

LAPORAN TUGAS BESAR
SISTEM INFORMASI PERPUSTAKAAN
MENGGUNAKAN METODE BINARY SEARCH
STRUKTUR DATA DAN ALGORITMA



Disusun Oleh:

- | | |
|-----------------------------|-------------|
| 1. Nurmelizah | (G1A024013) |
| 2. Nurul Khalifah | (G1A024023) |
| 3. Aulia Afdhal Hafizah | (G1A024029) |
| 4. Judika Ebenezer Sianturi | (G1A024031) |

Nama Asisten Dosen:

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

Dosen Pengampu:

1. Arie Vatesia, 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

Puji syukur atas kehadiran Allah Swt. dan Tuhan Yang Maha Esa yang telah memberikan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan laporan tugas besar "Sistem Informasi Perpustakaan Menggunakan Metode Binary Search" ini tepat pada waktunya. Adapun tujuan ditulisnya laporan ini adalah untuk memenuhi tugas besar pada mata kuliah Struktur Data dan Algoritma (SDA). Selain itu, laporan ini bertujuan untuk menambah wawasan mengenai sistem informasi perpustakaan yang menggunakan metode binary search baik bagi pembaca maupun penulis.

Ucapan terima kasih kami sampaikan kepada asisten dosen SDA yang telah memberikan tugas ini sehingga kami dapat meningkatkan kemampuan dan menambah wawasan kami mengenai Binary Search ini. Kami menyadari bahwasanya tugas yang kami buat masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran dari teman-teman semua sangat diperlukan untuk kesempurnaan laporan-laporan berikutnya.

Bengkulu, 21 Mei 2025

Penulis

DAFTAR ISI

KATA PENGANTAR	ii
DAFTAR ISI	iii
DAFTAR GAMBAR	iv
BAB I PENDAHULUAN	1
A. Latar Belakang.....	1
B. Rumusan Masalah.....	1
C. Tujuan dan Manfaat.....	1
BAB II LANDASAN TEORI.....	2
BAB III PEMBAHASAN	3
BAB IV KESIMPULAN DAN SARAN	25
A. Kesimpulan.....	25
B. Saran	25
DAFTAR PUSTAKA	26
LAMPIRAN	27

DAFTAR GAMBAR

Gambar 1. 1 Tampilan ui admin.....	3
Gambar 1. 2 Tampilan halaman login	4
Gambar 1. 3 Tampilan ui user	4
Gambar 1. 4 Kode untuk mendeklarasikan dashboard admin	5
Gambar 1. 5 Kode untuk mendeklarasikan global data.....	7
Gambar 1. 6 Kode untuk mendeklarasikan halaman login.....	8
Gambar 1. 7 Kode untuk mendeklarasikan dashboard user	10
Gambar 1. 8 Kode untuk mengimplementasikan dashboard admin.....	13
Gambar 1. 9 Kode untuk mengimplementasikan global data.....	17
Gambar 1. 10 Kode untuk mengimplementasikan main.cpp.....	18
Gambar 1. 11 Kode untuk mengimplementasikan halaman login.....	19
Gambar 1. 12 Kode untuk mengimplementasikan dashboard user	22

BAB I

PENDAHULUAN

A. Latar Belakang

Perpustakaan merupakan pusat informasi yang vital dalam mendukung kegiatan akademik dan pembelajaran. Dengan meningkatnya jumlah koleksi buku dan pengguna, pengelolaan informasi perpustakaan secara manual menjadi kurang efisien dan rentan terhadap kesalahan. Oleh karena itu, diperlukan sistem informasi perpustakaan yang terkomputerisasi untuk meningkatkan efisiensi dan akurasi dalam pengelolaan data.

Salah satu tantangan utama dalam sistem informasi perpustakaan adalah proses pencarian buku yang cepat dan akurat. Untuk mengatasi hal ini, algoritma pencarian seperti Binary Search dapat diterapkan. Binary Search adalah algoritma efisien untuk mencari data dalam kumpulan yang telah terurut, dengan kompleksitas waktu pencarian sebesar $O(\log n)$, yang berarti waktu pencarian meningkat secara logaritmik terhadap jumlah data.

Beberapa penelitian telah menunjukkan bahwa penerapan algoritma Binary Search dalam sistem informasi perpustakaan dapat meningkatkan kecepatan dan efisiensi pencarian data. Misalnya, penelitian oleh Umbu dan Sulaiman (2022) menerapkan algoritma Binary Search dalam sistem informasi perpustakaan di SMP Negeri 4 Mauliru, yang menghasilkan peningkatan efisiensi dalam pencarian buku.

Dengan mempertimbangkan pentingnya efisiensi dalam pencarian data dan pengelolaan informasi perpustakaan, laporan ini bertujuan untuk merancang dan mengimplementasikan sistem informasi perpustakaan yang mengintegrasikan algoritma Binary Search dalam struktur data yang sesuai. Diharapkan sistem ini dapat meningkatkan kinerja pencarian dan pengelolaan data perpustakaan secara keseluruhan.

B. Rumusan Masalah

1. Buatlah sebuah projek sistem informasi perpustakaan menggunakan metode *binary search*. Hanya menggunakan *array* atau *linked list* sebagai penyimpanan data, tidak menggunakan *database*. Minimal terdiri dari 3 *screen* halaman dan dibuat dari pandangan user dan admin. Kemudian admin bisa melakukan CRUD!

C. Tujuan dan Manfaat

Laporan ini bertujuan untuk menerapkan konsep binary search dalam sistem informasi perpustakaan dan menjelaskan langkah-langkah pembuatannya. Binary search membantu proses pencarian data buku menjadi lebih cepat dan efisien, terutama ketika data sudah terurut. Hal ini sangat membantu pengguna dan admin dalam mengakses dan mengelola data buku, apalagi jika data buku dalam jumlah yang banyak.

BAB II

LANDASAN TEORI

Menurut Wulandari (2023), penerapan sistem informasi digital di perpustakaan sekolah secara signifikan membantu proses pendataan dan pencarian buku, terutama ketika jumlah koleksi semakin banyak. Sistem ini biasanya dibangun berbasis web atau desktop, dilengkapi dengan fitur pencarian dan pengelolaan database yang mendukung kegiatan perpustakaan.

Struktur data adalah metode khusus untuk mengatur dan menyimpan data dalam komputer agar dapat digunakan secara efisien. Dalam sistem informasi, terutama pada sistem pencarian data seperti perpustakaan, pemilihan struktur data yang tepat sangat mempengaruhi kinerja sistem. Struktur data linier seperti array dan linked list serta struktur data non-linier seperti tree dan graph memiliki karakteristik berbeda dalam hal penyimpanan dan kecepatan akses data. Menurut Tirta (2023), dalam konteks pengelolaan data buku, penggunaan struktur data array memungkinkan pengurutan dan pencarian yang lebih cepat jika data sudah dalam keadaan terurut, sedangkan penggunaan binary search tree lebih fleksibel terhadap data yang sering berubah. Pemahaman terhadap struktur data sangat penting untuk menentukan algoritma pencarian yang akan digunakan dalam sistem.

Binary Search merupakan algoritma pencarian yang bekerja pada data yang sudah terurut. Proses pencarian dimulai dengan membandingkan elemen tengah dari array dengan elemen yang dicari. Jika elemen yang dicari lebih kecil, maka pencarian dilanjutkan pada bagian kiri array, dan sebaliknya. Proses ini terus diulang hingga elemen ditemukan atau jangkauan pencarian habis. Keunggulan algoritma ini adalah efisiensi waktu pencarian, dengan kompleksitas $O(\log n)$, dibandingkan metode pencarian linear yang memiliki kompleksitas $O(n)$. Dalam penelitian oleh Religia (2019), Binary Search terbukti lebih cepat dalam pencarian data buku pada sistem perpustakaan digital dibandingkan dengan Sequential Search, terutama ketika data mencapai ribuan entri. Penggunaan algoritma ini sangat relevan pada sistem yang memiliki database besar dan kebutuhan pencarian cepat.

Penggunaan algoritma Binary Search dalam sistem informasi perpustakaan sangat berguna untuk mempercepat pencarian data koleksi buku. Dalam implementasinya, data buku diurutkan berdasarkan atribut tertentu, seperti judul atau kode buku, agar proses pencarian bisa dilakukan secara efisien. Menurut Aryabimo et al. (2024), penggabungan antara Binary Search dan algoritma lainnya seperti Boyer-Moore dalam sistem pencarian berbasis web menunjukkan peningkatan performa pencarian dan kenyamanan pengguna. Aplikasi perpustakaan digital yang menerapkan metode ini umumnya memberikan antarmuka pencarian yang responsif dan cepat, memungkinkan pustakawan maupun pengunjung menemukan data dengan lebih efektif.

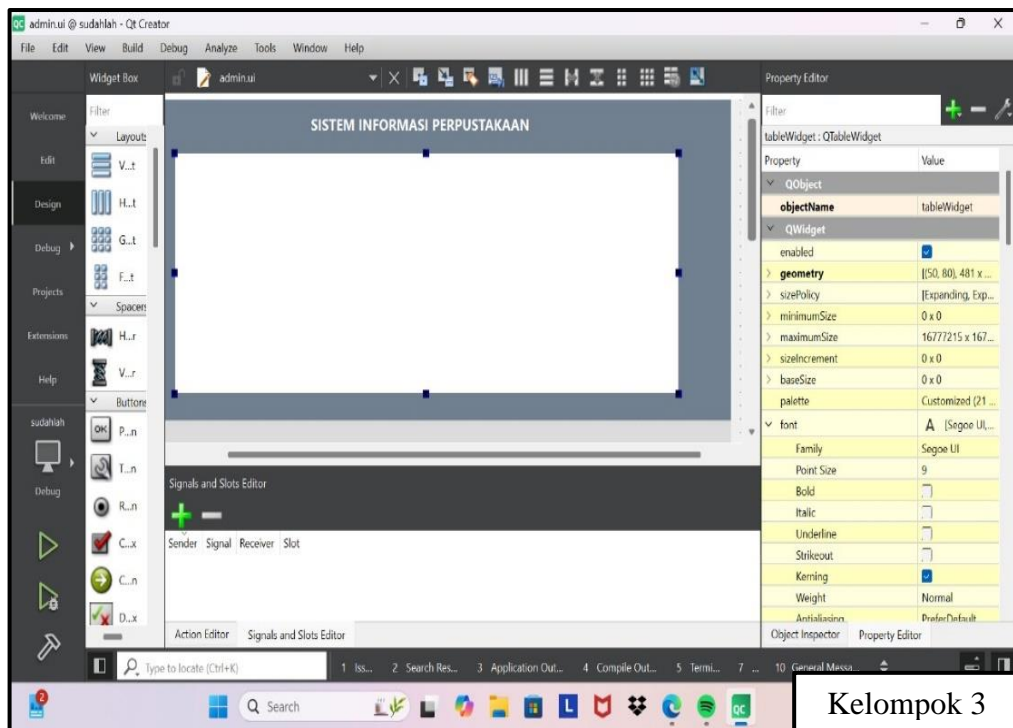
BAB III

PEMBAHASAN

1. Buatlah sebuah projek sistem informasi perpustakaan menggunakan metode binary search. Hanya menggunakan array atau linked list sebagai penyimpanan data, tidak menggunakan database. Minimal terdiri dari 3 screen halaman dan dibuat dari pandangan user dan admin. Kemudian admin bisa melakukan CRUD!

Jawab:

1. Membuat *dashboard* untuk admin

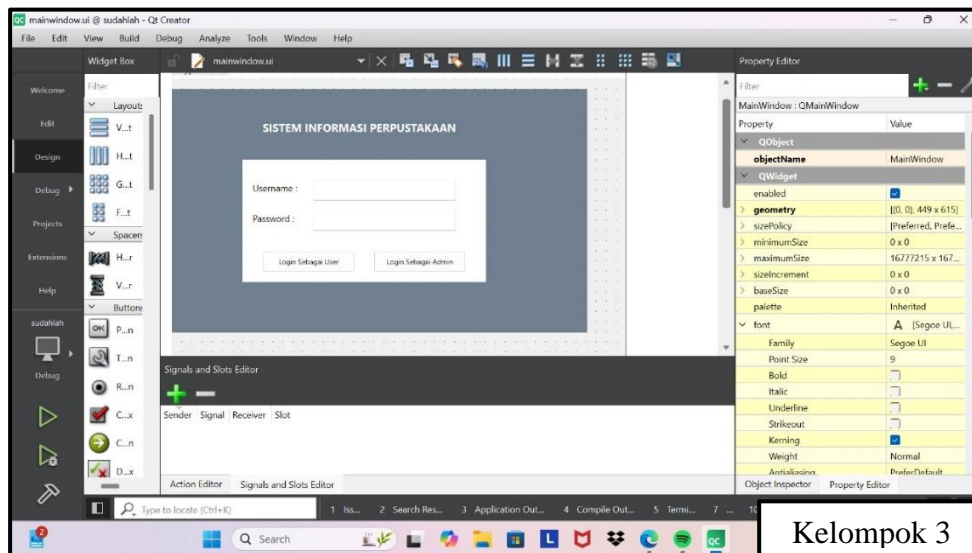


Gambar 1. 1 Tampilan ui admin

Penjelasan gambar:

Desain admin.ui ini menampilkan antarmuka admin sistem informasi perpustakaan yang dirancang untuk mengelola data buku, yang terdiri dari dua bagian utama yaitu formulir "Tambah Buku" di sebelah kiri dan "Edit Buku" di sebelah kanan. Pada masing-masing formulir terdapat kolom input untuk ISBN, judul buku, pengarang, dan tahun terbit. Admin dapat menambahkan buku baru dengan mengisi kolom di sisi kiri lalu menekan tombol "Tambah Buku", atau mengedit dan menghapus data buku yang sudah ada melalui sisi kanan menggunakan tombol "Edit" dan "Hapus". Tombol "Cari" di tengah berfungsi untuk mencari data buku berdasarkan input tertentu, dan tombol "Kembali" di bawah digunakan untuk kembali ke halaman sebelumnya atau keluar dari mode admin. Antarmuka ini dirancang untuk memudahkan admin dalam melakukan fungsi CRUD (*Create, Read, Update, Delete*) terhadap data buku secara efisien.

2. Membuat tampilan login

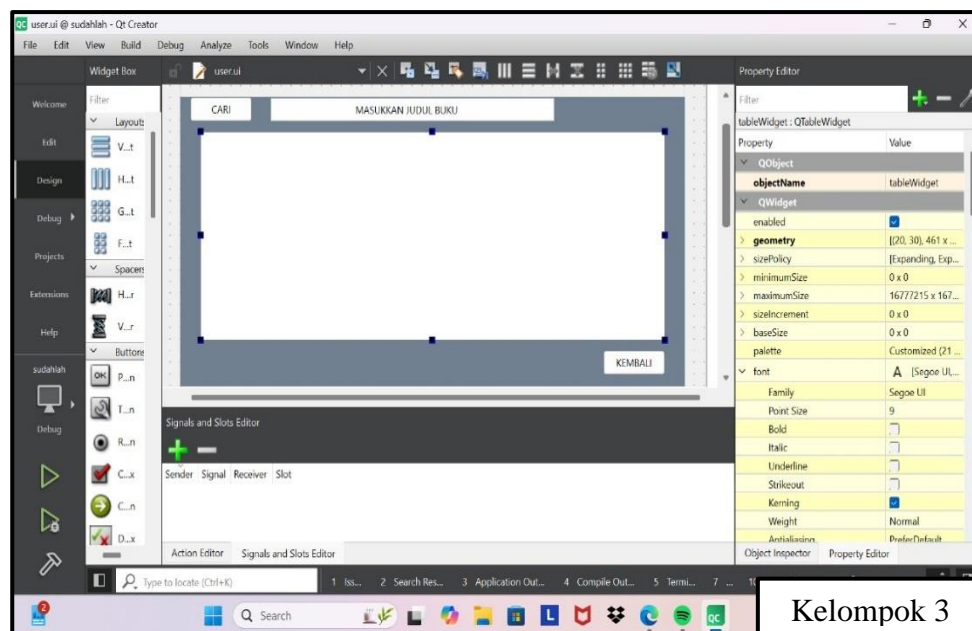


Gambar 1. 2 Tampilan halaman login

Penjelasan gambar:

Di halaman ini terdapat dua tombol utama yaitu tombol **"Login Sebagai User"** yang berfungsi untuk masuk ke sistem sebagai pengguna biasa. Jika username dan password yang dimasukkan benar, maka sistem akan membuka halaman khusus user. Di halaman tersebut, user bisa mencari buku dan melihat informasi yang tersedia, tapi tidak bisa mengubah data. Kemudian ada tombol **"Login Sebagai Admin"** yang digunakan untuk masuk sebagai admin atau pengelola perpustakaan. Setelah login, pengguna akan diarahkan ke halaman admin, di mana mereka bisa melihat dan mengelola data buku atau anggota.

3. Membuat dashboard untuk user

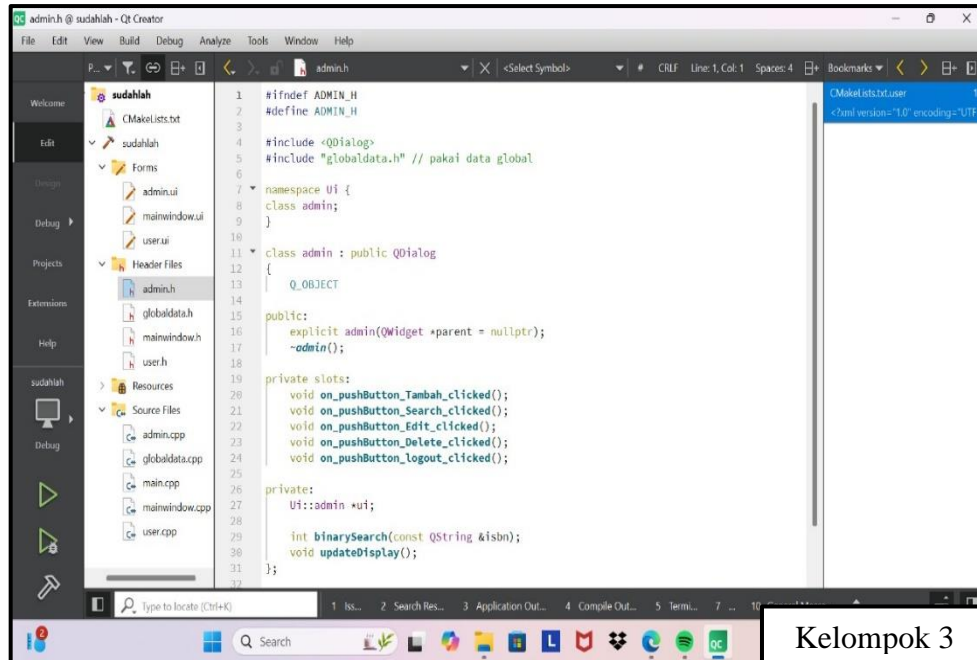


Gambar 1. 3 Tampilan ui user

Penjelasan gambar:

Kolom Input **"MASUKKAN JUDUL BUKU"** berfungsi untuk mengetikkan nama atau kata kunci dari judul buku yang ingin dicari oleh user. Tombol **"CARI"** digunakan untuk menjalankan proses pencarian. Hasil dari pencarian akan ditampilkan pada tabel di bawahnya. Tombol **"KEMBALI"** digunakan untuk kembali ke halaman login, misalnya jika user ingin keluar atau login sebagai peran yang berbeda (admin).

4. Mendeklarasikan dashboard admin



Gambar 1. 4 Kode untuk mendeklarasikan dashboard admin

Source code:

```

#ifndef ADMIN_H
#define ADMIN_H

#include <QDialog>
#include "globaldata.h" // pakai data global

namespace Ui {
class admin;
}

class admin : public QDialog
{
    Q_OBJECT

public:

```

```

explicit admin(QWidget *parent = nullptr);
~admin();

private slots:
    void on_pushButton_Tambah_clicked();
    void on_pushButton_Search_clicked();
    void on_pushButton_Edit_clicked();
    void on_pushButton_Delete_clicked();
    void on_pushButton_logout_clicked();

private:
    Ui::admin *ui;

    int binarySearch(const QString &isbn);
    void updateDisplay();
};

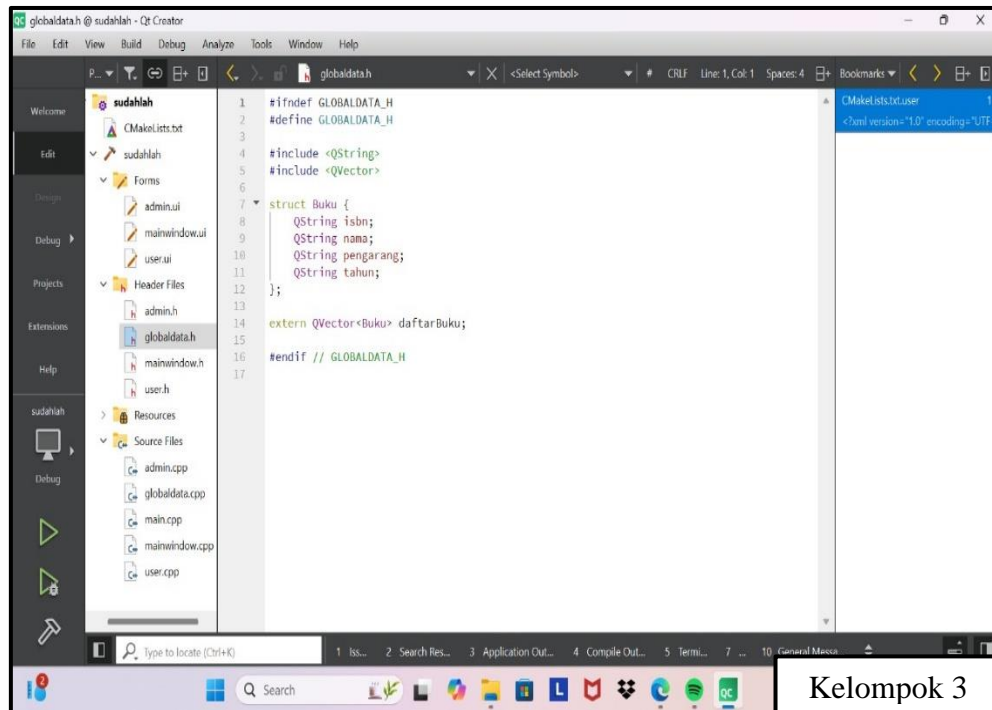
```

#endif // ADMIN_H

Penjelasan source code:

Pada file **admin.h** terdapat class bernama **admin** yang digunakan untuk menampilkan dan mengatur tampilan jendela admin di aplikasi perpustakaan. Class ini merupakan turunan dari **QDialog** yaitu jendela yang bisa muncul terpisah dan digunakan untuk interaksi. Di dalamnya didefinisikan beberapa fungsi (slot) yang akan dijalankan ketika tombol-tombol tertentu diklik. **on_pushButton_Tambah_clicked()** digunakan untuk menambah data buku, **on_pushButton_Search_clicked()** digunakan untuk mencari buku berdasarkan ISBN, **on_pushButton_Edit_clicked()** digunakan untuk mengedit data buku, **on_pushButton_Delete_clicked()** digunakan untuk menghapus buku, dan **on_pushButton_logout_clicked()** digunakan untuk keluar dari halaman admin. Selain itu, terdapat dua fungsi penting yang dibuat di bagian **private**, yaitu **binarySearch(const QString &isbn)** untuk mencari data buku dengan algoritma binary search, dan **updateDisplay()** untuk menampilkan data terbaru ke dalam tabel buku. Di bagian awal ada juga **#include "globaldata.h"**, yang artinya file ini menggunakan data global berupa daftar buku yang didefinisikan sebelumnya di file **globaldata.h**. Header guard **#ifndef ADMIN_H** dan **#define ADMIN_H** digunakan supaya file ini tidak dibaca berkali-kali saat dikompilasi.

5. Mendeklarasikan global data



Gambar 1. 5 Kode untuk mendeklarasikan global data

Source code:

```
#ifndef GLOBALDATA_H
```

```
#define GLOBALDATA_H
```

```
#include <QString>
```

```
#include <QVector>
```

```
struct Buku {
```

```
    QString isbn;
```

```
    QString nama;
```

```
    QString pengarang;
```

```
    QString tahun;
```

```
};
```

```
extern QVector<Buku> daftarBuku;
```

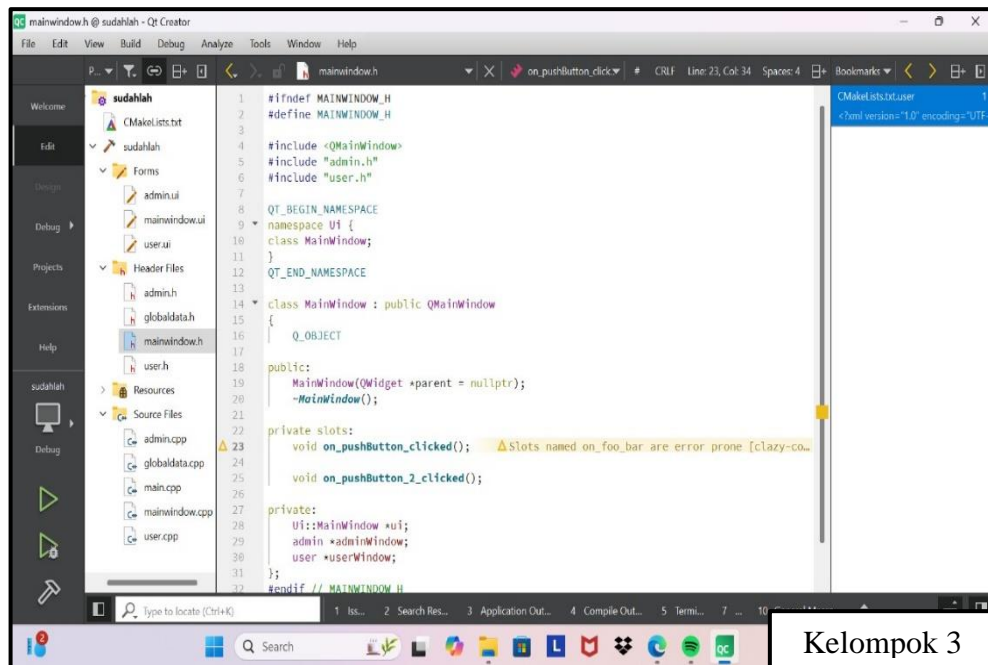
```
#endif // GLOBALDATA_H
```

Penjelasan source code:

Pada kode `globaldata.h` ini, struktur data **Buku** berisi informasi tentang buku, yaitu **isbn**, **nama**, **pengarang**, dan **tahun**, yang bertipe **QString** untuk menyimpan teks. Struktur

ini berfungsi sebagai template untuk menyimpan data buku-buku yang akan digunakan dalam aplikasi perpustakaan. deklarasi `extern QVector<Buku> daftarBuku;` berarti variabel `daftarBuku` adalah sebuah vektor (seperti array dinamis) yang berisi banyak objek `Buku`. Kata kunci `extern` di sini menandakan bahwa variabel ini didefinisikan di file lain, tetapi bisa diakses dari file mana saja yang meng-include `globaldata.h`. Fungsi `daftarBuku` berfungsi sebagai tempat penyimpanan global untuk semua data buku dalam aplikasi. Kode ini diawali dan diakhiri dengan header guard `#ifndef GLOBALDATA_H` dan `#define GLOBALDATA_H` untuk mencegah peng-include-an ganda saat kompilasi.

6. Mendeklarasikan halaman login



Gambar 1. 6 Kode untuk mendeklarasikan halaman login

Source code:

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

```

```

#include <QMainWindow>
#include "admin.h"
#include "user.h"

```

```

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

```

```

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

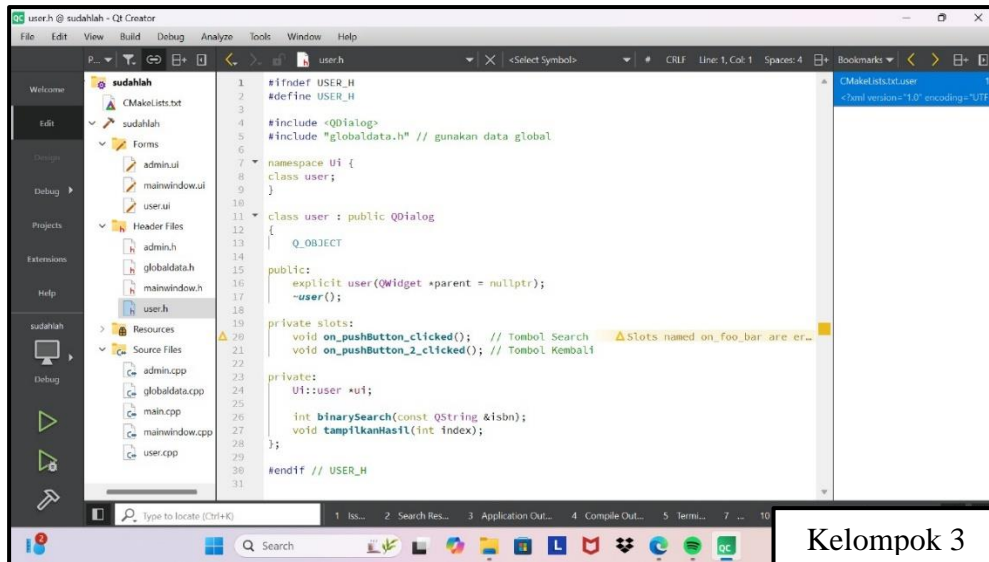
private:
    Ui::MainWindow *ui;
    admin *adminWindow;
    user *userWindow;
};
#endif // MAINWINDOW_H

```

Penjelasan source code:

Baris `#ifndef MAINWINDOW_H` dan `#define MAINWINDOW_H` sebagai header guard agar file ini tidak diproses lebih dari sekali saat kompilasi. Fungsi kode `<QMainWindow>` sebagai dasar pembuatan jendela utama. Fungsi `"admin.h"` dan `"user.h"` digunakan supaya jendela utama dapat memanggil tampilan admin dan user. Di dalam namespace `Ui`, deklarasi kelas `MainWindow` otomatis dibuat oleh Qt Designer. Pada kelas `MainWindow` yang merupakan turunan dari `QMainWindow`, terdapat konstruktor `MainWindow(QWidget *parent = nullptr)` untuk inisialisasi objek dan destruktork `MainWindow()` untuk membersihkan memori. Dalam bagian private slots, terdapat dua fungsi yaitu `on_pushButton_clicked()` dan `on_pushButton_2_clicked()`, yang akan berfungsi sebagai event handler ketika tombol ditekan (misalnya tombol login untuk admin dan user). Pada bagian private, terdapat pointer `Ui::MainWindow *ui` untuk menghubungkan kode dengan tampilan UI dari Qt Designer, serta dua pointer objek yaitu `admin *adminWindow` dan `user *userWindow` yang digunakan untuk membuka jendela admin dan user saat tombol ditekan.

7. Mendeklarasikan dashboard user



Gambar 1. 7 Kode untuk mendeklarasikan dashboard user

Source code:

```
#ifndef USER_H
```

```
#define USER_H
```

```
#include <QDialog>
```

```
#include "globaldata.h" // gunakan data global
```

```
namespace Ui {
```

```
class user;
```

```
}
```

```
class user : public QDialog
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    explicit user(QWidget *parent = nullptr);
```

```
    ~user();
```

```
private slots:
```

```
    void on_pushButton_clicked(); // Tombol Search
```

```
    void on_pushButton_2_clicked(); // Tombol Kembali
```

private:

Ui::user *ui;

int binarySearch(const QString &isbn);

void tampilkanHasil(int index);

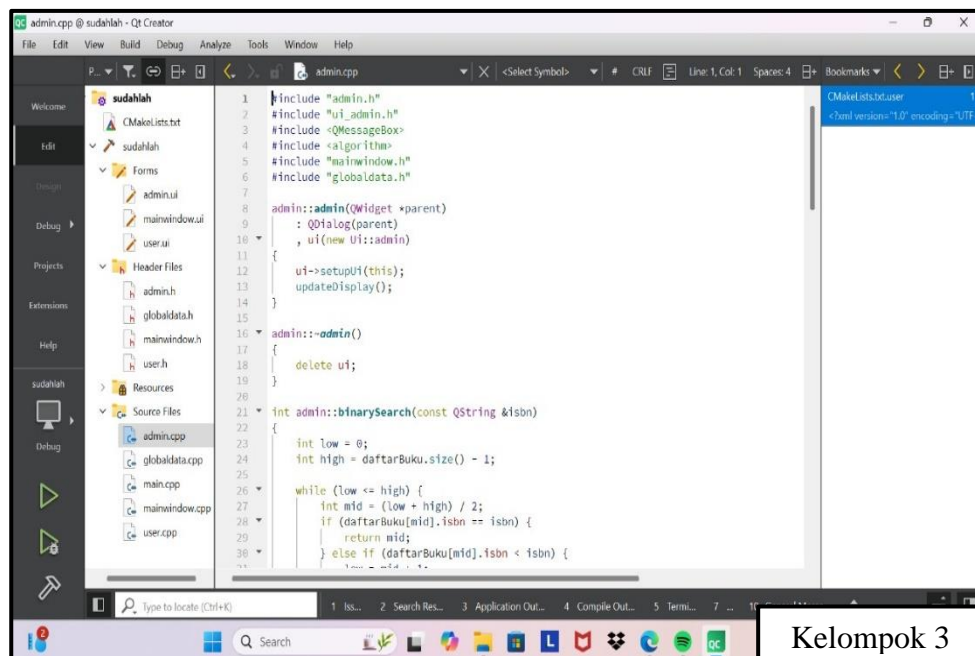
};

#endif // USER_H

Penjelasan source code:

Baris `#ifndef USER_H`, `#define USER_H` dan `#endif` berfungsi sebagai header guard agar file tidak dibaca dua kali saat kompilasi. Fungsi `<QDialog>` digunakan karena kelas user ini merupakan turunan dari `QDialog` agar tampilannya berupa jendela dialog, dan fungsi `"globaldata.h"` untuk menggunakan data buku yang sudah ada secara global. Di dalam kelas user, terdapat konstruktor `user(QWidget *parent = nullptr)` yang berfungsi untuk inisialisasi saat objek dibuat, dan destruktorkan `user()` untuk membersihkan memori saat objek dihapus. Kemudian, di bagian private slots, terdapat dua fungsi yaitu `on_pushButton_clicked()` untuk menangani event tombol pencarian dan `on_pushButton_2_clicked()` untuk tombol kembali atau menutup dialog. Di bagian private, ada pointer `Ui::user *ui` yang menghubungkan kode dengan tampilan yang dibuat di Qt Designer, serta fungsi `binarySearch(const QString &isbn)` untuk mencari data buku berdasarkan ISBN dengan metode binary search, dan fungsi `tampilkanHasil(int index)` untuk menampilkan hasil pencarian ke dalam tabel pada tampilan user.

8. Mengimplementasikan dashboard admin




```

30         } else if (daftarBuku[mid].isbn < isbn) {
31             low = mid + 1;
32         } else {
33             high = mid - 1;
34         }
35     }
36     return -1;
37 }
38
39 void admin::updateDisplay()
40 {
41     ui->tableWidget->clearContents();
42     ui->tableWidget->setRowCount(daftarBuku.size());
43     ui->tableWidget->setColumnCount(4);
44     QStringList headers = {"ISBN", "Nama", "Pengarang", "Tahun"};
45     ui->tableWidget->setHorizontalHeaderLabels(headers);
46
47     for (int i = 0; i < daftarBuku.size(); ++i) {
48         const Buku &b = daftarBuku[i];
49
50         ui->tableWidget->setItem(i, 0, new QTableWidgetItem(b.isbn));
51         ui->tableWidget->setItem(i, 1, new QTableWidgetItem(b.nama));
52         ui->tableWidget->setItem(i, 2, new QTableWidgetItem(b.pengarang));
53         ui->tableWidget->setItem(i, 3, new QTableWidgetItem(b.tahun));
54     }
55 }
56
57 void admin::on_pushButton_Tambah_clicked()
58 {
59     Buku buku;
60     buku.isbn = ui->lineEdit->text().trimmed();

```

Kelompok 3

```

61     buku.nama = ui->lineEdit_2->text().trimmed();
62     buku.pengarang = ui->lineEdit_3->text().trimmed();
63     buku.tahun = ui->lineEdit_4->text().trimmed();
64
65     if (buku.isbn.isEmpty()) {
66         QMessageBox::warning(this, "Peringatan", "ISBN tidak boleh kosong!");
67         return;
68     }
69
70     daftarBuku.append(buku);
71     std::sort(daftarBuku.begin(), daftarBuku.end(), [](const Buku &a, const Buku &b) {
72         return a.isbn < b.isbn;
73     });
74
75     updateDisplay();
76 }
77
78 void admin::on_pushButton_Search_clicked()
79 {
80     QString isbnCari = ui->lineEdit_5->text().trimmed();
81     int index = binarySearch(isbnCari);
82
83     if (index != -1) {
84         const Buku &b = daftarBuku[index];
85         QString info = "ISBN: " + b.isbn + "\nNama: " + b.nama + "\nPengarang: " + b.pengarang;
86         QMessageBox::information(this, "Buku Ditemukan", info);
87     } else {
88         QMessageBox::information(this, "Hasil", "Buku tidak ditemukan.");

```

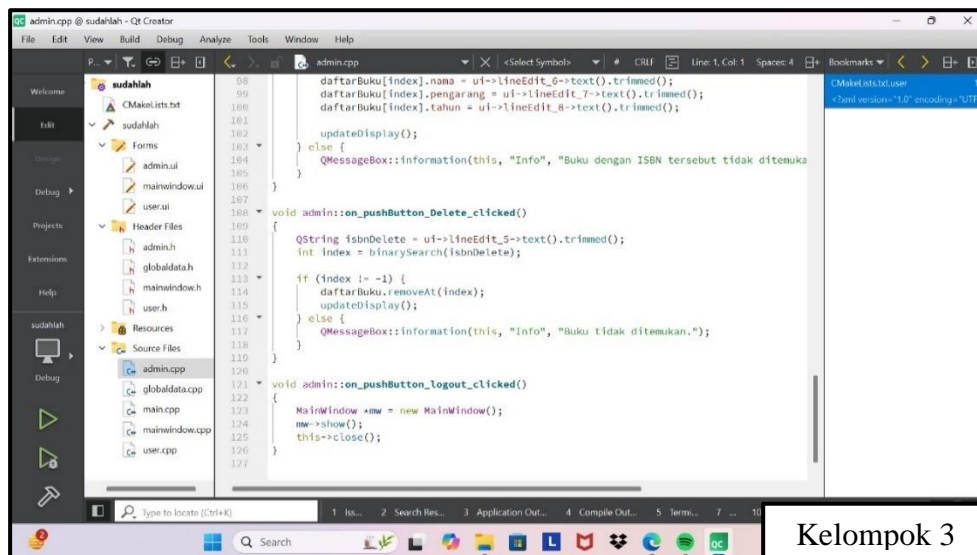
Kelompok 3

```

89     }
90     QMessageBox::information(this, "Hasil", "Buku tidak ditemukan.");
91 }
92
93 void admin::on_pushButton_Edit_clicked()
94 {
95     QString isbnEdit = ui->lineEdit_5->text().trimmed();
96     int index = binarySearch(isbnEdit);
97
98     if (index != -1) {
99         daftarBuku[index].nama = ui->lineEdit_6->text().trimmed();
100        daftarBuku[index].pengarang = ui->lineEdit_7->text().trimmed();
101        daftarBuku[index].tahun = ui->lineEdit_8->text().trimmed();
102
103        updateDisplay();
104    } else {
105        QMessageBox::information(this, "Info", "Buku dengan ISBN tersebut tidak ditemukan.");
106    }
107 }
108
109 void admin::on_pushButton_Delete_clicked()
110 {
111     QString isbnDelete = ui->lineEdit_5->text().trimmed();
112     int index = binarySearch(isbnDelete);
113
114     if (index != -1) {
115         daftarBuku.removeAt(index);
116         updateDisplay();
117     } else {
118         QMessageBox::information(this, "Info", "Buku tidak ditemukan.");

```

Kelompok 3



Gambar 1. 8 Kode untuk mengimplementasikan dashboard admin

Source code:

```
#include "admin.h"
#include "ui_admin.h"
#include <QMessageBox>
#include <algorithm>
#include "mainwindow.h"
#include "globaldata.h"
```

```
admin::admin(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::admin)
{
    ui->setUpUi(this);
    updateDisplay();
}
```

```
admin::~~admin()
{
    delete ui;
}
```

```
int admin::binarySearch(const QString &isbn)
{
    int low = 0;
```

```

int high = daftarBuku.size() - 1;

while (low <= high) {
    int mid = (low + high) / 2;
    if (daftarBuku[mid].isbn == isbn) {
        return mid;
    } else if (daftarBuku[mid].isbn < isbn) {
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}
return -1;
}

void admin::updateDisplay()
{
    ui->tableWidget->clearContents();
    ui->tableWidget->setRowCount(daftarBuku.size());
    ui->tableWidget->setColumnCount(4);
    QStringList headers = {"ISBN", "Nama", "Pengarang", "Tahun"};
    ui->tableWidget->setHorizontalHeaderLabels(headers);

    for (int i = 0; i < daftarBuku.size(); ++i) {
        const Buku &b = daftarBuku[i];

        ui->tableWidget->setItem(i, 0, new QTableWidgetItem(b.isbn));
        ui->tableWidget->setItem(i, 1, new QTableWidgetItem(b.nama));
        ui->tableWidget->setItem(i, 2, new QTableWidgetItem(b.pengarang));
        ui->tableWidget->setItem(i, 3, new QTableWidgetItem(b.tahun));
    }
}

void admin::on_pushButton_Tambah_clicked()
{

```

```

    Buku buku;
    buku.isbn = ui->lineEdit->text().trimmed();
    buku.nama = ui->lineEdit_2->text().trimmed();
    buku.pengarang = ui->lineEdit_3->text().trimmed();
    buku.tahun = ui->lineEdit_4->text().trimmed();

    if (buku.isbn.isEmpty()) {
        QMessageBox::warning(this, "Peringatan", "ISBN tidak boleh kosong!");
        return;
    }

    daftarBuku.append(buku);
    std::sort(daftarBuku.begin(), daftarBuku.end(), [](const Buku &a, const Buku &b) {
        return a.isbn < b.isbn;
    });

    updateDisplay();
}

void admin::on_pushButton_Search_clicked()
{
    QString isbnCari = ui->lineEdit_5->text().trimmed();
    int index = binarySearch(isbnCari);

    if (index != -1) {
        const Buku &b = daftarBuku[index];
        QString info = "ISBN: " + b.isbn + "\nNama: " + b.nama + "\nPengarang: " +
b.pengarang + "\nTahun: " + b.tahun;
        QMessageBox::information(this, "Buku Ditemukan", info);
    } else {
        QMessageBox::information(this, "Hasil", "Buku tidak ditemukan.");
    }
}

void admin::on_pushButton_Edit_clicked()

```

```

{
    QString isbnEdit = ui->lineEdit_5->text().trimmed();
    int index = binarySearch(isbnEdit);

    if (index != -1) {
        daftarBuku[index].nama = ui->lineEdit_6->text().trimmed();
        daftarBuku[index].pengarang = ui->lineEdit_7->text().trimmed();
        daftarBuku[index].tahun = ui->lineEdit_8->text().trimmed();

        updateDisplay();
    } else {
        QMessageBox::information(this, "Info", "Buku dengan ISBN tersebut tidak
ditemukan.");
    }
}

void admin::on_pushButton_Delete_clicked()
{
    QString isbnDelete = ui->lineEdit_5->text().trimmed();
    int index = binarySearch(isbnDelete);

    if (index != -1) {
        daftarBuku.removeAt(index);
        updateDisplay();
    } else {
        QMessageBox::information(this, "Info", "Buku tidak ditemukan.");
    }
}

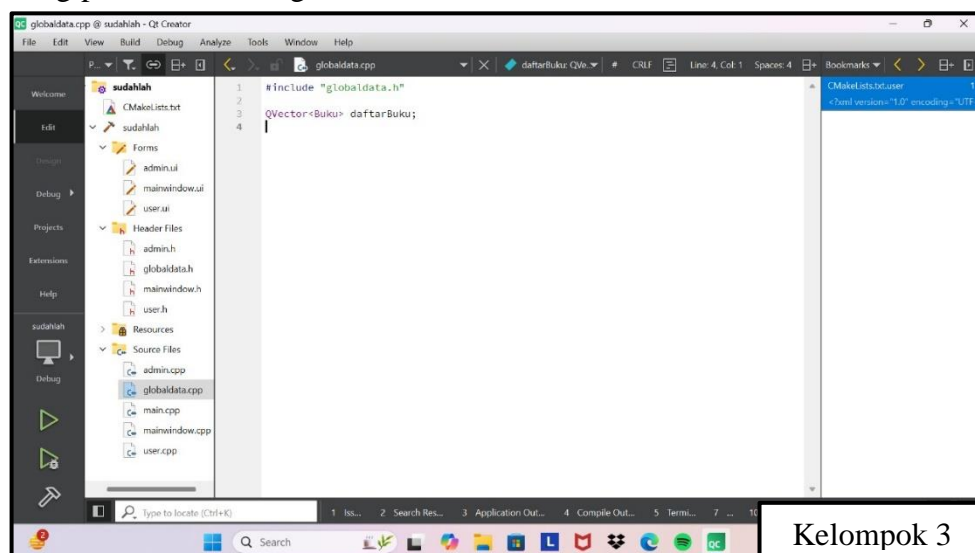
void admin::on_pushButton_logout_clicked()
{
    MainWindow *mw = new MainWindow();
    mw->show();
    this->close();
}

```

Penjelasan source code:

Pemanggilan beberapa file header seperti `admin.h`, `mainwindow.h`, dan `globaldata.h`. Fungsi `admin::admin()` digunakan sebagai konstruktor untuk mengatur tampilan awal jendela admin, sekaligus memanggil fungsi `updateDisplay()` agar data buku langsung ditampilkan dalam tabel. Fungsi `updateDisplay()` bertugas untuk menampilkan semua data buku ke dalam `tableWidget`, dengan kolom ISBN, Nama, Pengarang, dan Tahun. Fungsi `binarySearch()` untuk mencari data buku berdasarkan ISBN. Fungsi ini menggunakan logika pencarian biner, sehingga data harus dalam keadaan terurut. Jika ISBN yang dicari ditemukan, maka akan dikembalikan posisi indeksinya. Jika tidak ditemukan, hasilnya -1. Admin juga bisa menambahkan buku melalui tombol **"Tambah"**. Ketika tombol ditekan, data dari inputan (ISBN, nama, pengarang, dan tahun) akan disimpan dalam array `daftarBuku`. Setelah itu, array tersebut langsung diurutkan berdasarkan ISBN menggunakan `std::sort()` agar tetap rapi untuk keperluan pencarian. Setelah ditambah, tampilannya langsung di-refresh melalui `updateDisplay()`. Admin juga bisa mencari buku dengan memasukkan ISBN ke kolom pencarian, lalu menekan tombol **"Search"**. Jika ditemukan, maka detail bukunya ditampilkan melalui `QMessageBox`. Jika admin ingin mengubah informasi buku yang sudah ada, admin memasukkan ISBN lama, lalu mengisi kolom baru (nama, pengarang, tahun), dan menekan tombol **"Edit"**. Jika buku ditemukan, data langsung diperbarui. Untuk menghapus buku, admin cukup memasukkan ISBN lalu tekan tombol **"Delete"**. Jika buku dengan ISBN tersebut ditemukan, datanya langsung dihapus dari `daftarBuku`, dan tampilan diperbarui. Terakhir, jika admin ingin logout, tombol logout akan mengarahkan kembali ke halaman login (`MainWindow`) dan menutup tampilan admin.

9. Mengimplementasikan global data



Gambar 1. 9 Kode untuk mengimplementasikan global data

Source code:

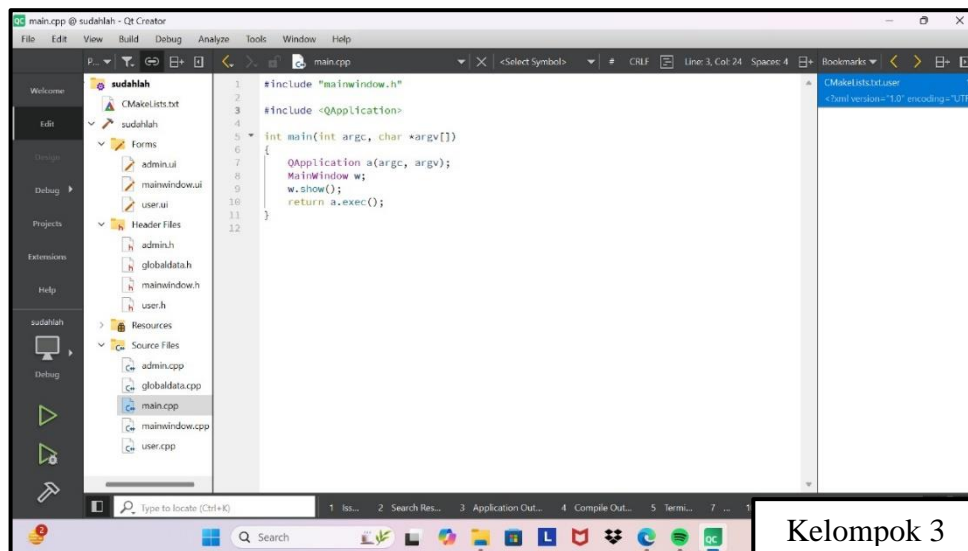
```
#include "globaldata.h"
```

```
QVector<Buku> daftarBuku;
```

Penjelasan source code:

Baris `#include "globaldata.h"` berfungsi untuk mengimpor isi dari file header bernama `globaldata.h`. File ini berisi definisi yang dibutuhkan oleh program, seperti struktur data buku, fungsi-fungsi tertentu, atau variabel global lainnya. Baris `QVector<Buku> daftarBuku;` digunakan untuk membuat sebuah variabel bernama `daftarBuku` yang bertipe `QVector`, yaitu semacam wadah atau list dinamis yang bisa menampung banyak data. Secara sederhana, kode ini sedang menyiapkan sebuah tempat untuk menyimpan daftar buku yang bisa dipakai oleh bagian lain dari program.

10. Mengimplementasikan main.cpp



Gambar 1. 10 Kode untuk mengimplementasikan main.cpp

Source code:

```
#include "mainwindow.h"
```

```
#include <QApplication>
```

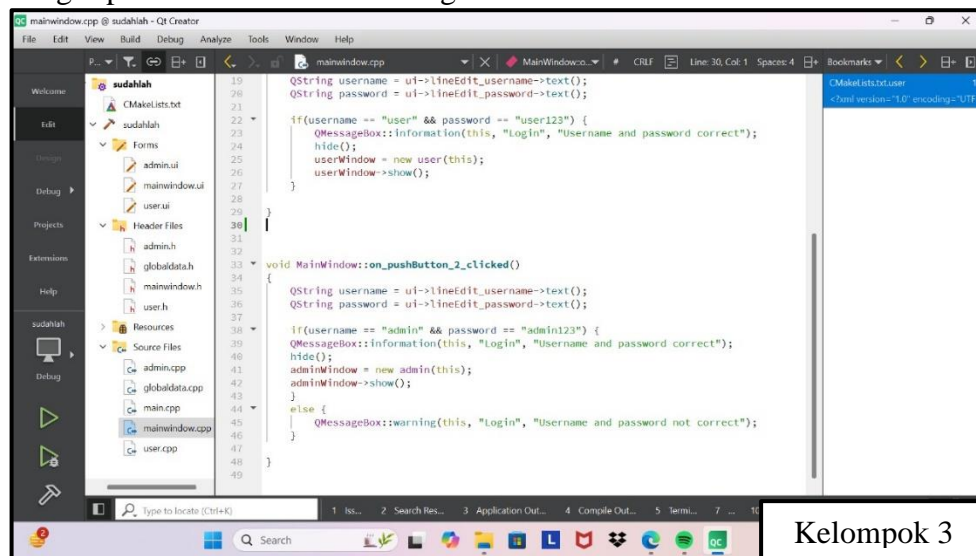
```
int main(int argc, char *argv[])  
{  
    QApplication a(argc, argv);  
    MainWindow w;  
    w.show();  
    return a.exec();  
}
```

```
}
```

Penjelasan source code:

Baris `#include "mainwindow.h"` berfungsi untuk memanggil file header yang isinya adalah desain dan logika dari jendela utama aplikasi. Lalu `#include <QApplication>` digunakan supaya program bisa menjalankan aplikasi GUI dari Qt. Di dalam fungsi `main`, baris `QApplication a(argc, argv);` membuat objek aplikasi Qt, yang wajib ada untuk menjalankan program dengan tampilan GUI. Selanjutnya, `MainWindow w;` membuat objek jendela utama bernama `w`, lalu `w.show();` menampilkan jendela tersebut ke layar. Terakhir, `return a.exec();` menjalankan aplikasi dan membuatnya tetap terbuka sampai pengguna menutupnya. Jadi, kode ini intinya untuk memulai dan menampilkan tampilan utama program GUI ke pengguna.

11. Mengimplementasikan halaman login



Gambar 1. 11 Kode untuk mengimplementasikan halaman login

Source code:

```
#include "mainwindow.h"
#include ".ui_mainwindow.h"
#include <QMessageBox>
```

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
```

```

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    QString username = ui->lineEdit_username->text();
    QString password = ui->lineEdit_password->text();

    if(username == "user" && password == "user123") {
        QMessageBox::information(this, "Login", "Username and password correct");
        hide();
        userWindow = new user(this);
        userWindow->show();
    }

}

void MainWindow::on_pushButton_2_clicked()
{
    QString username = ui->lineEdit_username->text();
    QString password = ui->lineEdit_password->text();

    if(username == "admin" && password == "admin123") {
        QMessageBox::information(this, "Login", "Username and password correct");
        hide();
        adminWindow = new admin(this);
        adminWindow->show();
    }
    else {
        QMessageBox::warning(this, "Login", "Username and password not correct");
    }

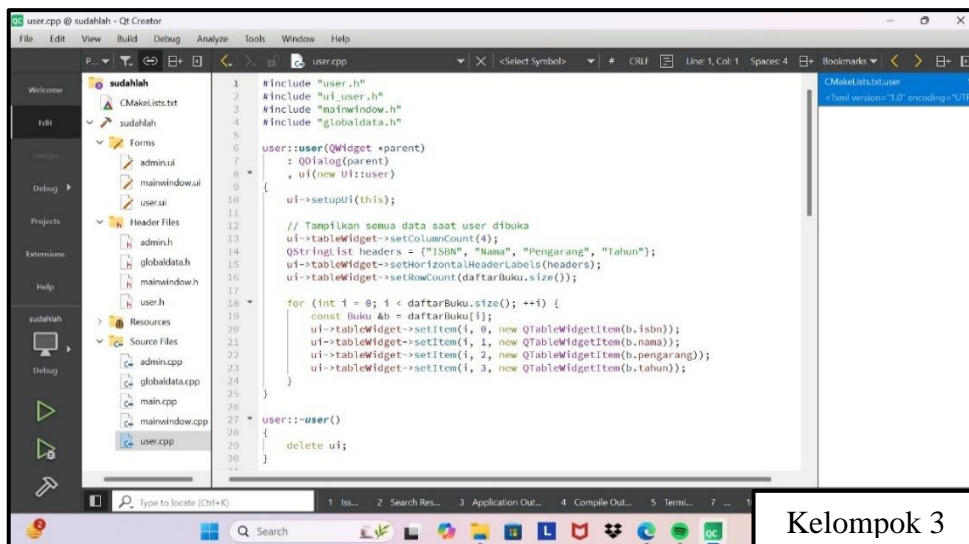
}

```


Penjelasan source code:

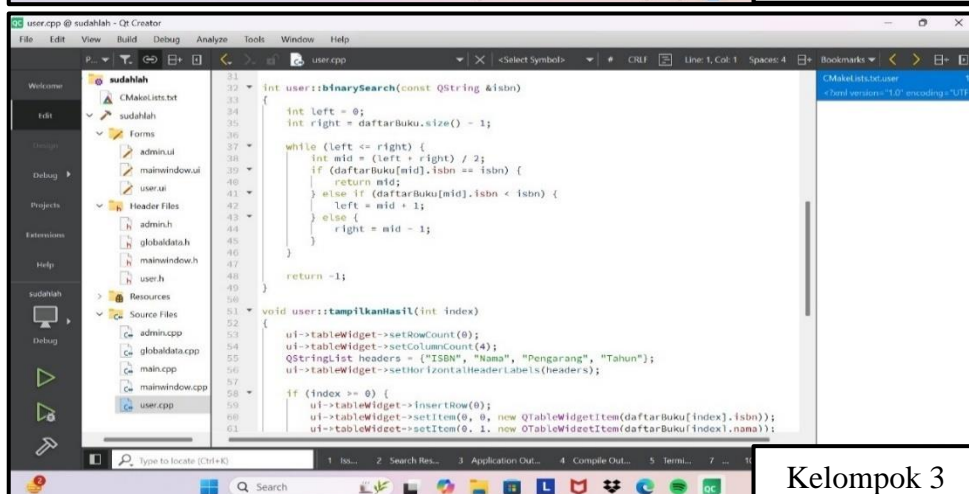
Baris `#include "mainwindow.h"` dan `#include "../ui_mainwindow.h"` berfungsi untuk menghubungkan file dengan desain tampilan (UI) dan logika jendela utama. Baris `#include <QMessageBox>` digunakan untuk menampilkan kotak pesan ke pengguna, misalnya saat login berhasil atau gagal. Di bagian `MainWindow::MainWindow`, program menyiapkan tampilan jendela utama sesuai desain di Qt Designer, dan di `MainWindow` ada perintah `delete ui` untuk membersihkan memori saat program ditutup. Fungsi `on_pushButton_clicked()` dipanggil saat tombol login untuk "user" ditekan. Program akan mengambil teks dari input username dan password, lalu dicek. Jika sesuai (`username = "user", password = "user123"`), maka akan muncul pesan login berhasil, jendela utama disembunyikan, dan jendela khusus user ditampilkan. Sedangkan `on_pushButton_2_clicked()` dipanggil saat tombol login "admin" ditekan. Prosesnya mirip, ambil input username dan password, lalu dicek apakah sama dengan "admin" dan "admin123". Jika benar, jendela admin akan dibuka dan jika salah, akan muncul pesan peringatan bahwa username atau password salah.

12. Mengimplementasikan dashboard user



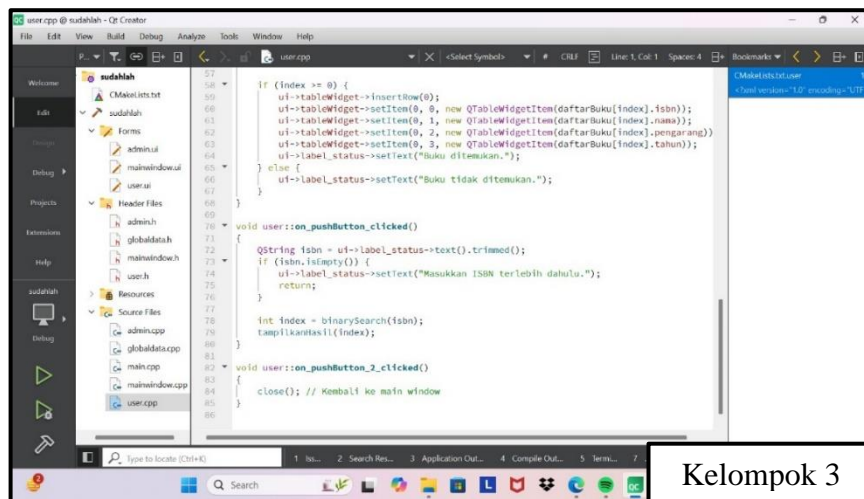
```
1  #include "user.h"
2  #include "ui_user.h"
3  #include "mainwindow.h"
4  #include "globaldata.h"
5
6  user::user(QWidget *parent)
7      : QDialog(parent)
8      , ui(new Ui::user)
9  {
10     ui->setupUi(this);
11
12     // Tampilkan semua data saat user dibuka
13     ui->tableWidget->setColumnCount(4);
14     QStringList headers = {"ISBN", "Nama", "Pengarang", "Tahun"};
15     ui->tableWidget->setHorizontalHeaderLabels(headers);
16     ui->tableWidget->setRowCount(daftarBuku.size());
17
18     for (int i = 0; i < daftarBuku.size(); ++i) {
19         const Buku &b = daftarBuku[i];
20         ui->tableWidget->setItem(i, 0, new QTableWidgetItem(b.isbn));
21         ui->tableWidget->setItem(i, 1, new QTableWidgetItem(b.nama));
22         ui->tableWidget->setItem(i, 2, new QTableWidgetItem(b.pengarang));
23         ui->tableWidget->setItem(i, 3, new QTableWidgetItem(b.tahun));
24     }
25
26     user::~user()
27     {
28         delete ui;
29     }
30 }
```

Kelompok 3



```
31 int user::binarySearch(const QString &isbn)
32 {
33     int left = 0;
34     int right = daftarBuku.size() - 1;
35
36     while (left <= right) {
37         int mid = (left + right) / 2;
38         if (daftarBuku[mid].isbn == isbn) {
39             return mid;
40         } else if (daftarBuku[mid].isbn < isbn) {
41             left = mid + 1;
42         } else {
43             right = mid - 1;
44         }
45     }
46     return -1;
47 }
48
49 void user::tampilkanHasil(int index)
50 {
51     ui->tableWidget->setRowCount(0);
52     ui->tableWidget->setColumnCount(4);
53     QStringList headers = {"ISBN", "Nama", "Pengarang", "Tahun"};
54     ui->tableWidget->setHorizontalHeaderLabels(headers);
55
56     if (index >= 0) {
57         ui->tableWidget->insertRow(0);
58         ui->tableWidget->setItem(0, 0, new QTableWidgetItem(daftarBuku[index].isbn));
59         ui->tableWidget->setItem(0, 1, new QTableWidgetItem(daftarBuku[index].nama));
60     }
61 }
```

Kelompok 3



Gambar 1. 12 Kode untuk mengimplementasikan dashboard user

Source code:

```
#include "user.h"
#include "ui_user.h"
#include "mainwindow.h"
#include "globaldata.h"
user::user(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::user)
{
    ui->setupUi(this);

    // Tampilkan semua data saat user dibuka
    ui->tableWidget->setColumnCount(4);
    QStringList headers = {"ISBN", "Nama", "Pengarang", "Tahun"};
    ui->tableWidget->setHorizontalHeaderLabels(headers);
    ui->tableWidget->setRowCount(daftarBuku.size());

    for (int i = 0; i < daftarBuku.size(); ++i) {
        const Buku &b = daftarBuku[i];
        ui->tableWidget->setItem(i, 0, new QTableWidgetItem(b.isbn));
        ui->tableWidget->setItem(i, 1, new QTableWidgetItem(b.nama));
        ui->tableWidget->setItem(i, 2, new QTableWidgetItem(b.pengarang));
        ui->tableWidget->setItem(i, 3, new QTableWidgetItem(b.tahun));
    }
}
```

```

user::~~user()
{
    delete ui;
}

int user::binarySearch(const QString &isbn)
{
    int left = 0;
    int right = daftarBuku.size() - 1;

    while (left <= right) {
        int mid = (left + right) / 2;
        if (daftarBuku[mid].isbn == isbn) {
            return mid;
        } else if (daftarBuku[mid].isbn < isbn) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}

void user::tampilkanHasil(int index)
{
    ui->tableWidget->setRowCount(0);
    ui->tableWidget->setColumnCount(4);
    QStringList headers = {"ISBN", "Nama", "Pengarang", "Tahun"};
    ui->tableWidget->setHorizontalHeaderLabels(headers);

    if (index >= 0) {
        ui->tableWidget->insertRow(0);
        ui->tableWidget->setItem(0, 0, new QTableWidgetItem(daftarBuku[index].isbn));
        ui->tableWidget->setItem(0, 1, new QTableWidgetItem(daftarBuku[index].nama));
        ui->tableWidget->setItem(0, 2, new QTableWidgetItem(daftarBuku[index].pengarang));
    }
}

```

```

        ui->tableWidget->setItem(0, 3, new QTableWidgetItem(daftarBuku[index].tahun));
        ui->label_status->setText("Buku ditemukan.");
    } else {
        ui->label_status->setText("Buku tidak ditemukan.");
    }
}

void user::on_pushButton_clicked()
{
    QString isbn = ui->label_status->text().trimmed();
    if (isbn.isEmpty()) {
        ui->label_status->setText("Masukkan ISBN terlebih dahulu.");
        return;
    }
    int index = binarySearch(isbn);
    tampilkanHasil(index);
}

void user::on_pushButton_2_clicked()
{
    close(); // Kembali ke main window
}

```

Penjelasan source code:

Ketika jendela user dibuka, program akan langsung menampilkan semua daftar buku yang sudah disimpan di **daftarBuku**. Buku-buku ini ditampilkan dalam bentuk tabel dengan kolom ISBN, Nama, Pengarang, dan Tahun. Setiap baris dalam tabel mewakili satu buku. Data buku diambil dari **globaldata.h** dan dimasukkan satu per satu ke dalam tabel menggunakan perulangan. Fungsi **binarySearch** digunakan untuk mencari buku berdasarkan ISBN. Algoritma pencarian yang dipakai adalah Binary Search, yang artinya data harus sudah dalam kondisi terurut. Fungsi ini akan mengembalikan posisi (index) buku jika ada atau -1 jika tidak ada. Jika user ingin mencari buku, user dapat memasukkan ISBN lalu klik tombol pencarian. Ketika tombol diklik, fungsi **on_pushButton_clicked()** akan dijalankan. Program akan mengambil isi dari label, lalu dicari menggunakan **binarySearch**, dan hasilnya ditampilkan di tabel. Jika buku ditemukan, maka hanya satu baris yang akan ditampilkan, dan status akan menunjukkan “**Buku ditemukan.**” dan jika tidak ditemukan, status akan bilang “**Buku tidak ditemukan.**” tombol keluar (**on_pushButton_2_clicked()**) akan menutup jendela user dan kembali ke jendela sebelumnya.

BAB IV

KESIMPULAN DAN SARAN

A. Kesimpulan

Penerapan algoritma pencarian dalam sistem informasi perpustakaan memberikan dampak besar terhadap efisiensi dan kecepatan dalam menemukan data buku, terutama saat data sudah dalam kondisi terurut. Dengan bantuan algoritma seperti binary search, proses pencarian menjadi lebih cepat dibandingkan dengan metode pencarian biasa yang harus memeriksa satu per satu dari awal sampai akhir. Hal ini sangat berguna ketika jumlah data yang dikelola sudah cukup banyak, karena bisa menghemat waktu dan tenaga, baik dari sisi pengguna maupun admin.

Selain itu, penggunaan struktur data yang sederhana seperti array atau linked list juga sangat membantu dalam menyusun sistem yang ringan dan tetap fungsional. Walaupun tidak menggunakan database eksternal, sistem tetap bisa menjalankan proses seperti menambah data buku, mengedit informasi buku, menghapus buku, atau melihat koleksi buku (CRUD).

Dengan menggabungkan struktur data yang tepat dan algoritma pencarian yang efisien, sistem dapat memberikan pengalaman yang responsif dan tidak membingungkan. Admin dapat mengelola data dengan lebih praktis, dan pengguna biasa pun tetap bisa mencari buku yang dibutuhkan dengan cepat. Hal ini menjadikan sistem informasi perpustakaan sebagai alat yang bisa diandalkan dalam mendukung pelayanan yang cepat, akurat, dan mudah dipahami. Integrasi sederhana ini bisa menjadi langkah awal yang kuat untuk membangun sistem digital perpustakaan yang lebih baik ke depannya.

B. Saran

Agar sistem informasi perpustakaan yang dikembangkan dapat memberikan manfaat optimal, perlu untuk terus melakukan evaluasi dan pengembangan fitur sesuai kebutuhan pengguna, seperti penambahan filter pencarian atau integrasi dengan sistem katalog digital yang lebih luas. Selain itu, penting untuk mempertimbangkan penggunaan struktur data yang lebih dinamis, seperti binary search tree, apabila data buku sering mengalami perubahan, guna menjaga efisiensi pencarian tanpa perlu melakukan pengurutan ulang secara manual. Pengujian berkala dan pelatihan kepada admin serta pengguna juga perlu dilakukan agar pemanfaatan sistem berjalan maksimal dan potensi kesalahan dapat diminimalkan. Dengan langkah-langkah tersebut, sistem informasi perpustakaan berbasis binary search dapat menjadi solusi yang andal dalam mendukung pengelolaan dan pelayanan perpustakaan di era digital.

DAFTAR PUSTAKA

- Umbu, L., & Sulaiman, M. (2022). *Penerapan Algoritma Binary Search Pada Sistem Informasi Perpustakaan di SMP Negeri 4 Mauliru*. Jurnal Sentimas.
- Nainggolan, R. (2022). *Implementasi Algoritma Binary Search Pada Pencarian Data Jemaat Gereja HKBP Manado*. Jurnal Informatika Polinema, 9(1), 17–24.
- Yesy Afrillia, S. R. (2022). *Sistem Informasi Pencarian Tata Letak Buku pada Perpustakaan Menggunakan Metode Binary Search*. Jurnal Ilmu Komputer, 6(1), 31–35.
- Hakiki, T. N., & Hasanah, F. N. (2020). *Pengembangan Sistem Informasi Perpustakaan Berbasis Web terhadap Kemudahan Pelayanan di Fakultas Psikologi dan Ilmu Pendidikan*. TECNOSCIENZA, 5(1).
- Markuci, A., & Prianto, E. (2022). *Analisis Perbandingan Penggunaan Algoritma Binary Search dan Sequential Search pada Sistem Pencarian Data*. Jurnal Informatika Polinema, 9(1), 25–30.

LAMPIRAN





KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BENGKULU

FAKULTAS TEKNIK

PROGRAM STUDI INFORMATIKA

LEMBAR ASISTENSI

PRAKTIKUM PROYEK PEMROGRAMAN BERORIENTASI OBJEK

Nama Mahasiswa	: 1. Nurmelizah	(G1A024013)
	1. Nurul Khalifah	(G1A023023)
	2. Aulia Afdhal Hafidzah	(G1A024029)
	3. Judika Ebenezer Sianturi	(G1A024031)
Dosen	: 1. Arie Vatesia, S.T., M.TI., Ph.D	
	2. Kurnia Anggraini, S.T., M.T., Ph.D	
Asisten	: 1. Diodo Arrahman	(G1A022027)
	2. Abdi Agung Kurniawan	(G1A022011)
	3. Sallaa Fikriyatul 'Arifah	(G1A023015)
	4. Anis Syarifatul Mursyidah	(G1A023036)
	5. Lio Kusnata	(G1A023013)
	6. Dinda Krisnauli Pakpahan	(G1A023076)
	7. Diosi Putri Arlita	(G1A023012)
	8. Rayhan Muhammad Adha	(G1A023051)
	9. Najwa Nabilah Wibisono	(G1A023065)
	10. Muhammad Yasser G.T. A.	(G1A023030)

Laporan Praktikum	Catatan dan Tanda Tangan
Laporan Tubes Struktur Data dan Algoritma	