baik di masa depan.

Akhir kata, kami mengucapkan terima kasih kepada semua pihak yang telah membantu dalam penyelesaian tugas besar ini, baik secara langsung maupun tidak langsung. Semoga proyek ini dapat bermanfaat bagi kita semua.

Wassalamualaikum Warahmatullahi Wabarakatuh.

Bengkulu, 16 Mei 2025

Penulis

DAFTAR ISI

KATA PENGANTAR.....	ii
DAFTAR ISI.....	iii
DAFTAR GAMBAR	iv
BAB I.....	1
PENDAHULUAN.....	1
A. Latar Belakang.....	1
B. Rumusan Masalah.....	1
C. Tujuan dan Manfaat.....	1
BAB II.....	2
LANDASAN TEORI.....	2
A. Bahasa C++.....	2
B. Array List.....	2
C. Aplikasi QT.....	2
BAB III.....	3
PEMBAHASAN.....	3
A. Pengertian Array List, Perbedaan Array List dan Linked List, Kelebihan dan Kekurangan dalam Pemrograman.....	3
B. Cara Membuat Array List dan Implementasi Elemen dalam Program.....	4
C. Pengertian Pointer dalam Array List dan Bagaimana Cara Menggunakannya.....	6
D. Cara Menambah atau Menghapus Elemen dalam Array List, dan Bagaimana penerapannya dalam Kehidupan Nyata.....	7
➤ Membuat Program dengan Tema SI Play Book dalam Linked List.....	8
➤ Tampilan Program dengan Tema SI Play Book.....	43
BAB IV.....	47
KESIMPULAN DAN SARAN.....	47
A. Kesimpulan.....	47
B. Saran.....	47
DAFTAR PUSTAKA.....	48
LAMPIRAN.....	49

DAFTAR GAMBAR

Gambar 1 Source Code mainwindow.h	8
Gambar 2 Source Code admindashboard.h.....	10
Gambar 3 Source Code userdashboard.h.....	12
Gambar 4 Source Code bukumanager.h.....	13
Gambar 5 Source Code tambahbuku.h.....	15
Gambar 6 Source Code editbuku.h	17
Gambar 7 Source Code mainwindow.cpp.....	19
Gambar 8 Source Code mainwindow.cpp.....	19
Gambar 9 Source Code admindashboard.cpp	23
Gambar 10 Source Code admindashboard.cpp	23
Gambar 11 Source Code admindashboard.cpp	23
Gambar 12 Source Code userdashboard.cpp.....	27
Gambar 13 Source Code userdashboard.cpp.....	27
Gambar 14 Source Code bukumanager.cpp.....	31
Gambar 15 Source Code bukumanager.cpp.....	31
Gambar 16 Source Code bukumanager.cpp.....	31
Gambar 17 Source Code tambahbuku.cpp.....	37
Gambar 18 Source Code editbuku.cpp	40
Gambar 19 Source Code editbuku.cpp	40
Gambar 20 Tampilan Login dengan ID Admin	43
Gambar 21 Tampilan Login dengan ID User.....	44
Gambar 22 Tampilan Admin	44
Gambar 23 Tampilan Admin Menambahkan Buku	45
Gambar 24 Tampilan Admin Menghapus Buku	45
Gambar 25 Tampilan Admin Mengupdate Buku.....	46
Gambar 26 Tampilan User.....	46

BAB I

PENDAHULUAN

A. Latar Belakang

Visual Studio Code adalah alat pengembangan sumber terbuka dan lintas platform yang berfokus pada penyuntingan kode di berbagai skenario pengembangan, termasuk pengembangan web, seluler, dan cloud. Perkembangan teknologi informasi yang pesat telah mendorong pentingnya penguasaan keterampilan pemrograman, terutama dalam dunia pendidikan dan industri. Salah satu bahasa pemrograman yang banyak digunakan dan diajarkan di tingkat dasar hingga lanjutan adalah C++. Bahasa ini dikenal karena kemampuannya, mulai dari prosedural hingga berorientasi objek, serta efisiensinya dalam pengelolaan memori dan kecepatan eksekusi program. Dalam upaya membekali mahasiswa dengan dasar-dasar pemrograman yang kuat, kegiatan praktikum ini dirancang menggunakan Visual Studio Code (VSC) sebagai lingkungan pengembangan. VSC merupakan salah satu *code editor* modern yang bersifat ringan, fleksibel, dan mendukung berbagai ekstensi yang memudahkan pengembangan program C++, termasuk fitur *IntelliSense*, *syntax highlighting*, dan *integrated debugger*. *Visual Studio Code* juga adalah alat pengembangan sumber terbuka dan lintas platform yang berfokus pada penyuntingan kode di berbagai skenario pengembangan, termasuk pengembangan web, seluler, dan cloud.

Melalui praktikum ini, mahasiswa diharapkan mampu memahami struktur dasar bahasa C++, mengembangkan logika pemrograman, serta membiasakan diri menggunakan alat bantu pengembangan perangkat lunak secara profesional. Praktikum ini juga menjadi fondasi penting bagi mata kuliah lanjutan yang menuntut kemampuan logika dan analisis dalam menyelesaikan permasalahan komputasi.

B. Rumusan Masalah

1. Buatlah sebuah projek *SI PLAY BOOK (LINKED LIST)* dengan ketentuan yang telah diberikan?

C. Tujuan

Praktikum dengan bahasa pemrograman C++ menggunakan *visual studio code* bertujuan untuk:

1. Melatih kemampuan logika dan algoritma dalam menyelesaikan permasalahan sederhana melalui implementasi program
2. Mengetahui dan menguasai penggunaan Visual Studio Code sebagai lingkungan pengembangan perangkat lunak yang mendukung bahasa C++.
3. Memahami dasar-dasar sintaks C++, seperti struktur program, tipe data, variabel, operator, dan fungsi.

BAB II

LANDASAN TEORI

A. Bahasa C++

C++ adalah bahasa pemrograman komputer yang dikembangkan oleh Bjarne Stroustrup pada awal 1980-an di Bell Labs. Bahasa ini merupakan pengembangan dari bahasa C, yang dikenal karena efisiensinya dan kemampuannya dalam pengelolaan memori. C++ dirancang untuk mendukung pemrograman berorientasi objek (OOP), yang memungkinkan pengembang untuk membuat program yang lebih modular, terstruktur, dan mudah dipelihara. Dengan adanya OOP, programmer dapat menggunakan konsep seperti enkapsulasi, pewarisan, dan polimorfisme, yang membantu dalam pengorganisasian kode.

C++ juga dilengkapi dengan Standard Template Library (STL), yang menyediakan berbagai struktur data dan algoritma siap pakai. STL mencakup kontainer seperti vector, list, map, dan set, serta algoritma untuk pencarian, pengurutan, manipulasi data. Penggunaan STL memungkinkan pengembang untuk menghemat waktu dan usaha dalam menulis kode, karena mereka dapat menggunakan komponen yang telah teruji dan dioptimalkan. Bahasa ini juga sangat populer di berbagai bidang-bidang, termasuk pada pengembangan perangkat lunak pada sistem, pada aplikasi desktop, game, dan perangkat lunak yang juga memerlukan pengolahan data intensif sehingga juga menarik apabila digunakan.

B. Array List

ArrayList adalah struktur data yang digunakan dalam pemrograman untuk menyimpan elemen dalam bentuk daftar yang dapat diubah ukurannya secara dinamis. Berasal dari bahasa pemrograman Java, ArrayList merupakan bagian dari Java Collections Framework dan memungkinkan pengguna untuk menyimpan objek dalam urutan tertentu. Salah satu keunggulan utama dari ArrayList adalah kemampuannya untuk menambah, menghapus, dan mengakses elemen dengan mudah, tanpa perlu menentukan ukuran awal, sehingga sangat fleksibel untuk digunakan dalam berbagai aplikasi. ArrayList menyimpan elemen dalam array yang dapat diperluas, yang berarti ketika kapasitas array terisi, ArrayList secara otomatis akan membuat array baru dengan ukuran yang lebih besar.

Meskipun ArrayList menawarkan akses cepat ke elemen menggunakan indeks, operasi penyisipan dan penghapusan elemen di tengah daftar dapat memerlukan waktu $O(N)$ karena elemen-elemen lainnya perlu digeser. Dengan demikian, ArrayList bisa dapat menjadi pilihan yang sangat populer untuk pengelolaan pada koleksi data yang sangat dinamis dan juga sering digunakan pada atau dalam melakukan pengembangan pada suatu aplikasi yang digunakan.

BAB III

HASIL DAN PEMBAHASAN

A. Pengertian Array List, Perbedaan Array List dan Linked List, Kelebihan dan Kekurangan dalam Pemrograman

ArrayList dan Linked List adalah dua struktur data yang sering digunakan dalam pemrograman untuk menyimpan dan mengelola kumpulan data, namun keduanya memiliki karakteristik, kelebihan, dan kekurangan yang berbeda. ArrayList adalah struktur data yang mendasarkan penyimpanan elemen pada sebuah array dinamis. Ini berarti bahwa ukuran ArrayList dapat berubah secara otomatis sesuai dengan jumlah elemen yang disimpan. Salah satu keunggulan utama dari ArrayList adalah kemudahannya dalam mengakses elemen secara langsung menggunakan indeks, sehingga akses data menjadi cepat dan efisien. Misalnya, jika kita ingin mengakses elemen ke-5 dalam ArrayList, kita dapat melakukannya dalam waktu konstan $O(1)$ karena kita hanya perlu menggunakan indeks untuk langsung menuju elemen tersebut. Selain itu, ArrayList juga memudahkan proses penambahan elemen di akhir daftar dengan menggunakan metode seperti `add()`, yang memungkinkan pengguna untuk menambahkan elemen yang baru tanpa perlu menentukan bagaimana ukuran yang awalnya. Namun, karena berbasis array, operasi penyisipan atau penghapusan elemen di tengah ArrayList biasanya memerlukan waktu dan proses penggeseran elemen-elemen lainnya, yang menjadikan performanya sedikit menurun untuk operasi tersebut. Jika kita menghapus elemen dari tengah ArrayList, semua elemen setelah elemen yang dihapus harus digeser satu posisi ke kiri, yang dapat memakan waktu $O(N)$ dalam kasus terburuk. Meskipun demikian, ArrayList tetap menjadi pilihan yang populer dalam banyak aplikasi karena kemudahan penggunaannya.

Di sisi lain, Linked List adalah struktur data linier yang tersusun dari node-node yang saling terhubung menggunakan pointer. Setiap node dalam Linked List menyimpan dua komponen utama: data dan pointer yang menunjuk ke node berikutnya dalam urutan. Keistimewaan Linked List terletak pada kemampuannya mengalokasikan memori secara dinamis saat dijalankan, sehingga ukuran Linked List dapat bertambah atau berkurang sesuai kebutuhan tanpa perlu menyisihkan ruang memori secara tetap seperti pada ArrayList. Hal ini memungkinkan Linked List untuk lebih efisien dalam penggunaan memori, terutama ketika jumlah data yang disimpan tidak dapat diprediksi sebelumnya. Linked List juga sangat efisien dalam operasi penyisipan dan penghapusan elemen, terutama di tengah daftar, karena cukup mengubah pointer antar node tanpa perlu menggeser elemen lainnya. Misalnya, untuk menyisipkan elemen baru di antara dua node, kita hanya perlu mengubah pointer dari node sebelumnya dan node berikutnya, yang menjadikan operasi ini sangat cepat. Namun,

kelemahan dari Linked List adalah akses ke elemen menjadi lebih lambat karena tidak mendukung akses acak seperti ArrayList untuk mencapai sebuah elemen, pengaksesan harus dilakukan secara berurutan dari awal Linked List. Berarti mengakses elemen ke-5, harus melintasi semua node sebelumnya, yang akan memakan waktu $O(N)$. Selain itu, penggunaan memori Linked List lebih banyak karena node harus menyimpan pointer tambahan selain data, yang juga dapat menjadi suatu masalah yang terjadi jika apabila jumlah node yang digunakan sangat besar.

Dengan memahami perbedaan mendasar antara ArrayList dan Linked List, kita dapat lebih baik dalam memilih struktur data yang tepat untuk kebutuhan pemrograman kita. ArrayList lebih cocok untuk aplikasi yang memerlukan akses cepat ke elemen berdasarkan indeks dan di mana ukuran data relatif stabil, sementara Linked List lebih ideal untuk situasi di mana penyisipan dan penghapusan elemen sering dilakukan, dan ukuran data dapat bervariasi secara signifikan. Keduanya memiliki tempatnya masing-masing dalam pengembangan perangkat lunak, dan pemilihan keduanya harus didasarkan pada kebutuhan masing-masing.

Dengan memahami perbedaan mendasar antara ArrayList dan Linked List, kita dapat lebih baik dalam memilih struktur data yang tepat untuk kebutuhan pemrograman kita. ArrayList lebih cocok untuk aplikasi yang memerlukan akses cepat ke elemen berdasarkan indeks dan di mana ukuran data relatif stabil, sementara Linked List lebih ideal untuk situasi di mana penyisipan dan penghapusan elemen sering dilakukan, dan ukuran data dapat bervariasi secara signifikan. Keduanya memiliki tempatnya masing-masing dalam pengembangan perangkat lunak, dan pemilihan keduanya harus didasarkan pada kebutuhan masing-masing.

B. Cara Membuat Array List dan Implementasi Elemen dalam Program

a. Definisi Struktur Elemen

Setiap elemen dalam array list disimpan dalam sebuah struktur yang berisi data utama, misalnya tipe int. Contoh struktur elemen:

```
struct Element {  
    int data; // nilai yang disimpan dalam elemen array list  
};
```

b. Membuat Array List Kosong

Array list adalah kumpulan elemen yang disimpan dalam array statis atau dinamis. Kita perlu membuat variabel size yang menyimpan jumlah elemen yang sudah ada dalam array list. Pada awalnya, size diinisialisasi 0, menandakan array list kosong. Tidak ada konsep head atau pointer seperti di linked list, karena array list menggunakan

indeks untuk mengakses elemen.

c. Menambahkan Elemen ke Array List

Untuk menambahkan elemen baru, simpan data-data indeks size dalam array data. Setelah penyimpanan, tingkatkan nilai size sebesar 1, menandakan ada satu elemen baru dalam array list. Sebaiknya sebelum menambahkan, lakukan pengecekan apakah size sudah mencapai kapasitas maksimum array agar menghindari (buffer overflow).

d. Bagian Kode yang Perlu Disesuaikan

Jika data adalah array dari Element, maka untuk menyimpan nilai harus menggunakan `data[size].data = value;` bukan `data[size] = value;`. Variabel data dan size harus dideklarasikan sebagai anggota dari sebuah struktur atau kelas (misal ArrayList). Fungsi add juga akan harus mengembalikan nilai boolean untuk menandakan apakah penambahan berhasil atau penambahan itu menjadi gagal (apabila array tersebut jadi penuh).

e. Contoh Implementasi Fungsi add yang benar

```
bool add(int value) {  
  
    if (size >= MAX_SIZE) {  
        // Array list sudah penuh, tidak bisa menambahkan elemen baru return false;  
    }  
  
    data[size].data = value; // simpan nilai pada elemen ke size size++;  
    // tingkatkan ukuran array list  
    return true;  
}
```

Setelah membuat definisi struktur data untuk elemen array list, selanjutnya kita membuat Array List yang kosong: Buat sebuah variabel yang akan menjadi penanda posisi elemen pertama dalam array list. Pada awalnya, array list kosong dengan ukuran 0. Untuk menambahkan elemen baru ke array list, kita hanya cukup menyimpan data pada indeks berikutnya kedalam array dan kemudian akan memperbarui ukurannya kedalam array list.

Implementasi penambahan elemen:

```
bool add(int value) {  
    data[size] = value; // simpan elemen baru pada indeks terakhir (size) size++;  
    // tingkatkan ukuran array list  
    return true;  
}
```

}

Memahami array list membuka pintu untuk pengembangan perangkat lunak yang lebih efisien dalam konteks akses data berurutan dengan kecepatan tinggi. Array list memberikan keunggulan utama berupa akses elemen secara *random access* dengan waktu konstan $O(1)$, sehingga operasi pengambilan data berdasarkan indeks menjadi sangat cepat dan efisien. Selain itu, array list memiliki *cache locality* yang baik, yang berarti elemen-elemen disimpan secara berurutan di memori sehingga pemrosesan data menjadi lebih cepat. Dengan menguasai penggunaan array list, pengembang dapat mengembangkan solusi yang efektif terutama pada aplikasi yang membutuhkan akses cepat ke data dan ukuran data yang relatif tetap. Namun, perlu dipahami bahwa array list memiliki keterbatasan dalam hal fleksibilitas ukuran karena ukurannya bersifat statis setelah dialokasikan. Operasi penyisipan dan penghapusan elemen di tengah array list memerlukan pergeseran elemen-elemen lain itu.

Secara keseluruhan, pemahaman mendalam tentang cara kerja array list dan dampak operasi seperti penambahan, penghapusan, dan akses elemen terhadap performa aplikasi adalah kunci untuk mengoptimalkan penggunaannya. Pengembang harus mempertimbangkan trade-off antara kecepatan akses dan keterbatasan fleksibilitas ukuran untuk memilih solusi yang paling sesuai dengan kebutuhan spesifik program mereka. Dengan demikian, array list tetap menjadi alat yang sangat berharga dan efisien dalam kotak alat pemrograman, terutama untuk aplikasi yang memerlukan akses cepat dan juga terstruktur.

C. Pengertian Pointer dalam Array List dan Bagaimana Cara Menggunakannya

Pointer adalah variabel yang menyimpan alamat memori dari variabel lain. Dalam konteks array, pointer dapat digunakan untuk menunjuk ke elemen tertentu dalam array tersebut. Dalam banyak bahasa pemrograman, seperti C atau C++, nama array itu dapat dianggap sebagai pointer yang menunjuk ke elemen pertama dari array. Misalnya, jika kita memiliki array `arr`, maka `arr` digunakan sebagai pointer ke `arr[0]`. Cara Menggunakan Pointer dalam Array List:

a. Deklarasi pointer

Untuk menggunakan pointer, pertama-tama kita perlu mendeklarasikan pointer yang sesuai dengan tipe data yang akan ditunjuk. Contoh kode:

```
int *ptr; // Pointer untuk tipe data integer
```

b. Menginisialisasi Pointer

Pointer dapat diinisialisasi dengan menggunakan alamat dari elemen array. Contoh kode:

```
int arr[] = {10, 20, 30, 40, 50};
```

```
ptr = arr; // Pointer sekarang menunjuk ke elemen pertama dari arr
```

c. Akses Elemen Menggunakan Pointer

Kita dapat mengakses elemen array menggunakan pointer dengan menggunakan operator dereference. Contoh kode:

```
printf("%d\n", *ptr); // Mengakses elemen pertama (10)
```

d. Iterasi melalui Array

Pointer juga dapat digunakan untuk iterasi melalui array. Contoh kode:

```
for (int i = 0; i < 5; i++) {  
    printf("%d ", *(ptr + i)); // Mengakses setiap elemen dalam array  
}
```

e. Mengubah Nilai Melalui Pointer

```
*(ptr + 2) = 100; // Mengubah elemen ketiga menjadi 100
```

Pointer dalam array list adalah alat yang sangat berguna untuk mengakses dan memanipulasi data. Dengan memahami cara kerja pointer, kita dapat menulis kode yang lebih efisien dan fleksibel. Pastikan untuk selalu berhati-hati dalam menggunakan pointer untuk menghindari kesalahan seperti dereferencing pointer yang tidak valid.

D. Cara Menambah atau Menghapus Elemen dalam Array List, dan Bagaimana Penerapannya dalam Kehidupan Nyata

a. Menambah Elemen dalam Array List

- `push_back()` digunakan untuk menambahkan elemen di akhir `std::vector`.
- `insert()` digunakan untuk menambahkan elemen pada indeks tertentu.

b. Menghapus Elemen:

- `erase()` digunakan untuk menghapus elemen berdasarkan indeks atau nilai.
- Untuk menghapus berdasarkan nilai, kita menggunakan `std::remove` untuk memindahkan elemen yang ingin dihapus ke akhir dan kemudian memanggil `erase()` untuk menghapusnya.

c. Penerapan dalam Kehidupan Nyata

- Manajemen Daftar Belanja:

Kita dapat menggunakan `std::vector` untuk menyimpan daftar belanja, menambah item baru saat berbelanja, dan menghapus item yang sudah dibeli.

- Pengelolaan Kontak:

Dalam aplikasi kontak, kita dapat menggunakan `std::vector` untuk menyimpan nama dan nomor telepon, menambah kontak baru, dan menghapus kontak yang tidak lagi diperlukan.

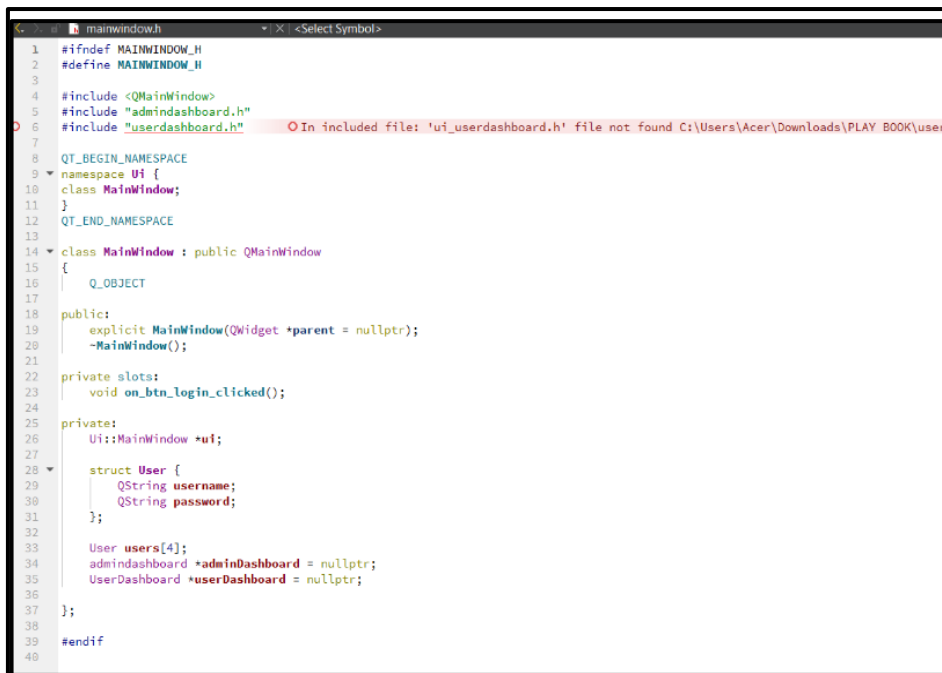
- Sistem Pemesanan:

Dalam sistem pemesanan makanan, std::vector dapat digunakan untuk menyimpan pesanan pelanggan, menambah item pesanan baru, dan menghapus item yang sudah tidak diinginkan.

- **Pengelolaan Tugas:**

Dalam aplikasi manajemen tugas, kita dapat menggunakan std::vector untuk menyimpan daftar tugas, menambah tugas baru, dan menghapus tugas yang sudah selesai.

➤ **Membuat Program dengan Tema SI Play Book dalam Linked List**



```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5  #include "admindashboard.h"
6  #include "userdashboard.h"
7
8  QT_BEGIN_NAMESPACE
9  namespace Ui {
10     class MainWindow;
11 }
12 QT_END_NAMESPACE
13
14 class MainWindow : public QMainWindow
15 {
16     Q_OBJECT
17
18 public:
19     explicit MainWindow(QWidget *parent = nullptr);
20     ~MainWindow();
21
22 private slots:
23     void on_btn_login_clicked();
24
25 private:
26     Ui::MainWindow *ui;
27
28     struct User {
29         QString username;
30         QString password;
31     };
32
33     User users[4];
34     admindashboard *adminDashboard = nullptr;
35     UserDashboard *userDashboard = nullptr;
36
37 };
38
39 #endif
```

Gambar 1 Source Code mainwindow.h

Source Code:

```
#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;

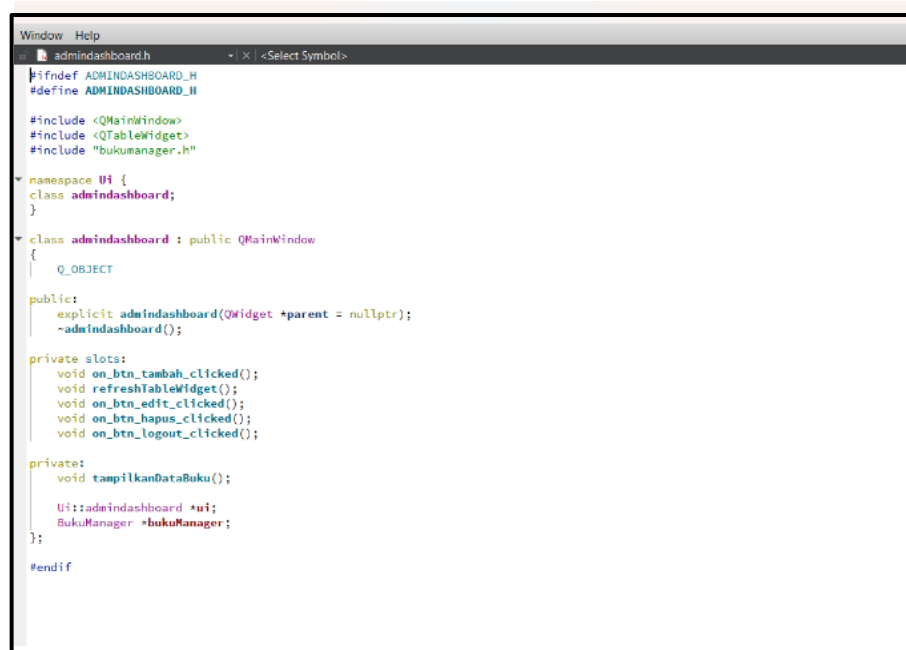
};

#endif
```

Penjelasan Source Code:

Kode di atas merupakan header file `mainwindow.h` dalam proyek berbasis Qt (framework C++ untuk GUI). File ini mendeklarasikan kelas `MainWindow` yang merupakan turunan dari `QMainWindow`, berfungsi sebagai jendela utama aplikasi. Di dalamnya terdapat deklarasi konstruktor dan destruktur serta sebuah slot bernama `on_btn_login_clicked()` yang

kemungkinan besar akan menangani logika ketika tombol login ditekan. Kelas ini juga menyimpan pointer ke UI (**Ui::MainWindow *ui**) yang mengacu pada antarmuka pengguna yang didesain di file .ui, serta dua pointer untuk dashboard: adminDashboard dan userDashboard, yang masing-masing digunakan untuk menampilkan tampilan admin dan pengguna biasa setelah login. Selain itu, terdapat struktur User yang menyimpan pasangan username dan password, dan sebuah array **users[4]** yang berisi data pengguna yang dapat login. Header guard **#ifndef**, **#define**, dan **#endif** digunakan untuk mencegah duplikasi saat file ini disertakan di beberapa file sumber lainnya.



Gambar 2 Source Code admindashboard.h

Source Code:

```

#ifndef ADMINDASHBOARD_H
#define ADMINDASHBOARD_H

#include <QMainWindow>
#include <QTableWidgetItem>
#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 Source Code:

Kode di atas merupakan header file admindashboard.h dalam aplikasi berbasis Qt yang mendefinisikan kelas admindashboard, sebuah turunan dari **QMainWindow**. Kelas ini bertanggung jawab untuk menangani tampilan dan logika dashboard khusus untuk admin, seperti mengelola data buku. Kode ini menyertakan beberapa library penting seperti **QMainWindow**, **QTableWidget**, dan file header bukumanager.h yang kemungkinan berisi logika manajemen data buku. Di dalam kelas terdapat deklarasi konstruktor dan destruktur, serta beberapa slot privat seperti **on_btn_tambah_clicked()**, **on_btn_edit_clicked()**, **on_btn_hapus_clicked()**, dan yang masing-masing menangani aksi tombol tambah, edit, hapus, dan logout. Fungsi **refreshTableWidget()** digunakan untuk memperbarui tampilan tabel data, sementara fungsi privat **tampilkanDataBuku()** kemungkinan bertugas menampilkan seluruh data buku ke antarmuka tabel. Kelas ini juga memiliki pointer ke antarmuka pengguna (**Ui::admindashboard *ui**) dan objek BukuManager ***bukuManager** yang berfungsi sebagai pengelola data buku. Seperti biasa, header guard (**#ifndef**, **#define**, **#endif**) digunakan agar file ini tidak disertakan lebih dari sekali dalam proses kompilasi.

```

1  #ifndef USERDASHBOARD_H
2  #define USERDASHBOARD_H
3
4  #include <QMainWindow>
5  #include "bukumanager.h"
6  #include "ui_userdashboard.h"
7
8  namespace Ui {
9  class userdashboard;
10 }
11
12 class UserDashboard : public QMainWindow
13 {
14     Q_OBJECT
15
16 public:
17     explicit UserDashboard(QWidget *parent = nullptr);
18     ~UserDashboard();
19
20 public slots:
21     void on_btn_cari_clicked();
22     void on_btn_logout_clicked();
23
24 private slots:
25     void on_btn_beli_clicked();
26
27 private:
28     Ui::userdashboard *ui;
29     Bukumanager *bukumanager;
30     void refreshTableWidget();
31     void tampilkanDataBuku();
32 };
33
34 #endif // USERDASHBOARD_H

```

Gambar 3 Source Code userdashboard.h

Source Code:

```

#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 slots:
    void on_btn_beli_clicked();

```



```

private:

    Ui::userdashboard *ui;
    BukuManager *bukuManager;

    void refreshTableWidget();
    void tampilkanDataBuku();

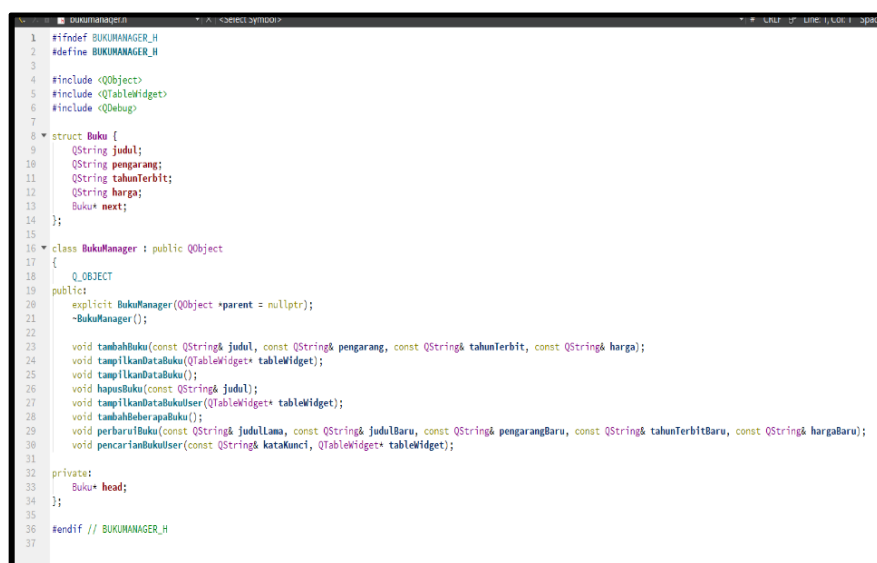
};

#endif // USERDASHBOARD_H

```

Penjelasan Source Code:

Kode di atas merupakan header file userdashboard.h yang mendeklarasikan kelas UserDashboard, yaitu jendela utama untuk pengguna biasa dalam aplikasi berbasis Qt. Kelas ini diturunkan dari QMainWindow, yang merupakan komponen utama dalam pembuatan GUI di Qt. Kelas ini berfungsi untuk menampilkan data buku serta menyediakan fitur interaksi seperti pencarian, pembelian buku, dan logout. Kelas ini memiliki konstruktor dan destruktur untuk inisialisasi dan pembersihan objek. Slot publik `on_btn_cari_clicked()` dan `on_btn_logout_clicked()` digunakan untuk menangani aksi tombol pencarian dan keluar, sementara slot privat `on_btn_beli_clicked()` digunakan saat pengguna ingin membeli buku. Kelas ini juga memiliki dua metode privat, yaitu `refreshTableWidget()` untuk memperbarui tampilan tabel dan `tampilkanDataBuku()` untuk menampilkan daftar buku dari BukuManager, yang merupakan kelas manajemen data buku. Pointer `Ui::userdashboard *ui` menunjuk ke elemen GUI yang dirancang melalui Qt Designer. Penggunaan `#ifndef`, `#define`, dan `#endif` berfungsi sebagai header guard untuk mencegah duplikasi saat file ini dimuat di bagian lain proyek.



```

1  #ifndef BUKUMANAGER_H
2  #define BUKUMANAGER_H
3
4  #include <QObject>
5  #include <QTableWidget>
6  #include <QDebug>
7
8  struct Buku {
9      QString judul;
10     QString pengarang;
11     QString tahunTerbit;
12     QString harga;
13     Buku* next;
14 };
15
16 class BukuManager : public QObject
17 {
18     Q_OBJECT
19 public:
20     explicit BukuManager(QObject *parent = nullptr);
21     ~BukuManager();
22
23     void tambahBuku(const QString& judul, const QString& pengarang, const QString& tahunTerbit, const QString& harga);
24     void tampilkanDataBuku(QTableWidget* tableWidget);
25     void hapusBuku(const QString& judul);
26     void tambahkanDataBuku(QTableWidget* tableWidget);
27     void tambahkanDataBuku();
28     void perbaruiBuku(const QString& judulLama, const QString& judulBaru, const QString& pengarangBaru, const QString& tahunTerbitBaru, const QString& hargaBaru);
29     void pencarianBuku(const QString& kataKunci, QTableWidget* tableWidget);
30
31 private:
32     Buku* head;
33 };
34
35 #endif // BUKUMANAGER_H

```

Gambar 4 Source Code BukuManager.h

Source Code:

```

#ifndef BUKUMANAGER_H
#define BUKUMANAGER_H

#include <QObject>
#include <QWidget>
#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 Source Code:

Kode di atas adalah header file bukumanager.h yang mendefinisikan kelas BukuManager, yaitu kelas yang bertanggung jawab untuk mengelola data buku dalam aplikasi berbasis Qt. Kelas ini menggunakan struktur data linked list, ditandai dengan struct Buku, yang menyimpan informasi buku seperti judul, pengarang, tahun terbit, harga, dan pointer ke buku berikutnya (next). Kelas BukuManager merupakan turunan dari QObject, yang memungkinkan penggunaan mekanisme sinyal dan slot milik Qt jika diperlukan.

Kelas ini menyediakan berbagai metode publik untuk mengelola data buku, seperti **tambahBuku()** untuk menambahkan buku baru, **hapusBuku()** untuk menghapus buku berdasarkan judul, serta **perbaruiBuku()** untuk memperbarui data buku. Metode **tampilkanDataBuku()** dan **tampilkanDataBukuUser()** digunakan untuk menampilkan data buku ke dalam QTableWidgetItem, baik untuk admin maupun pengguna biasa. Ada juga metode **tambahBeberapaBuku()** yang mungkin digunakan untuk mengisi data awal, serta **pencarianBukuUser()** untuk mencari buku berdasarkan kata kunci dan menampilkannya ke tabel. Properti privat head adalah pointer ke buku pertama dalam linked list, yang menjadi titik awal untuk semua operasi traversal data. Seperti biasa, header guard (**#ifndef**, **#define**, **#endif**) digunakan agar file ini tidak dimuat lebih dari sekali selama proses kompilasi.

```
1  #ifndef TAMBAHBUKU_H
2  #define TAMBAHBUKU_H
3
4  #include <QDialog>
5  #include "bukumanager.h"
6
7  namespace Ui {
8  class tambahbuku;
9  }
10
11 class tambahbuku : public QDialog
12 {
13     Q_OBJECT
14
15 public:
16     explicit tambahbuku(BukuManager *bukuManager, QWidget *parent = nullptr);
17     ~tambahbuku();
18
19 private slots:
20     void tambahBuku();
21
22 signals:
23     void dialogClosed();
24
25 private:
26     Ui::tambahbuku *ui;
27     BukuManager *bukuManager;
28 };
29
30 #endif // TAMBAHBUKU_H
31
```

Gambar 5 Source Code tambahbuku.h

Source Code:

```
#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 Source Code:

Kode di atas adalah header file tambahbuku.h yang mendeklarasikan kelas tambahbuku, sebuah turunan dari QDialog dalam aplikasi berbasis Qt. Kelas ini dirancang untuk menampilkan dialog khusus yang memungkinkan admin menambahkan data buku baru ke dalam sistem. Konstruktor tambahbuku menerima pointer ke objek BukuManager sebagai parameter agar dialog ini dapat langsung berinteraksi dengan struktur data buku yang dikelola. Di dalam kelas terdapat slot privat **tambahBuku()**, yaitu fungsi yang akan

dijalankan ketika pengguna menekan tombol untuk menambahkan buku. Terdapat pula **sinyal `dialogClosed()`** yang digunakan untuk memberi tahu komponen lain bahwa dialog telah ditutup—biasanya untuk menyegarkan data di tampilan utama setelah penambahan buku. Pointer **`Ui::tambahbuku *ui`** mengacu pada antarmuka pengguna yang dibuat dengan Qt Designer, sedangkan pointer **`BukuManager *bukuManager`** digunakan untuk menambahkan data ke struktur buku yang ada. Seperti umumnya, header guard **`#ifndef`**, **`#define`**, dan **`#endif`** digunakan untuk mencegah duplikasi pemanggilan file saat proses kompilasi.

```

1  #ifndef EDITBUKU_H
2  #define EDITBUKU_H
3
4  #include <QDialog>
5  #include "bukumanager.h"
6
7  namespace Ui {
8  class editbuku;
9  }
10
11 class editbuku : public QDialog
12 {
13     Q_OBJECT
14
15 public:
16     explicit editbuku(BukuManager *bukuManager, const QString &judulLama, QWidget *parent = nullptr);
17     ~editbuku();
18
19     void setJudul(const QString &judul);
20     void setPengarang(const QString &pengarang);
21     void setTahunTerbit(const QString &tahunTerbit);
22     void setHarga(const QString &harga);
23
24 signals:
25     void dialogClosed();
26
27 private slots:
28     void editBuku();
29
30 private:
31     Ui::editbuku *ui;
32     BukuManager *bukuManager;
33     QString judulLama;
34 };
35
36 #endif // EDITBUKU_H
37

```

Gambar 6 Source Code editbuku.h

Source Code:

`#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 Source Code:

Kode di atas adalah header file `editbuku.h` yang mendeklarasikan kelas `editbuku`, yaitu sebuah dialog berbasis Qt (`QDialog`) yang digunakan untuk mengedit data buku yang sudah ada dalam aplikasi. Kelas ini memiliki konstruktor yang menerima pointer ke objek `BukuManager` serta parameter `judulLama` yang digunakan untuk mengidentifikasi buku yang akan diedit. Terdapat beberapa metode publik seperti `setJudul()`, `setPengarang()`, `setTahunTerbit()`, dan `setHarga()` yang memungkinkan pengisian awal data buku ke dalam antarmuka form edit, biasanya sebelum dialog ditampilkan. Slot privat `editBuku()` akan menangani logika saat pengguna menyimpan perubahan pada data buku. Selain itu, terdapat sinyal `dialogClosed()` yang bisa digunakan untuk memberi tahu tampilan utama bahwa dialog telah ditutup, misalnya agar data dapat diperbarui. Properti privat yang dimiliki kelas ini meliputi pointer ke `Ui::editbuku` (antarmuka pengguna yang didesain melalui Qt Designer), pointer ke `BukuManager` untuk memodifikasi data, serta `QString judulLama` untuk menyimpan judul asli buku yang akan diedit. Header guard `#ifndef`, `#define`, dan `#endif` digunakan agar file ini tidak di-include lebih dari satu kali saat kompilasi.

```

1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include "admindashboard.h"
4 #include "userdashboard.h"
5
6 MainWindow::MainWindow(QWidget *parent) :
7     QMainWindow(parent),
8     ui(new Ui::MainWindow)
9 {
10     // Menginisialisasi komponen UI
11     ui->setupUi(this);
12
13     // Menginisialisasi data pengguna dengan username dan password default
14     users[0].username = "user";
15     users[0].password = "user";
16
17     users[1].username = "user1";
18     users[1].password = "user1";
19
20     users[2].username = "user2";
21     users[2].password = "user2";
22
23     users[3].username = "admin";
24     users[3].password = "admin";
25
26     // Menghubungkan event klik tombol login ke slot yang sesuai
27     connect(ui->btn_login, &QPushButton::clicked, this, &MainWindow::on_btn_login_clicked);
28 }
29
30 // Destruktor untuk kelas MainWindow
31 MainWindow::~MainWindow()
32 {
33     delete ui;
34 }
35
36 // Slot untuk menangani event klik tombol login
37 void MainWindow::on_btn_login_clicked()

```

Gambar 7 Source Code mainwindow.cpp

```

37 void MainWindow::on_btn_login_clicked()
38 {
39     // Mengambil username dan password yang dimasukkan dari field UI
40     QString username = ui->tf_username->text();
41     QString password = ui->tf_password->text();
42
43     // Menginisialisasi flag keberhasilan login
44     bool loginSuccess = false;
45
46     // Iterasi melalui data pengguna untuk mencari username dan password yang cocok
47     for (const User &user : users) {
48         if (user.username == username && user.password == password) {
49             // Memeriksa apakah pengguna yang masuk adalah admin atau pengguna biasa
50             if (username == "admin") {
51                 // Menutup jendela utama dan membuka admin dashboard
52                 this->close();
53                 if (!adminDashboard) {
54                     adminDashboard = new adminDashboard(this);
55                 }
56                 adminDashboard->show();
57             } else {
58                 // Menutup jendela utama dan membuka user dashboard
59                 this->close();
60                 if (!userDashboard) {
61                     userDashboard = new UserDashboard(this);
62                 }
63                 userDashboard->show();
64             }
65             // Mengatur flag keberhasilan login menjadi true dan keluar dari loop
66             loginSuccess = true;
67             break;
68         }
69     }
70
71     // Menampilkan pesan error jika login tidak berhasil
72     if (!loginSuccess) {
73         qDebug() << "Login gagal. Username atau password tidak valid.";
74     }
75 }
76

```

Gambar 8 Lanjutan Source code mainwindow.cpp

Source code :

#include "mainwindow.h"

#include "./ui_mainwindow.h"

#include "admindashboard.h"

#include "userdashboard.h"

```

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    // Menginisialisasi komponen UI
    ui->setupUi(this);

    // Menginisialisasi data pengguna dengan username dan password default
    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";

    // Menghubungkan event klik tombol login ke slot yang sesuai
    connect(ui->btn_login, &QPushButton::clicked, this,
        &MainWindow::on_btn_login_clicked);
}

// Destruktor untuk kelas MainWindow
MainWindow::~MainWindow()
{

```



```

delete ui;

}

// Slot untuk menangani event klik tombol login
void MainWindow::on_btn_login_clicked()
{
    // Mengambil username dan password yang dimasukkan dari field UI
    QString username = ui->tf_username->text();
    QString password = ui->tf_password->text();

    // Menginisialisasi flag keberhasilan login
    bool loginSuccess = false;

    // Iterasi melalui data pengguna untuk mencari username dan password yang cocok
    for (const User &user : users) {
        if (user.username == username && user.password == password) {
            // Memeriksa apakah pengguna yang masuk adalah admin atau pengguna biasa
            if (username == "admin") {
                // Menutup jendela utama dan membuka admin dashboard
                this->close();

                if (!adminDashboard) {
                    adminDashboard = new admindashboard(this);
                }

                adminDashboard->show();
            } else {
                // Menutup jendela utama dan membuka user dashboard
                this->close();
            }
        }
    }
}

```

```

        if (!userDashboard) {

            userDashboard = new UserDashboard(this);

        }

        userDashboard->show();

    }

    // Mengatur flag keberhasilan login menjadi true dan keluar dari loop

    loginSuccess = true;

    break;

}

}

// Menampilkan pesan error jika login tidak berhasil

if (!loginSuccess) {

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

}

}

```

Penjelasan Source code :

Kode di atas merupakan implementasi dari kelas MainWindow, yaitu jendela utama aplikasi yang berfungsi sebagai halaman login. Di dalam konstruktor, UI diinisialisasi menggunakan `setupUi`, dan data pengguna dimasukkan secara hardcoded dengan empat akun: tiga pengguna biasa (`user`, `user1`, `user2`) dan satu admin (`admin`). Tombol login (`btn_login`) dikaitkan dengan slot `on_btn_login_clicked()`, yang akan menangani proses autentikasi. Saat tombol diklik, username dan password dari input pengguna akan dibandingkan dengan data yang ada. Jika cocok, dan username adalah admin, maka jendela utama ditutup dan `adminDashboard` ditampilkan. Jika bukan admin, maka jendela `UserDashboard` yang dibuka. Jika login gagal (tidak ditemukan kecocokan data), maka sistem hanya mencetak pesan kesalahan ke konsol melalui `qDebug()`. Kode ini berfungsi sebagai gerbang kontrol akses, membedakan antara pengguna biasa dan admin untuk mengarahkan mereka ke tampilan yang sesuai.

```

1 #include "admindashboard.h"
2 #include "ui_admindashboard.h"
3 #include "bukumanager.h"
4 #include "tambahbuku.h"
5 #include "QMessageBox"
6 #include "editbuku.h"
7 #include "mainwindow.h"
8
9 admindashboard::admindashboard(QWidget *parent) :
10     QMainWindow(parent),
11     ui(new Ui::admindashboard),
12     bukuManager(new BukuManager)
13 {
14     ui->setupUi(this);
15     tampilkanDataBuku();
16 }
17
18 admindashboard::~admindashboard()
19 {
20     delete bukuManager;
21     delete ui;
22 }
23
24 void admindashboard::on_btn_tambah_clicked()
25 {
26     tambahbuku *dialog = new tambahbuku(bukuManager, this);
27     dialog->exec();
28     connect(dialog, &tambahbuku::dialogClosed, this, &admindashboard::refreshTableWidget);
29 }
30
31 void admindashboard::refreshTableWidget()
32 {
33     tampilkanDataBuku();
34 }
35
36 void admindashboard::tampilkanDataBuku()
37 {
38     bukuManager->tampilkanDataBuku(ui->tableWidget);
39 }

```

Gambar 9 Source Code admindashboard.cpp

```

40 void admindashboard::on_btn_edit_clicked()
41 {
42     QModelIndexList selectedIndexes = ui->tableWidget->selectionModel()->selectedRows();
43
44     if(selectedIndexes.isEmpty()) {
45         QMessageBox::warning(this, "Peringatan", "Harap pilih data terlebih dahulu.");
46         return;
47     }
48
49     int selectedRow = selectedIndexes.at(0).row();
50
51     QTableWidgetItem judulItem = ui->tableWidget->item(selectedRow, 0);
52     QTableWidgetItem pengarangItem = ui->tableWidget->item(selectedRow, 1);
53     QTableWidgetItem tahunTerbitItem = ui->tableWidget->item(selectedRow, 2);
54     QTableWidgetItem hargaItem = ui->tableWidget->item(selectedRow, 3);
55
56     QString judul = judulItem->text();
57     QString pengarang = pengarangItem->text();
58     QString tahunTerbit = tahunTerbitItem->text();
59     QString harga = hargaItem->text();
60
61     editbuku *dialog = new editbuku(bukuManager, judul, this);
62     dialog->setJudul(judul);
63     dialog->setPengarang(pengarang);
64     dialog->setTahunTerbit(tahunTerbit);
65     dialog->setHarga(harga);
66     dialog->exec();
67 }
68
69 void admindashboard::on_btn_hapus_clicked()
70 {
71     QModelIndexList selectedIndexes = ui->tableWidget->selectionModel()->selectedRows();
72
73     if(selectedIndexes.isEmpty()) {
74         QMessageBox::warning(this, "Peringatan", "Harap pilih data terlebih dahulu.");
75         return;
76     }
77
78     int choice = QMessageBox::question(this, "Konfirmasi Hapus", "Anda yakin ingin menghapus buku?", QMessageBox::Yes | QMessageBox::No);
79     if(choice == QMessageBox::No)
80         return;

```

Gambar 10 Source Code admindashboard.cpp

```

81 void admindashboard::on_btn_hapus_clicked()
82 {
83     QModelIndexList selectedIndexes = ui->tableWidget->selectionModel()->selectedRows();
84
85     if(selectedIndexes.isEmpty()) {
86         QMessageBox::warning(this, "Peringatan", "Harap pilih data terlebih dahulu.");
87         return;
88     }
89
90     int choice = QMessageBox::question(this, "Konfirmasi Hapus", "Anda yakin ingin menghapus buku?", QMessageBox::Yes | QMessageBox::No);
91     if(choice == QMessageBox::No)
92         return;
93
94     int selectedRow = selectedIndexes.at(0).row();
95     QTableWidgetItem judulItem = ui->tableWidget->item(selectedRow, 0);
96     QString judul = judulItem->text();
97     bukuManager->hapusBuku(judul);
98
99     ui->tableWidget->removeRow(selectedRow);
100
101     QMessageBox::information(this, "Hapus Buku", "Buku berhasil dihapus.");
102
103     qDebug() << "Isi linked list setelah penghapusan:";
104     bukuManager->tampilkanDataBuku();
105 }
106
107 void admindashboard::on_btn_logout_clicked()
108 {
109     this->close();
110     this->close();
111
112     MainWindow *mainwindow = new MainWindow();
113     mainwindow->show();
114 }

```

Gambar 11 Source Code admindashboard.cpp

Source Code :

```

#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 Source Code :

Kode adminDashboard adalah implementasi tampilan untuk admin dalam aplikasi manajemen buku, yang memungkinkan admin untuk melihat, menambah, mengedit, dan menghapus data buku. Pada saat dashboard diinisialisasi, UI disiapkan dan data buku ditampilkan dalam tabel menggunakan BukuManager. Saat tombol Tambah ditekan, dialog tambahbuku ditampilkan, dan setelah ditutup, tabel akan diperbarui. Tombol Edit

memungkinkan admin memilih satu baris buku, lalu membuka dialog editbuku untuk mengubah data tersebut. Tombol Hapus akan meminta konfirmasi sebelum menghapus buku yang dipilih dari daftar dan tabel. Terakhir, tombol Logout menutup dashboard admin dan mengembalikan pengguna ke halaman login utama. Semua interaksi berbasis UI ini memanfaatkan kelas QTableWidgetItem, QMessageBox, dan koneksi sinyal-slot khas dari Qt untuk mengatur alur kerja secara dinamis.

```

1 #include "userdashboard.h"
2 #include "bukumanager.h"
3 #include <QMainWindow>
4 #include <QTableWidgetItem>
5 #include <QMessageBox>
6 #include "mainwindow.h"
7
8 // Konstruktor untuk kelas UserDashboard
9 UserDashboard::UserDashboard(QWidget *parent) :
10     QMainWindow(parent),
11     ui(new Ui::UserDashboard),
12     bukuManager(new BukuManager())
13 {
14     // Menginisialisasi UI
15     ui->setupUi(this);
16
17     // Memperbarui data buku yang ditampilkan dalam tabel
18     refreshTableWidget();
19
20     // Menghubungkan event klik tombol cari ke slot yang sesuai
21     connect(ui->btn_cari, &QPushButton::clicked, this, &UserDashboard::on_btn_cari_clicked);
22 }
23
24 // Destruktor untuk kelas UserDashboard
25 UserDashboard::~UserDashboard()
26 {
27     delete ui;
28     delete bukuManager;
29 }
30
31 // Metode untuk memperbarui data buku yang ditampilkan dalam tabel
32 void UserDashboard::refreshTableWidget()
33 {
34     bukuManager->tampilkanDataBukuUser(ui->tableWidget);
35 }
36
37 // Slot untuk menangani event klik tombol cari
38 void UserDashboard::on_btn_cari_clicked()
39 {
40     // Mengambil kata kunci pencarian dari input pengguna
41     QString kataKunci = ui->tf_cari->text();
42
43     // Melakukan pencarian buku berdasarkan kata kunci dan menampilkan hasilnya dalam tabel
44     bukuManager->pencarianBukuUser(kataKunci, ui->tableWidget);
45 }

```

Gambar 12 Source Code userdashboard.cpp

```

// Melakukan pencarian buku berdasarkan kata kunci dan menampilkan hasilnya dalam tabel
bukuManager->pencarianBukuUser(kataKunci, ui->tableWidget);

// Slot untuk menangani event klik tombol Logout
void UserDashboard::on_btn_logout_clicked()
{
    // Menutup jendela dashboard pengguna
    this->close();

    // Membuat dan menampilkan jendela utama untuk login
    QMainWindow *loginWindow = new QMainWindow();
    loginWindow->show();
    this->close();
}

void UserDashboard::on_btn_hapus_clicked()
{
    QModelIndex selectedIndexes = ui->tableWidget->selectionModel()->selectedIndexes();
    if(selectedIndexes.isEmpty()) {
        QMessageBox::warning(this, "Peringatan", "Harap pilih data terlebih dahulu.");
        return;
    }

    int choice = QMessageBox::question(this, "Konfirmasi Hapus", "Apakah anda yakin ingin menghapus buku ini?", QMessageBox::Yes | QMessageBox::No);
    if(choice == QMessageBox::Yes) {
        return;
    }

    int selectedRow = selectedIndexes.at(0).row();
    QTableWidgetItem *judulBaris = ui->tableWidget->item(selectedRow, 0);
    QString judul = judulBaris->text();
    bukuManager->hapusBuku(judul);

    ui->tableWidget->removeRow(selectedRow);

    QMessageBox::information(this, "Buku Terhapus", "Buku berhasil dihapus!");
    qDebug() << "Tel selesai proses setelah penulisan!";
    bukuManager->tampilkanDataBuku();
}

```

Gambar 13 Source Code userdashboard.cpp

Source Code :

#include "userdashboard.h"

#include "bukumanager.h"

#include <QMainWindow>

#include <QTableWidgetItem>

#include <QMessageBox>

```

#include "mainwindow.h"

// Konstruktor untuk kelas UserDashboard
UserDashboard::UserDashboard(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::userdashboard),
    bukuManager(new BukuManager())
{
    // Menginisialisasi UI
    ui->setupUi(this);

    // Memperbarui data buku yang ditampilkan dalam tabel
    refreshTableWidget();

    // Menghubungkan event klik tombol cari ke slot yang sesuai
    connect(ui->btn_cari, &QPushButton::clicked, this,
        &UserDashboard::on_btn_cari_clicked);
}

// Destruktor untuk kelas UserDashboard
UserDashboard::~UserDashboard()
{
    delete ui;
    delete bukuManager;
}

// Metode untuk memperbarui data buku yang ditampilkan dalam tabel
void UserDashboard::refreshTableWidget()
{
    bukuManager->tampilkanDataBukuUser(ui->tableWidget);
}

// Slot untuk menangani event klik tombol cari
void UserDashboard::on_btn_cari_clicked()

```



```

{
    // Mengambil kata kunci pencarian dari input pengguna
    QString kataKunci = ui->tf_cari->text();

    // Melakukan pencarian buku berdasarkan kata kunci dan menampilkan hasilnya dalam
    tabel
    bukuManager->pencarianBukuUser(kataKunci, ui->tableWidget);
}

// Slot untuk menangani event klik tombol logout
void UserDashboard::on_btn_logout_clicked()
{
    // Menutup jendela dashboard pengguna
    this->close();

    // Membuat dan menampilkan jendela utama untuk login
    MainWindow *mainWindow = new MainWindow();

    mainWindow->show();
    this->close();
}

void UserDashboard::on_btn_beli_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 Pembelian", "Apakah anda yakin
    ingin membeli buku ini?", 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, "Buku Terbeli", "Buku berhasil dibeli!");

qDebug() << "Isi linked list setelah pembelian:";
bukuManager->tampilkanDataBuku();

}

```

Penjelasan Source Code :

Kode UserDashboard merupakan bagian dari sistem manajemen buku berbasis GUI yang dirancang dengan Qt. Kelas ini mewakili tampilan dan fungsionalitas untuk pengguna biasa setelah berhasil login. Saat objek UserDashboard dibuat, konstruktor menginisialisasi antarmuka pengguna dengan `setupUi(this)` dan langsung memanggil fungsi `refreshTableWidget()` untuk menampilkan seluruh data buku yang tersedia dalam bentuk tabel menggunakan metode `tampilkanDataBukuUser()` dari objek BukuManager. Objek BukuManager sendiri bertanggung jawab mengelola data buku, seperti menyimpan, menampilkan, mencari, dan menghapus buku. Selain itu, tombol Cari dihubungkan ke slot `on_btn_cari_clicked()` yang berfungsi mengambil teks pencarian dari input `tf_cari`, kemudian mencocokkannya dengan data buku menggunakan metode `pencarianBukuUser()`, dan menampilkan hasilnya di tabel. Tombol Beli memungkinkan pengguna membeli buku dengan cara memilih baris dalam tabel, lalu sistem akan menampilkan dialog konfirmasi. Jika pengguna menyetujui, buku yang dipilih akan dihapus dari struktur data internal melalui `hapusBuku()` dan juga dihapus dari tampilan tabel. Setelah pembelian berhasil, muncul pesan informasi, dan isi linked list (**tempat penyimpanan data buku**) dicetak ke konsol untuk debugging. Tombol Logout mengarahkan pengguna kembali ke halaman login (**MainWindow**) dengan menutup UserDashboard dan menampilkan jendela utama baru. Keseluruhan fungsionalitas ini menggunakan pendekatan event-driven programming dari Qt, dengan mekanisme sinyal dan slot, serta pengelolaan UI secara langsung menggunakan pointer terhadap komponen-komponen antarmuka.

```

1 #include "bukumanager.h"
2
3 BukuManager::BukuManager(QObject *parent) : QObject(parent), head(nullptr) {
4     tambahBeberapaBuku();
5 }
6
7 // Destructur untuk kelas BukuManager
8 BukuManager::~BukuManager()
9 {
10     Buku* current = head;
11     while (current) {
12         Buku* next = current->next;
13         delete current;
14         current = next;
15     }
16 }
17
18 // Metode untuk menambahkan beberapa buku ke dalam manajer buku
19 void BukuManager::tambahBeberapaBuku() {
20     tambahBuku("Solo Leveling", "Jang Sung-Rak", "2018", "50000");
21     tambahBuku("The Beginning After the End", "TurtleMe", "2020", "100000");
22     tambahBuku("The Boxer", "Jung Si-Moon", "2019", "75000");
23     tambahBuku("Lookism", "PTT", "2024", "150000");
24     tambahBuku("Detective Conan", "Gosho Aoyama", "1994", "100000");
25     tambahBuku("Wind Breaker", "Yongseok Jo", "2013", "100000");
26     tambahBuku("The World After The Fall", "Undead Sanja", "2022", "50000");
27 }
28
29 // Metode untuk menambahkan buku baru ke dalam manajer buku
30 void BukuManager::tambahBuku(const QString& judul, const QString& pengarang, const QString& tahunTerbit, const QString& harga)
31 {
32     // Membuat node buku baru
33     Buku* newBuku = new Buku;
34     newBuku->judul = judul;
35     newBuku->pengarang = pengarang;
36     newBuku->tahunTerbit = tahunTerbit;
37     newBuku->harga = harga;
38     newBuku->next = nullptr;
39
40     // Menambahkan buku baru ke dalam daftar buku
41     if (!head) {
42         head = newBuku;
43     }
44     else {
45         Buku* current = head;
46         while (current->next) {
47             current = current->next;
48         }
49         current->next = newBuku;
50     }
51 }
52
53 // Metode untuk menampilkan data buku dalam tabel
54 void BukuManager::tampilkanDataBuku(QTableWidget* tableWidget)
55 {
56     tableWidget->clearContents();
57     tableWidget->setRowCount(0);
58     Buku* current = head;
59     int row = 0;
60 }

```

Gambar 14 Source Code bukumanager.cpp

```

61 // Buku current = head;
62 int row = 0;
63 while (current) {
64     tableWidget->insertRow(row);
65     tableWidget->setItem(row, 0, new QTableWidgetItem(current->judul));
66     tableWidget->setItem(row, 1, new QTableWidgetItem(current->pengarang));
67     tableWidget->setItem(row, 2, new QTableWidgetItem(current->tahunTerbit));
68     tableWidget->setItem(row, 3, new QTableWidgetItem(current->harga));
69     current = current->next;
70     ++row;
71 }
72
73 // Metode untuk menampilkan data buku dalam tabel
74 void BukuManager::tampilkanDataBuku()
75 {
76     Buku* current = head;
77     while (current) {
78         qDebug() << "Judul" << current->judul << ", Pengarang" << current->pengarang << ", Tahun Terbit" << current->tahunTerbit << ", Harga" << current->harga;
79         current = current->next;
80     }
81 }
82
83 // Metode untuk menghapus informasi buku
84 void BukuManager::hapusBuku(const QString& judulHapus, const QString& judulBaru, const QString& pengarangBaru, const QString& tahunTerbitBaru, const QString& hargaBaru)
85 {
86     Buku* current = head;
87     while (current) {
88         if (current->judul == judulHapus) {
89             current->judul = judulBaru;
90             current->pengarang = pengarangBaru;
91             current->tahunTerbit = tahunTerbitBaru;
92             current->harga = hargaBaru;
93             break;
94         }
95         current = current->next;
96     }
97 }
98
99 // Metode untuk menghapus buku dari daftar berdasarkan judul
100 void BukuManager::hapusBuku(const QString& judul)
101 {
102     if (!head) {
103         qDebug() << "Tidak ada buku dalam daftar.";
104         return;
105     }
106     if (head->judul == judul) {
107         Buku* temp = head;
108         head = head->next;
109         delete temp;
110         qDebug() << "Buku dengan judul" << judul << "telah dihapus.";
111         return;
112     }
113     Buku* current = head;
114     while (current->next) {
115         if (current->next->judul == judul) {
116             Buku* temp = current->next;
117             current->next = current->next->next;
118             delete temp;
119             qDebug() << "Buku dengan judul" << judul << "telah dihapus.";
120             return;
121         }
122         current = current->next;
123     }
124     qDebug() << "Buku dengan judul" << judul << "tidak ditemukan dalam daftar.";
125 }
126
127 // Metode untuk menampilkan data buku dalam tabel untuk pengguna
128 void BukuManager::tampilkanDataBukuUser(QTableWidget* tableWidget)
129 {
130     tableWidget->clearContents();
131     tableWidget->setRowCount(0);
132
133     Buku* current = head;
134     int row = 0;
135     while (current) {
136         tableWidget->insertRow(row);
137         tableWidget->setItem(row, 0, new QTableWidgetItem(current->judul));
138         tableWidget->setItem(row, 1, new QTableWidgetItem(current->harga));
139         current = current->next;
140         ++row;
141     }
142 }
143
144 // Metode untuk melakukan pencarian buku berdasarkan kata kunci untuk pengguna
145 void BukuManager::pencarianBukuUser(const QString& kataKunci, QTableWidget* tableWidget)
146 {
147     // Iterasi melalui semua baris dan kolom dalam tabel
148     for (int row = 0; row < tableWidget->rowCount(); ++row) {
149         for (int col = 0; col < tableWidget->columnCount(); ++col) {
150             // Mendapatkan item di setiap sel
151             QTableWidgetItem* item = tableWidget->item(row, col);
152             // Memeriksa apakah item mengandung kata kunci (tidak case sensitive)
153             if (item && item->text().contains(kataKunci, Qt::CaseInsensitive)) {
154                 tableWidget->setRowHidden(row, false);
155                 break;
156             }
157             else {
158                 tableWidget->setRowHidden(row, true);
159             }
160         }
161     }
162 }

```

Gambar 15 Source Code bukumanager.cpp

```

153
154 Buku* current = head;
155 while (current->next) {
156     if (current->next->judul == judul) {
157         Buku* temp = current->next;
158         current->next = current->next->next;
159         delete temp;
160         qDebug() << "Buku dengan judul" << judul << "telah dihapus.";
161         return;
162     }
163     current = current->next;
164 }
165
166 qDebug() << "Buku dengan judul" << judul << "tidak ditemukan dalam daftar.";
167 }
168
169 // Metode untuk menampilkan data buku dalam tabel untuk pengguna
170 void BukuManager::tampilkanDataBukuUser(QTableWidget* tableWidget)
171 {
172     tableWidget->clearContents();
173     tableWidget->setRowCount(0);
174
175     Buku* current = head;
176     int row = 0;
177     while (current) {
178         tableWidget->insertRow(row);
179         tableWidget->setItem(row, 0, new QTableWidgetItem(current->judul));
180         tableWidget->setItem(row, 1, new QTableWidgetItem(current->harga));
181         current = current->next;
182         ++row;
183     }
184 }
185
186 // Metode untuk melakukan pencarian buku berdasarkan kata kunci untuk pengguna
187 void BukuManager::pencarianBukuUser(const QString& kataKunci, QTableWidget* tableWidget)
188 {
189     // Iterasi melalui semua baris dan kolom dalam tabel
190     for (int row = 0; row < tableWidget->rowCount(); ++row) {
191         for (int col = 0; col < tableWidget->columnCount(); ++col) {
192             // Mendapatkan item di setiap sel
193             QTableWidgetItem* item = tableWidget->item(row, col);
194             // Memeriksa apakah item mengandung kata kunci (tidak case sensitive)
195             if (item && item->text().contains(kataKunci, Qt::CaseInsensitive)) {
196                 tableWidget->setRowHidden(row, false);
197                 break;
198             }
199             else {
200                 tableWidget->setRowHidden(row, true);
201             }
202         }
203     }
204 }

```

Gambar 16 Source Code bukumanager.cpp

Source Code :

```
#include "bukumanager.h"

BukuManager::BukuManager(QObject *parent) : QObject(parent), head(nullptr) {
    tambahBeberapaBuku();
}

// Destruktor untuk kelas BukuManager
BukuManager::~BukuManager()
{
    Buku* current = head;
    while (current) {
        Buku* next = current->next;
        delete current;
        current = next;
    }
}

// Metode untuk menambahkan beberapa buku ke dalam manajer buku
void BukuManager::tambahBeberapaBuku() {
    tambahBuku("Solo Leveling", "Jang Sung-Rak", "2018", "50000");
    tambahBuku("The Beginning after the end", "TurtleMe", "2018", "100000");
    tambahBuku("The Boxer", "Jung Ji-Hoon", "2019", "75000");
    tambahBuku("Lookism", "PTJ", "2014", "150000");
    tambahBuku("Detective Conan", "Gosho Aoyama", "1994", "100000");
    tambahBuku("Wind Breaker", "Yongseok Jo", "2013", "150000");
    tambahBuku("The World After The Fall", "Undead Gamja", "2022", "50000");
}

// Metode untuk menambahkan buku baru ke dalam manajer buku
void BukuManager::tambahBuku(const QString& judul, const QString& pengarang,
const QString& tahunTerbit, const QString& harga)
{

```

```

// Membuat node buku baru

Buku* newBuku = new Buku;
newBuku->judul = judul;
newBuku->pengarang = pengarang;
newBuku->tahunTerbit = tahunTerbit;
newBuku->harga = harga;
newBuku->next = nullptr;

// Menambahkan buku baru ke dalam daftar buku
if (!head) {
    head = newBuku;
} else {
    Buku* current = head;
    while (current->next) {
        current = current->next;
    }
    current->next = newBuku;
}

// Metode untuk menampilkan data buku dalam tabel
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;
    }
}

// Metode untuk menampilkan data buku dalam konsol
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;
    }
}

// Metode untuk memperbarui informasi buku
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;

    }

}

// Metode untuk menghapus buku dari daftar berdasarkan judul
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.";
    }

// Metode untuk menampilkan data buku dalam tabel untuk pengguna
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;
    }
}

// Metode untuk melakukan pencarian buku berdasarkan kata kunci untuk pengguna
void BukuManager::pencarianBukuUser(const QString& kataKunci, QTableWidgetItem*
tableWidget)
{
    // Iterasi melalui semua baris dan kolom dalam tabel
    for (int row = 0; row < tableWidget->rowCount(); ++row) {
        for (int col = 0; col < tableWidget->columnCount(); ++col) {
            // Mendapatkan item di setiap sel
            QTableWidgetItem* item = tableWidget->item(row, col);
            // Memeriksa apakah item mengandung kata kunci (tidak case sensitive)
            if (item && item->text().contains(kataKunci, Qt::CaseInsensitive)) {
                tableWidget->setRowHidden(row, false);
            }
        }
    }
}

```



```

        break;
    } else {
        tableWidget->setRowHidden(row, true);
    }
}
}
}

```

Penjelasan Source Code :

Kode di atas merupakan implementasi dari kelas BukuManager, yang berperan sebagai pengelola utama data buku dalam aplikasi manajemen buku berbasis Qt. Kelas ini menggunakan struktur data *linked list* untuk menyimpan informasi setiap buku, yang terdiri dari judul, pengarang, tahun terbit, dan harga. Pada saat inisialisasi (**BukuManager()**), beberapa buku langsung ditambahkan ke dalam daftar melalui metode **tambahBeberapaBuku()**. Setiap buku baru ditambahkan ke akhir daftar dengan **tambahBuku()**. Informasi buku dapat ditampilkan ke dalam tabel GUI melalui **tampilkanDataBuku()** untuk admin (menampilkan seluruh data), dan **tampilkanDataBukuUser()** untuk pengguna (hanya menampilkan judul dan harga). Selain itu, pengguna dapat memperbarui informasi buku yang sudah ada dengan **perbaruiBuku()**, menghapus buku berdasarkan judul dengan **hapusBuku()**, serta mencari buku dengan kata kunci menggunakan **pencarianBukuUser()**, yang menyaring baris dalam QTableWidgetItem berdasarkan kecocokan teks. Destruktor kelas akan secara otomatis membebaskan memori dari seluruh node dalam linked list. Seluruh metode bekerja secara terintegrasi dengan komponen UI Qt, memungkinkan admin dan pengguna untuk mengelola data buku secara interaktif melalui antarmuka grafis.

```

1 #include "tambahbuku.h"
2 #include "ui_tambahbuku.h"
3 #include "QMessageBox"
4 #include "QDialogButtonBox"
5
6 // Konstruktor untuk kelas tambahbuku
7 tambahbuku::tambahbuku(BukuManager *bukuManager, QWidget *parent) :
8     QDialog(parent),
9     ui(new Ui::tambahbuku),
10    bukuManager(bukuManager)
11 {
12     // Menginisialisasi UI
13     ui->setupUi(this);
14 }
15
16 // Destruktor untuk kelas tambahbuku
17 tambahbuku::~tambahbuku()
18 {
19     delete ui;
20     // Mengirim sinyal bahwa dialog telah ditutup
21     emit dialogClosed();
22 }
23
24 // Metode untuk menambahkan buku baru
25 void tambahbuku::tambahbuku() {
26     // Memeriksa apakah semua kolom diisi
27     if (ui->tf_judul->text().isEmpty() || ui->tf_pengarang->text().isEmpty() || ui->tf_tahun->text().isEmpty() || ui->tf_harga->text().isEmpty()) {
28         // Menampilkan pesan peringatan jika ada kolom yang kosong
29         QMessageBox::warning(this, "Peringatan", "Mohon lengkapi semua kolom.");
30         return;
31     }
32     // Mengambil informasi buku dari input pengguna
33     QString judul = ui->tf_judul->text();
34     QString pengarang = ui->tf_pengarang->text();
35     QString tahunTerbit = ui->tf_tahun->text();
36     QString harga = ui->tf_harga->text();
37
38     // Menambahkan buku baru ke dalam manajer buku
39     bukuManager->tambahbuku(judul, pengarang, tahunTerbit, harga);
40
41     // Menampilkan pesan informasi bahwa buku berhasil ditambahkan
42     QMessageBox::information(this, "Informasi", "Buku berhasil ditambahkan ke dalam daftar.");
43
44     // Menutup dialog
45     this->close();
46 }
47

```

Gambar 17 Source Code tambahbuku.cpp

Source Code :

```
#include "tambahbuku.h"

#include "ui_tambahbuku.h"

#include "QMessageBox"

#include "QDialogButtonBox"


// Konstruktor untuk kelas tambahbuku

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

    QDialog(parent),

    ui(new Ui::tambahbuku),

    bukuManager(bukuManager)

{

    // Menginisialisasi UI

    ui->setupUi(this);

}


// Destruktor untuk kelas tambahbuku

tambahbuku::~tambahbuku()

{

    delete ui;

    // Mengirim sinyal bahwa dialog telah ditutup

    emit dialogClosed();

}


// Metode untuk menambahkan buku baru

void tambahbuku::tambahBuku() {

    // Memeriksa apakah semua kolom diisi
```

```

        if (ui->tf_judul->text().isEmpty() || ui->tf_pengarang->text().isEmpty() || ui->tf_tahun->text().isEmpty() || ui->tf_harga->text().isEmpty()) {

            // Menampilkan pesan peringatan jika ada kolom yang kosong

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

            return;

        }

        // Mengambil informasi buku dari input pengguna

        QString judul = ui->tf_judul->text();

        QString pengarang = ui->tf_pengarang->text();

        QString tahunTerbit = ui->tf_tahun->text();

        QString harga = ui->tf_harga->text();


        // Menambahkan buku baru ke dalam manajer buku

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


        // Menampilkan pesan informasi bahwa buku berhasil ditambahkan

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


        // Menutup dialog

        this->close();

    }

```

Penjelasan source code :

Kode di atas adalah implementasi kelas tambahbuku, yang merupakan dialog antarmuka grafis (*QDialog*) dalam aplikasi manajemen buku berbasis Qt. Kelas ini digunakan khusus oleh admin untuk menambahkan data buku baru ke sistem. Ketika objek tambahbuku dibuat, konstruktor tambahbuku::tambahbuku menerima pointer ke objek BukuManager agar dapat langsung menambahkan data ke daftar buku yang dikelola oleh kelas tersebut. Antarmuka pengguna diinisialisasi dengan **setupUi(this)**, yang memuat elemen-elemen input seperti kolom teks untuk judul, pengarang, tahun terbit, dan harga

buku. Fungsi utama dalam kelas ini adalah **tambahBuku()**, yang bertanggung jawab mengekstrak data dari field input, memverifikasi bahwa semua kolom telah diisi, lalu memanggil **bukuManager->tambahBuku()** untuk menyimpan data buku baru ke dalam *linked list* buku. Jika salah satu field kosong, akan muncul pesan peringatan menggunakan **QMessageBox**. Setelah buku berhasil ditambahkan, dialog ditutup dan pesan konfirmasi akan ditampilkan kepada pengguna. Destruktor kelas **~tambahbuku()** juga mengirimkan sinyal **dialogClosed()** untuk memberi tahu antarmuka utama bahwa dialog telah selesai digunakan, agar tampilan data bisa diperbarui. Kode ini memungkinkan admin menambah buku secara interaktif dan efisien melalui jendela dialog.

```

1  #include "editbuku.h"
2  #include "ui_editbuku.h"
3  #include <QMessageBox>
4
5  // Konstruktor untuk kelas editbuku
6  editbuku::editbuku(BukuManager *bukuManager, const QString &judulLama, QWidget *parent) :
7      QDialog(parent),
8      ui(new Ui::editbuku),
9      bukuManager(bukuManager),
10     judulLama(judulLama)
11 {
12     // Menginisialisasi UI
13     ui->setupUi(this);
14     // Menghubungkan event klik tombol edit ke slot yang sesuai
15     connect(ui->btn_edit, &QPushButton::clicked, this, &editbuku::editBuku);
16 }
17
18 // Destruktor untuk kelas editbuku
19 editbuku::~editbuku()
20 {
21     delete ui;
22 }
23
24 // Metode untuk mengatur judul buku pada UI
25 void editbuku::setJudul(const QString &judul)
26 {
27     ui->tf_judul->setText(judul);
28 }
29
30 // Metode untuk mengatur pengarang buku pada UI
31 void editbuku::setPengarang(const QString &pengarang)
32 {
33     ui->tf_pengarang->setText(pengarang);
34 }
35
36 // Metode untuk mengatur tahun terbit buku pada UI
37 void editbuku::setTahunTerbit(const QString &tahunTerbit)
38 {
39     ui->tf_tahun->setText(tahunTerbit);
40 }
41
42 // Metode untuk mengatur harga buku pada UI
43 void editbuku::setHarga(const QString &harga)
44 {

```

Gambar 18 Source Code editbuku.cpp

```

45     ui->tf_harga->setText(harga);
46 }
47
48 // Metode untuk melakukan pengeditan informasi buku
49 void editbuku::editBuku()
50 {
51     // Mengambil informasi buku yang baru dari input pengguna
52     QString judulBaru = ui->tf_judul->text();
53     QString pengarangBaru = ui->tf_pengarang->text();
54     QString tahunTerbitBaru = ui->tf_tahun->text();
55     QString hargaBaru = ui->tf_harga->text();
56
57     // Memperbarui informasi buku di dalam buku manager
58     bukuManager->perbaruiBuku(judulLama, judulBaru, pengarangBaru, tahunTerbitBaru, hargaBaru);
59
60     // Menampilkan pesan informasi bahwa buku berhasil diperbarui
61     QMessageBox::information(this, "Edit Buku", "Buku berhasil diperbarui.");
62
63     // Mengirim sinyal bahwa dialog telah ditutup
64     emit dialogClosed();
65     // Menutup dialog
66     this->close();
67 }
68

```

Gambar 19 Source Code editbuku.cpp

Source code :

```
#include "editbuku.h"

#include "ui_editbuku.h"

#include <QMessageBox>

// Konstruktor untuk kelas editbuku
editbuku::editbuku(BukuManager *bukuManager, const QString &judulLama, QWidget
*parent) :
    QDialog(parent),
    ui(new Ui::editbuku),
    bukuManager(bukuManager),
    judulLama(judulLama)
{
    // Menginisialisasi UI
    ui->setupUi(this);

    // Menghubungkan event klik tombol edit ke slot yang sesuai
    connect(ui->btn_edit, &QPushButton::clicked, this, &editbuku::editBuku);
}

// Destruktor untuk kelas editbuku
editbuku::~editbuku()
{
    delete ui;
}

// Metode untuk mengatur judul buku pada UI
void editbuku::setJudul(const QString &judul)
{
    ui->tf_judul->setText(judul);
}

// Metode untuk mengatur pengarang buku pada UI
```

```

void editbuku::setPengarang(const QString &pengarang)
{
    ui->tf_pengarang->setText(pengarang);
}

// Metode untuk mengatur tahun terbit buku pada UI
void editbuku::setTahunTerbit(const QString &tahunTerbit)
{
    ui->tf_tahun->setText(tahunTerbit);
}

// Metode untuk mengatur harga buku pada UI
void editbuku::setHarga(const QString &harga)
{
    ui->tf_harga->setText(harga);
}

// Metode untuk melakukan pengeditan informasi buku
void editbuku::editBuku()
{
    // Mengambil informasi buku yang baru dari input pengguna
    QString judulBaru = ui->tf_judul->text();
    QString pengarangBaru = ui->tf_pengarang->text();
    QString tahunTerbitBaru = ui->tf_tahun->text();
    QString hargaBaru = ui->tf_harga->text();

    // Memperbarui informasi buku di dalam buku manager
    bukuManager->perbaruiBuku(judulLama, judulBaru, pengarangBaru, tahunTerbitBaru,
    hargaBaru);

    // Menampilkan pesan informasi bahwa buku berhasil diperbarui
    QMessageBox::information(this, "Edit Buku", "Buku berhasil diperbarui.");

    // Mengirim sinyal bahwa dialog telah ditutup
    emit dialogClosed();
}

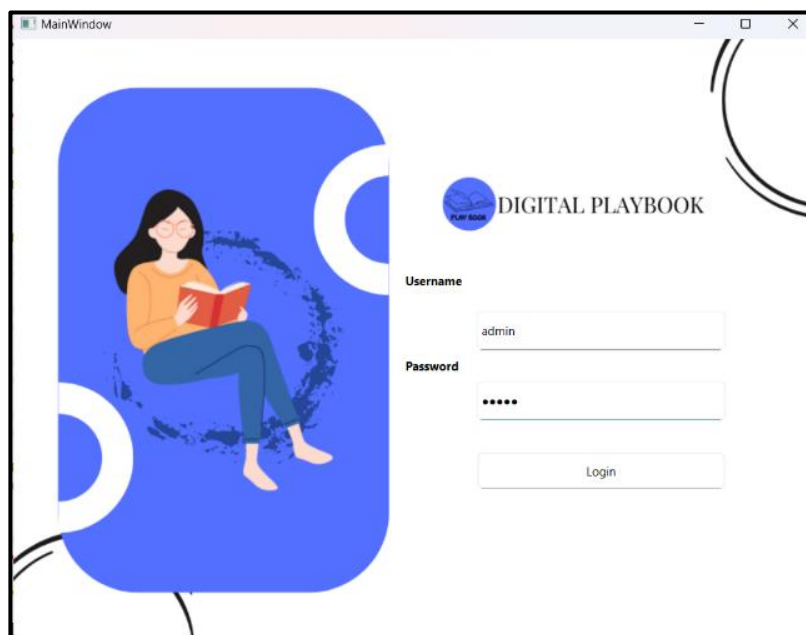
```

```
// Menutup dialog
this->close();
}
```

Penjelasan Source Code :

Kode di atas merupakan implementasi kelas `editbuku`, sebuah *dialog window* berbasis Qt yang digunakan untuk mengedit informasi buku yang sudah ada dalam sistem. Kelas ini menerima parameter `BukuManager` dan `judulLama` agar dapat mengakses serta memodifikasi buku yang dipilih oleh admin. Di dalam konstruktor `editbuku::editbuku`, antarmuka pengguna diinisialisasi melalui `setupUi(this)` dan tombol "Edit" dikaitkan dengan fungsi `editBuku()` menggunakan mekanisme *signal-slot*. Metode `setJudul`, `setPengarang`, `setTahunTerbit`, dan `setHarga` digunakan untuk menampilkan data lama buku ke dalam *text field*, sehingga admin dapat melihat dan mengubahnya. Saat tombol edit diklik, fungsi `editBuku()` akan mengambil nilai baru dari input pengguna dan memanggil metode `perbaruiBuku()` pada `bukuManager` untuk mengganti data berdasarkan judul buku lama (`judulLama`). Setelah data berhasil diperbarui, sebuah pesan konfirmasi akan ditampilkan dengan `QMessageBox`, lalu sinyal `dialogClosed()` dikirim untuk memberi tahu bahwa perubahan telah dilakukan, yang biasanya digunakan untuk menyegarkan tampilan data di jendela utama. Terakhir, dialog ditutup dengan `this->close()`. Kelas ini memfasilitasi proses edit buku dengan cara yang terstruktur dan interaktif, memungkinkan admin memperbarui data secara efisien tanpa harus mengakses struktur data secara langsung.

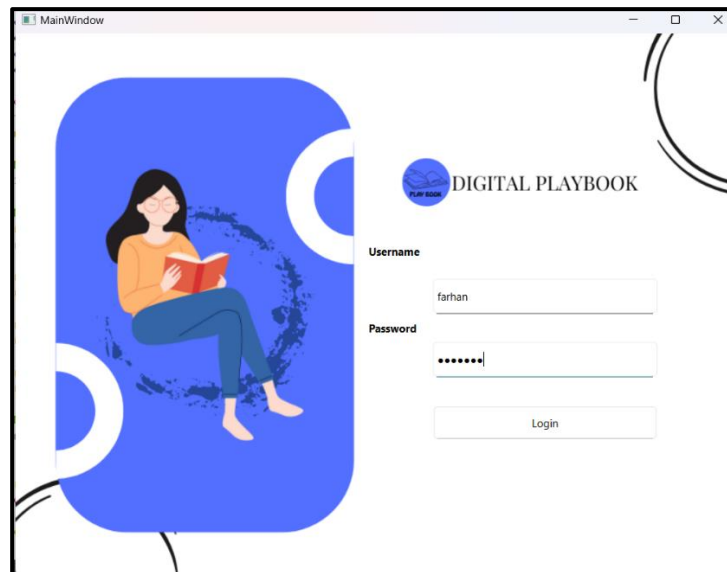
➤ Tampilan Program dengan Tema SI PI



Gambar 20 Tampilan Login dengan ID Admin

Penjelasan :

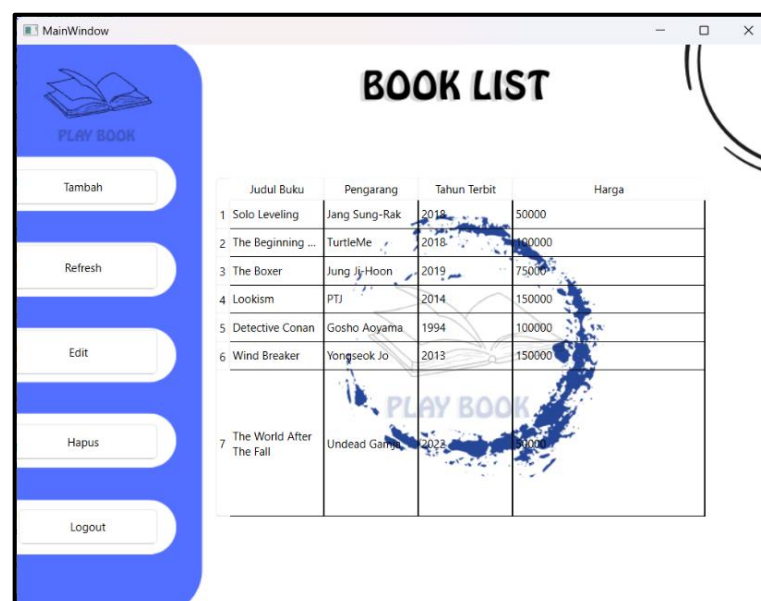
Tampilan ini adalah halaman login, apabila ingin mengakses sebagai admin. Admin perlu memasukkan username dan password sesuai untuk mengakses halaman book list. Pada tampilan login admin terdiri dari dua input, yaitu username & password serta tombol login.



Gambar 21 Tampilan Login dengan ID User

Penjelasan :

Tampilan ini adalah halaman *login* khusus untuk pengguna berperan sebagai *user*. *User* perlu memasukkan *username* dan *password* sesuai untuk mengakses halaman daftar buku. Pada tampilan *login user* terdiri dari dua input, yaitu *username* & *password* serta tombol *login*.



Gambar 22 Tampilan Admin

Penjelasan :

Setelah berhasil login, admin akan langsung diarahkan ke dashboard utama. Di sini, admin dapat melihat seluruh daftar buku dan tersedia tombol untuk melakukan aksi *CRUD* seperti tombol tambah, hapus, dan juga tombol edit data.

Gambar 23 Tampilan Admin Menambahkan Buku

Penjelasan :

Saat *admin* menekan tombol “tambah”, akan muncul *form dialog* untuk menambahkan data baru. *Admin* mengisi nama judul, pengarang, tahun buku, dan harga sebelum menyimpannya.

	Judul Buku	Pengarang	Tahun Terbit	Harga
1	Solo Leveling	Jang Sung-Rak	2018	50000
2	The Beginning ...	TurtleMe	2018	100000
3	Ti			50000
4	Lk			50000
5	D			00000
6	W			50000
7	The World After The Fall	Undead Gangs	2022	50000

Gambar 24 Tampilan Admin Menghapus Buku

Penjelasan :

Menunjukkan kondisi ketika *admin* memilih salah satu baris barang dari tabel tersebut dan akan menekan tombol hapus. Akan muncul konfirmasi sebelum data benar-benar dihapus.

The screenshot shows a dialog box titled "EDIT BUKU". It has a logo "PLAY BOOK" with a book icon. The form contains four input fields: "Judul" with the value "The Boxer", "Pengarang" with "Jung Ji-Hoon", "Tahun Terbit" with "2019", and "Harga" with "90000". A blue circular watermark with a book icon and the text "PLAY BOOK" is centered over the form. An "Edit" button is located at the bottom right.

Gambar 25 Tampilan Admin Mengupdate Buku

Penjelasan :

Saat *admin* menekan tombol “edit”, akan muncul *form dialog* untuk mengedit isi data. *Admin* dapat mengubah nama judul, pengarang, tahun buku, dan harga sebelum menyimpannya

The screenshot shows a user interface titled "DAFTAR BUKU". It features a search bar at the top with a "Cari" button. Below it is a table with 7 rows of book data. At the bottom, there are "Logout" and "Beli" buttons. A blue circular watermark with a book icon and the text "PLAY BOOK" is centered over the table.

	Judul	Harga
1	Solo Leveling	50000
2	The Beginning ...	100000
3	The Boxer	75000
4	Lookism	150000
5	Detective Conan	100000
6	Wind Breaker	150000
7	The World After The Fall	50000

Gambar 26 Tampilan User

Penjelasan :

Tampilan ini diperuntukkan bagi pengguna biasa (*user*). Mereka hanya dapat melihat data yang ada dalam bentuk table saja, tanpa bisa menambah, menghapus, ataupun memperbarui barang. dalam aplikasi ini dirancang untuk memudahkan pengguna untuk membeli buku yang terdapat pada daftar buku. Tampilan utama mencari buku terdiri dari sebuah tabel yang menampilkan daftar buku yang tersedia, dengan kolom-kolom yang menampilkan informasi tentang setiap buku seperti judul dan harga.

BAB IV

PENUTUP

A. Kesimpulan

Dari laporan praktikum ini, dapat disimpulkan bahwa aplikasi Play Book yang dikembangkan berhasil menciptakan antarmuka pengguna yang intuitif dan fungsional untuk menampilkan data yang diambil dari file CSV. Aplikasi S1 Play Book berhasil dikembangkan sebagai media pembelajaran interaktif yang membantu mahasiswa memahami konsep dasar dari struktur data Linked List. Dengan memanfaatkan Qt Framework dan bahasa pemrograman C++, aplikasi ini mampu menyajikan simulasi visual yang intuitif dan interaktif, sehingga pengguna dapat melihat secara langsung proses penyisipan, penghapusan, pencarian, dan penelusuran elemen dalam Linked List.

Secara keseluruhan, aplikasi ini memberikan Melalui pendekatan visual dan GUI yang mudah digunakan, aplikasi ini tidak hanya meningkatkan pemahaman konseptual mahasiswa, tetapi juga menjembatani kesenjangan antara teori dan implementasi praktis dalam pembelajaran struktur data. Dengan demikian, S1 Play Book dapat menjadi alat bantu yang efektif dalam proses belajar mandiri maupun pembelajaran di kelas. Visualisasi yang ditampilkan dalam antarmuka grafis mempermudah pengguna dalam memahami alur logika dan dinamika struktur data dibandingkan pendekatan berbasis teks atau terminal. Aplikasi ini juga memberikan pengalaman belajar yang menyenangkan dan fleksibel, karena dapat digunakan secara mandiri oleh mahasiswa maupun sebagai alat bantu dalam pembelajaran di kelas. Dengan fitur-fitur yang telah tersedia, S1 Play Book membuktikan bahwa integrasi antara teori dan praktik melalui simulasi grafis dapat meningkatkan pemahaman konsep dasar struktur data. Selain itu, proyek ini menunjukkan bahwa penggunaan teknologi yang tepat seperti Qt dan C++ sangat mendukung pengembangan aplikasi edukatif yang ringan, responsif, dan mudah digunakan.

B. Saran

Agar aplikasi S1 Play Book dapat memberikan manfaat yang lebih luas dan maksimal, beberapa pengembangan lebih lanjut disarankan. Pertama, aplikasi ini sebaiknya dikembangkan untuk mendukung berbagai struktur data lainnya, seperti Stack, Queue, Double Linked List, dan Tree, sehingga cakupan pembelajaran menjadi lebih komprehensif. Kedua, penambahan modul teori atau penjelasan singkat tentang konsep dasar struktur data di dalam aplikasi akan sangat membantu pengguna dalam memahami materi sebelum melakukan simulasi. Selain itu, fitur tambahan seperti Undo/Redo atau langkah mundur akan meningkatkan fleksibilitas penggunaan dan memungkinkan pengguna mengevaluasi setiap langkah operasinya dengan lebih baik.

DAFTAR PUSTAKA

- Anita Sindar, R. M. S. (2019). *Struktur Data Dan Algoritma Dengan C++* (Vol. 1). CV. AA. RIZKY.
- Carrete, J., López-Suárez, M., Raya-Moreno, M., Bochkarev, A. S., Royo, M., Madsen, G. K., ... & Rurali, R. (2019). Phonon transport across crystal-phase interfaces and twin boundaries in semiconducting nanowires. *Nanoscale*, 11(34), 16007-16016.
- Dewi, L. J. E. (2010). Media Pembelajaran Bahasa Pemrograman C++. *Jurnal Pendidikan Teknologi dan Kejuruan*, 7(1).
- GeeksforGeeks. (2024). Array List Data Structure
- Malik, D. S. (2018). *C++ Programming: From Problem Analysis to Program Design* (8th ed.). Cengage Learning.
- Microsoft Docs. (2024). std::vector - C++ Reference
- Ramakrishnan, R., & Kaur, H. (n.d.). *Reference model for effective performance and availability monitoring in large scale software systems*.
- Ritonga, A., & Yahfizham, Y. (2023). Studi literatur perbandingan bahasa pemrograman C++ dan bahasa pemrograman Python pada algoritma pemrograman. *Jurnal Teknik Informatika dan Teknologi Informasi*, 3(3), 56-63.
- TORRES GAITÁN, R. (1976). La teoría del comercio internacional de Adam Smith. *Problemas del Desarrollo*, 135-152.

LAMPIRAN



Gambar 27 Lampiran 1



Gambar 28 Lampiran 2



Gambar 29 Lampiran 3



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

LEMBAR ASISTENSI

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Nama Anggota kelompok 2 : 1. Aggra Kurnia Idhan G1A024003
2. Atikah Putri Utami G1A024027
3. Agief Vemas Afrivanzah G1A024037
4. Farhan Khairullah G1A024043

Dosen : 1. Arie Vatriesia, S.T.,M.TI,Ph.D.
2.Kurnia Anggraini, S.T.,M.T,Ph.D

Asisten Dosen : 1. Abdi Agung Kurniawan G1A022011
2. Diodo Arrahman G1A022027
3. Diosi Putri Arlita G1A023012
4. Lio Kusnata G1A023013
5. Sallaa Fikriyatul 'Arifah G1A023015
6. Muhammad Yasser G.T. A. G1A023030
7. Anis Syarifatul Mursyidah G1A023036
8. Rayhan Muhammad Adha G1A023051
9. Najwa Nabilah Wibisono G1A023065
10.Dinda Krisnauli Pakpahan G1A023076

Laporan Tubes Kedua	Catatan dan Tanda Tangan
Kelompok 2	