LAPORAN FINAL PROJECT
PRAKTIKUM PEMOGRAMAN BERORIENTASI OBJEK

APLIKASI PROGRAM MANAJEMEN TOKO BUKU

Oleh :
ALVIN ANUGERAH PRATAMA
(22343019)

Dosen Pengampu :
RANDI PROSKA SANDRA, S.Pd., M.Sc.
(Seksi : 202313430043)

PRODI INFORMATIKA
DEPARTEMEN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023

A. Latar Belakang Aplikasi

Perkembangan globalisasi yang diiringi oleh kemajuan teknologi informasi mengakibatkan arus informasi yang dulunya sulit diperoleh, kini dapat diperoleh dengan mudah pada saat dibutuhkan. Perkembangan teknologi yang semakin maju membutuhkan kinerja yang relatif cepat dan akurat dalam suatu instansi atau bisnis untuk menghasilkan informasi yang dibutuhkan. Salah satunya adalah penerapan teknologi komputer sebagai penunjang utama dalam persaingan usaha dan penunjang sumber daya manusia yang baik. Untuk dapat menciptakan dan mengatasi kondisi tersebut maka diperlukan suatu sistem informasi yang dapat melayani segala macam aspek informasi mengenai kapasitas, keterampilan, keahlian, pengalaman dan potensi pegawai secara cepat dan tepat yang kemudian dapat digunakan untuk menentukan kebijakan atau keputusan dan implementasinya[1]

Permasalahan yang paling penting adalah proses pemasukan data, proses transaksi penjualan dan pelaporan ini selalu dilakukan secara konvensional yaitu dengan mencatat semua transaksinya didalam buku besar, sehingga data yang diperoleh seringkali bermasalah karena human error, misalnya pada saat administrasi membutuhkan informasi mengenai stok barang masuk dan keluar[1]

Aplikasi Program Manajemen Toko Buku adalah sebuah aplikasi yang dibangun agar dapat mempermudah dalam mengelola operasional toko buku. Tujuan dari aplikasi ini adalah membantu penjual dalam meneglola pejualan, jumlah buku, dan pelanggan dengan lebih efisisen. Aplikasi ini dapat digunakan untuk menghitung jumlah penjualan, pelanggan, pemasukan bahkan hingga melacak stok buku.[2]

Fitur lainnya dari Aplikasi Program Manajemen Toko Buku adalah kemampuan mengelola penjualan. Sistem ini memungkinkan pemilik toko buku untuk melacak penjualan, termasuk tanggal penjualan, dan buku yang dibeli. Fitur ini membantu pemilik toko buku mengetahui buku mana yang laku di pasaran dan mana yang tidak.

Diharapkan dengan adanya suatu aplikasi penjualan dapat menangani permasalahan yang ada di toko tersebut. Aplikasi ini memudahkan pekerjaan Admin dalam pembuatan laporan dan memudahkan dalam memproses data pembelian dan penjualan.

Kesimpulannya, Aplikasi Program Manajemen Toko Buku adalah aplikasi yang dirancang untuk membantu pemilik toko buku mengelola inventaris, penjualan, dan pelanggan mereka dengan lebih efektif. Sistem ini memiliki sejumlah fitur yang berguna bagi pemilik toko buku. Dengan menggunakan sistem manajemen toko buku, pemilik toko buku dapat menghemat waktu dan meningkatkan operasional bisnisnya.

B. Unsur atau Konsep PBO Yang Dilibatkan

1. Inheritance

   Pewarisan merupakan konsep dasar dalam pemrograman berorientasi objek yang mendorong penggunaan kembali kode dan organisasi. Pewarisan memungkinkan kelas untuk berbagi sifat dan perilaku yang sama, sehingga menghasilkan kode yang lebih efisien dan dapat dirawat.

   Pewarisan dalam kode di atas terjadi antara kelas `ProgramManajemenTokoBuku` dan kelas `Application`. Kelas `ProgramManajemenTokoBuku` merupakan subkelas dari kelas `Application`, artinya kelas `ProgramManajemenTokoBuku` mewarisi semua properti dan metode dari kelas `Application`.

```
public class ProgramManajemenTokoBuku extends Application {
    private double x = 0;
    private double y = 0;
```

2. Polymorphism

   Polimorfisme adalah salah satu konsep dalam pemrograman berorientasi objek yang memungkinkan objek dari kelas yang berbeda untuk merespons dengan cara yang berbeda terhadap panggilan metode yang sama. Dalam konteks sistem manajemen toko buku (Aplikasi Program Manajemen Toko Buku),

   Dalam toko buku, dapat memiliki berbagai jenis buku, seperti novel, buku pelajaran, buku referensi, dan sebagainya. Meskipun semua ini adalah buku, mereka mungkin memiliki metode yang berbeda untuk mengambil informasi seperti harga, penulis, atau ISBN. Polimorfisme memungkinkan kita untuk menggunakan metode yang sama, misalnya "setTitle()", pada berbagai jenis buku, dan masing-masing jenis buku dapat merespons dengan informasi yang sesuai.

   Overriding: Metode login() di kelas FXMLDocumentController memiliki kode yang berbeda dari metode login() di kelas Controller. Ini karena metode login() di kelas FXMLDocumentController dirancang untuk menangani login untuk pengguna admin.

3. Encapsulation

   Adalah sebuah konsep yang menggabungkan data (variabel) dan metode yang bberoperasi pada data tersebut ke dalam kelas. Dalam Aplikasi Program Manajemen Toko Buku, encapsulation digunakan dalam melindungi data yang sensitif, seperti username dan password.

   Dalam Aplikasi Program Manajemen Toko Buku:Atribut (data) dalam kelas yang tidak boleh diakses secara langsung dari luar kelas harus dinyatakan sebagai private. Misalnya, atribut seperti harga buku atau jumlah stok harus dideklarasikan sebagai private sehingga mereka hanya dapat diakses melalui metode-metode kelas tersebut.

Kode ini menggunakan encapsulation dengan mendeklarasikan variabel x dan y sebagai variabel privat (private) dan mengaksesnya melalui metode-metode public (setter dan getter) di dalam kelas.

```
public class ProgramManajemenTokoBuku extends Application {
    private double x = 0;
    private double y = 0;

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(url:getClass().getResource(name:"FXMLDocument.fxml"));

        Scene scene = new Scene(parent: root);

        root.setOnMousePressed((MouseEvent event) ->{
            x = event.getSceneX();
            y = event.getSceneY();
        });

        root.setOnMouseDragged((MouseEvent event) ->{
            stage.setX(event.getScreenX() - x);
            stage.setY(event.getScreenY() - y);
```

4. Abstraction

   Adalah sebuah konsep yang melakukan penyembunyian detail implementasi dan hanya mengekspos fungsi yang relavan dari suatu objek. Dalam Aplikasi Program Manajemen Toko Buku, abstraction digunakan untuk membuat kelas yang umum seperti pada kelas buku.

   Ada elemen abstraksi dalam pembuatan metode-metode untuk berbagai tugas seperti login, pendaftaran, dan perpindahan antar form. Elemen abstraksi muncul dalam hal pemisahan tugas-tugas yang berbeda ke dalam metode-metode terpisah.

5. Class

   Class adalah blue print yang menciptakan objek pada PBO. Dalam Aplikasi Program Manajemen Toko Buku, kelas dapat mencakup buku, customer, pengguna, dll.

   Kelas utama adalah `FXMLDocumentController`. Selain itu, ada beberapa kelas JavaFX seperti `Stage`, `Scene`, dan `Alert` yang digunakan di dalamnya.

```
public class FXMLDocumentController implements Initializable {

    @FXML
    private Button close;

    @FXML
    private FontAwesomeIcon close_icon;
```

6. Object dan Method

   Object adalah Representasi konkret dari suatu class. Objek memiliki atribut yang ditentukan oleh class dan dapat menjalankan metode yang dimilikinya. Method adalah Fungsi atau perilaku yang dimiliki oleh objek atau class. Metode mendefinisikan tindakan atau operasi yang dapat dilakukan oleh objek tersebut.

   Pada kode diatas Objek-objek tidak terlihat secara eksplisit dalam kode, tetapi metode-metode seperti loginAdmin(), daftar(), dan lain-lain mewakili tindakan-tindakan yang dapat dilakukan pada objek.

```
public void loginAdmin(){

    if (username.getText().isEmpty() || password.getText().isEmpty()) {
        alert = new Alert(at: AlertType.ERROR);
        alert.setTitle(string: "Error ");
        alert.setHeaderText(string: null);
        alert.setContentText(string: "Username/Password Salah");
        alert.showAndWait();
    } else {
public void daftar() {

    if (su_username.getText().isEmpty() || su_password.getText().isEmpty()
            || su_pertanyaan.getSelectionModel().getSelectedItem() == null
            || su_jawaban.getText().isEmpty()) {
        alert = new Alert(at: AlertType.ERROR);
        alert.setTitle(string: "Error ");
        alert.setHeaderText(string: null);
        alert.setContentText(string: "Lengkapi Username, Password, Pertanyaan
        alert.showAndWait();
```

7. Property

   Adalah variabel atau atribut yang dimiliki suatu objek dalam suatu Class. Dalam Aplikasi Program Manajemen Toko Buku, properti dapat berupa judul, penerbit, tahun terbit, harga buku, jumlah buku, dll.

   Properti-properti termasuk variabel-variabel seperti username, password, questionList, dan lain-lain. Mereka digunakan untuk menyimpan data yang berkaitan dengan objek.

8. Construcctor

   Adalah sebuah metode khusus yang digunakan untuk menginisialisasi objek saat objek tersebut dibuat. Dalam Aplikasi Program Manajemen Toko Buku, constructor digunakan dalam membuat objek buku baru dengan properti tertentu.

9. Visibility

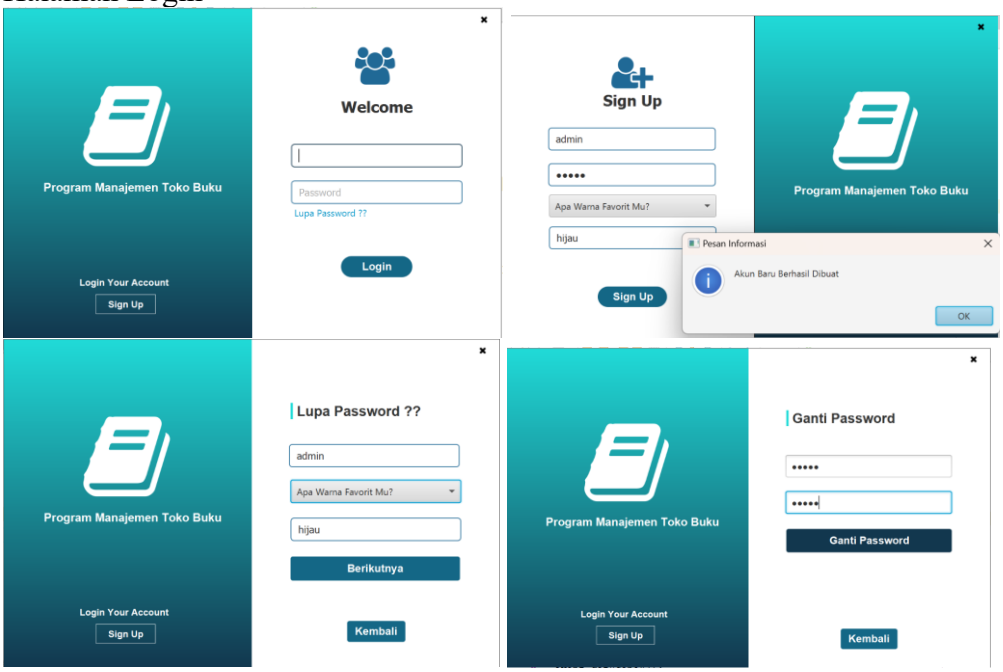
   Adalah aturan yang mengatur sejauh mana suatu kelas, atribut atau metode dapat diakses oleh kelas-kelas lain dalam program. . Dalam Aplikasi Program Manajemen Toko Buku, visibilitas digunakan untuk mengontrol akses suatu kelas terhadap atribut atau metode yang ada pada kelas lain.

   Variabel x dan y memiliki tingkat visibilitas private, yang berarti hanya dapat diakses di dalam kelas itu sendiri. Metode-metode dan properti-properti lainnya tidak diberi tingkat visibilitas tertentu, sehingga secara default menggunakan tingkat visibilitas package-private.

```
public class ProgramManajemenTokoBuku extends Application {
    private double x = 0;
    private double y = 0;
```

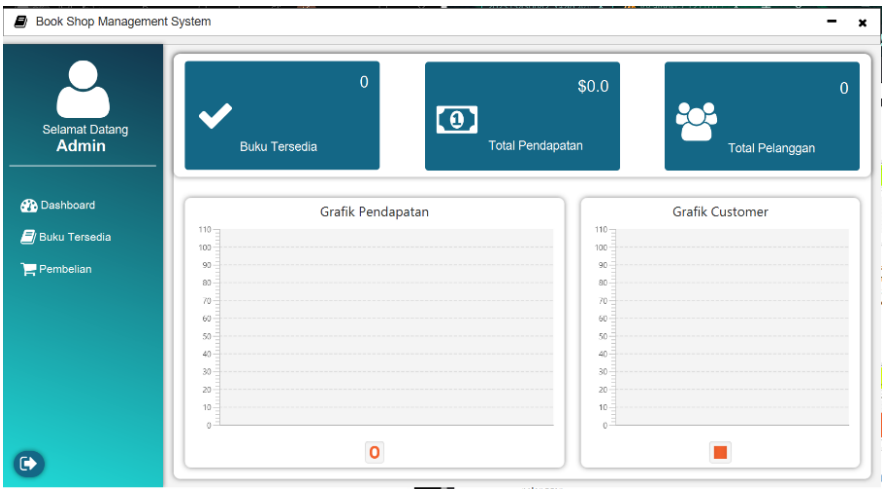
C. Penjelasan Aplikasi

1. Halaman Login



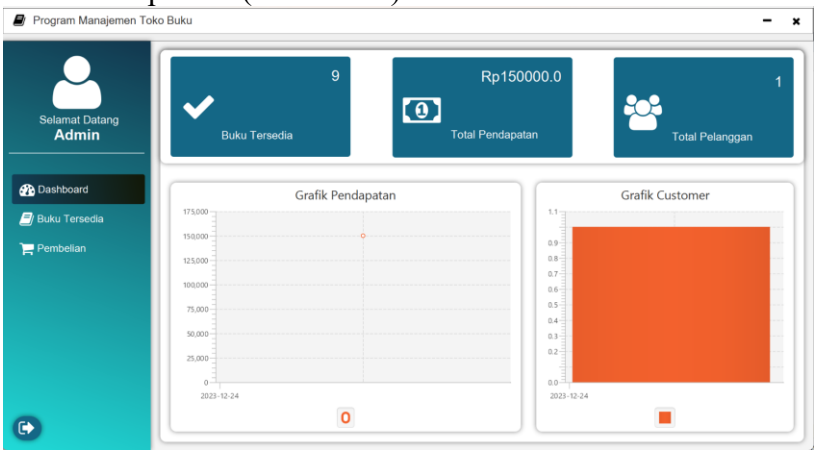
Halaman login ini adalah halaman awal sebelum masuk kedalam halaman aplikasi utamanya. Pada halaman ini user akan diminta untuk memasukkan username dan password yang telah terdaftar sebelumnya. Jika belum mempunyai akun user dapat membuat akun terlebih dahulu dengan menekan tombol sign up dan melengkapi kolom yang diminta pada bagian sign up. Jika seandainya sewaktu-waktu user lupan akan password akunnya, user dapat meng klik link lupa password, lalu user akan diminta untuk mengisi username dan menjawab pertanyaan, setelah valid maka user diminta untuk memasukkan password baru dan mengkonfirmasinya

2. Halaman Aplikasi



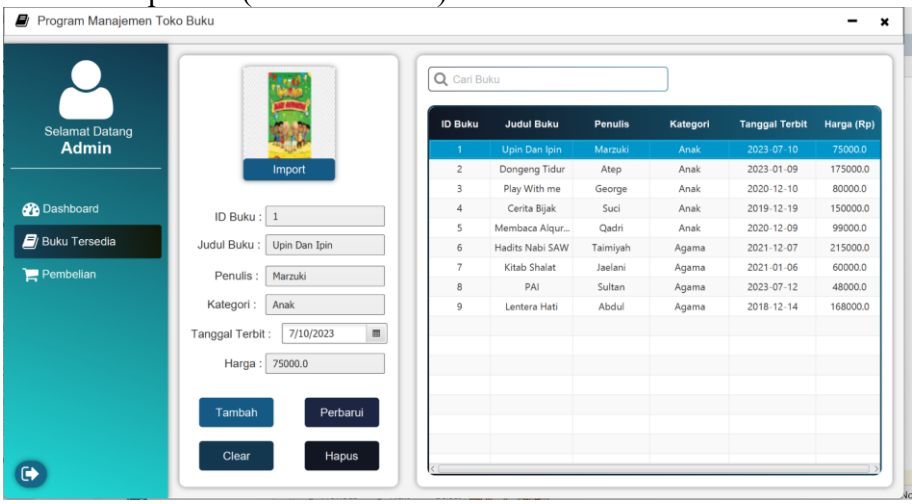
Halaman aplikasi adalah halaman yang akan muncul ketika user telah berhasil memasukkan username dan password dengan benar. Pada halaman ini user dapat melihat jumalh buku yang tersedia, total pendapatan, dan total pelanggan beserta grafiknya. Dan juga pada kiri bawah terdapat tombol log out dari aplikasi.

3. Halaman Aplikasi (Dashboard)



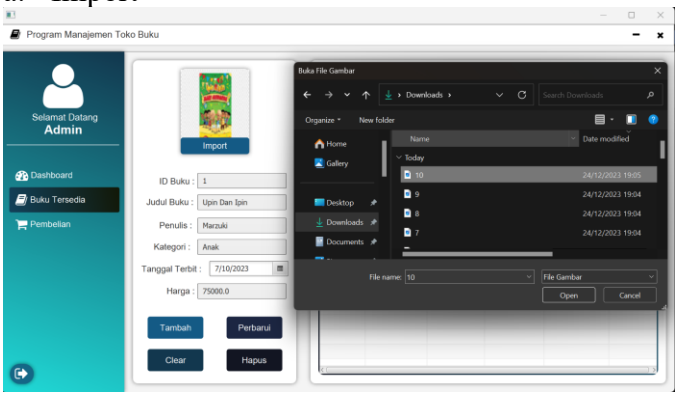
Bagian yang pertama adalah Dashboard / halaman utama dari aplikasi ini. User dapat melihat total buku yang tersedia, total pendapatan, dan total pelanggan serta grafiknya.

4. Halaman Aplikasi (Buku Tersedia)



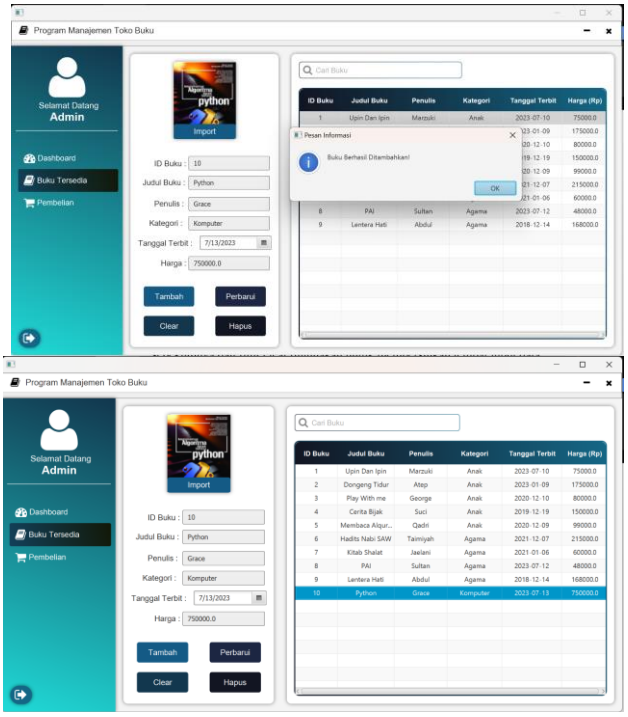
Bagian kedua adalah halaman yang menampilkan daftar buku yang tersedia beserta dengan penulis, judul, kategori tahun terbit hingga harga dari buku tersebut. Terdapat juga beberapa fitur yaitu import, tambah, perbarui, clear dan hapus. Yang mana keseluruhannya digunakan untuk mengatur data buku tersebut.

a. Import



Fitur ini digunakan untuk menambahkan gambar cover buku ke dalam aplikasi

b. Tambah

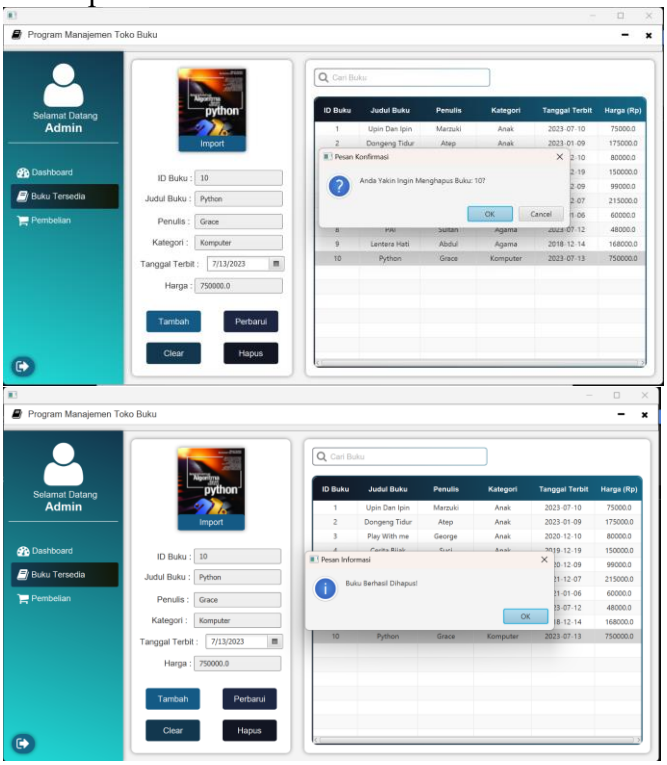


Fitur ini digunakan untuk menambahkan data buku baru

c. Perbarui dan Clear

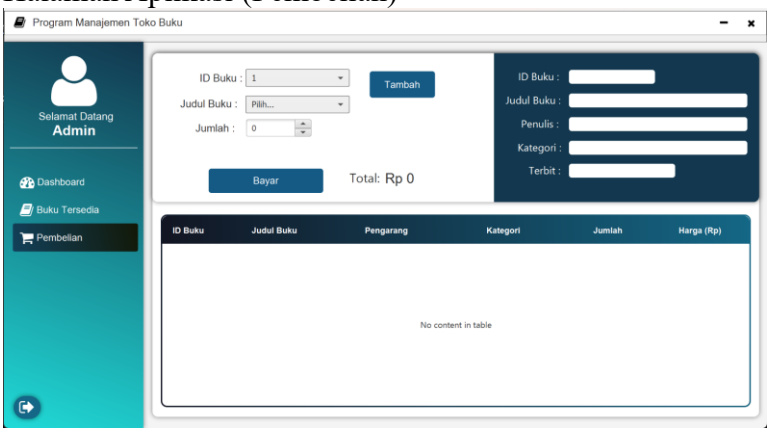
Fitur perbarui digunkan untuk memperbarui data buku yang telah ditambahkan sebekumnya dan fitur clear digunakan untuk membersihkan tempat input data buku

d. Hapus

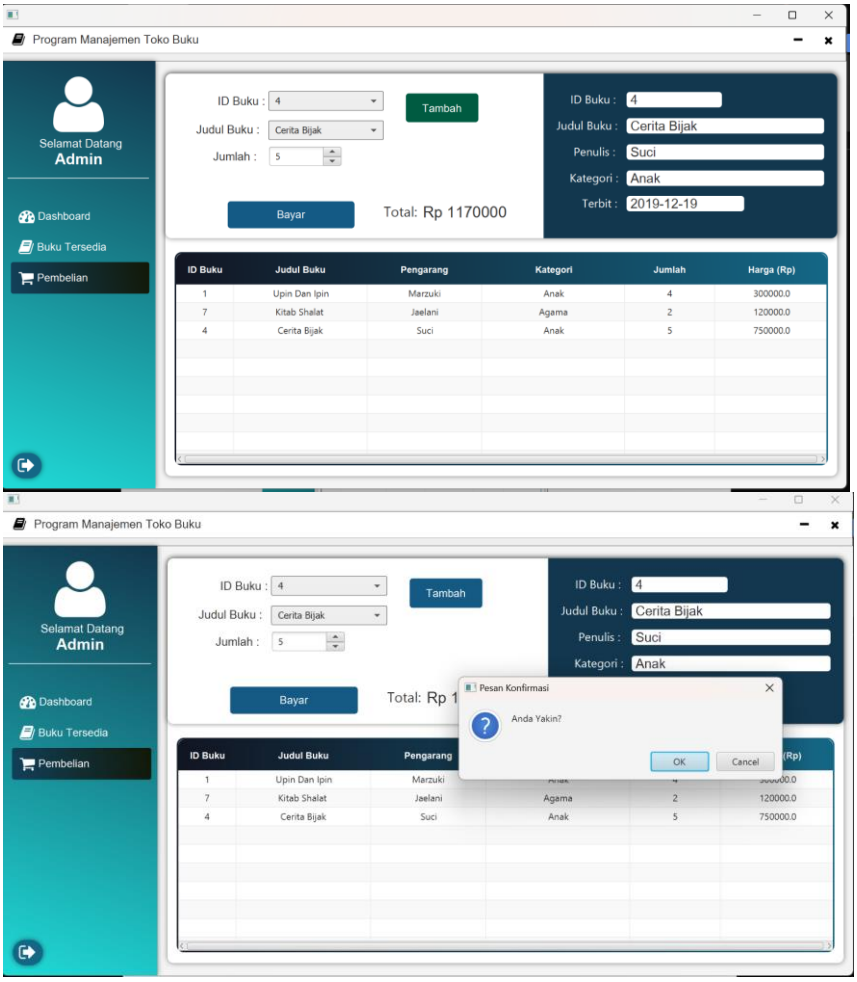


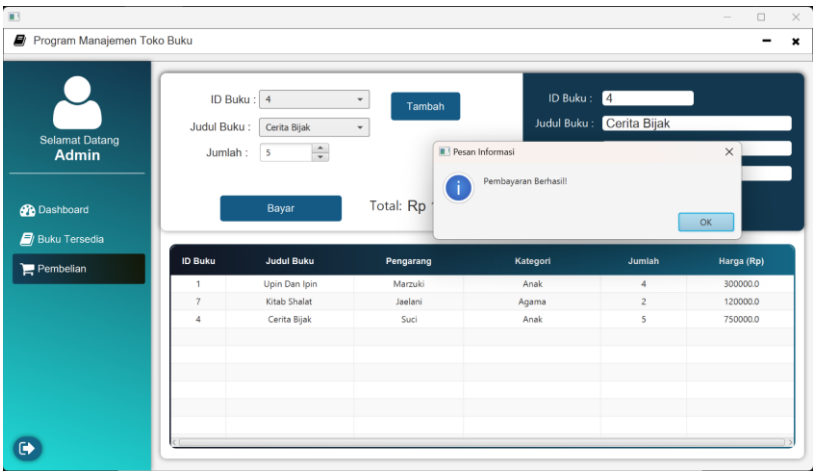
Fitur ini digunakan untuk menghapus buku dari daftar buku yang tersedia apabila buku tersebut tidak dibutuhkan lagi

5. Halaman Aplikasi (Pembelian)



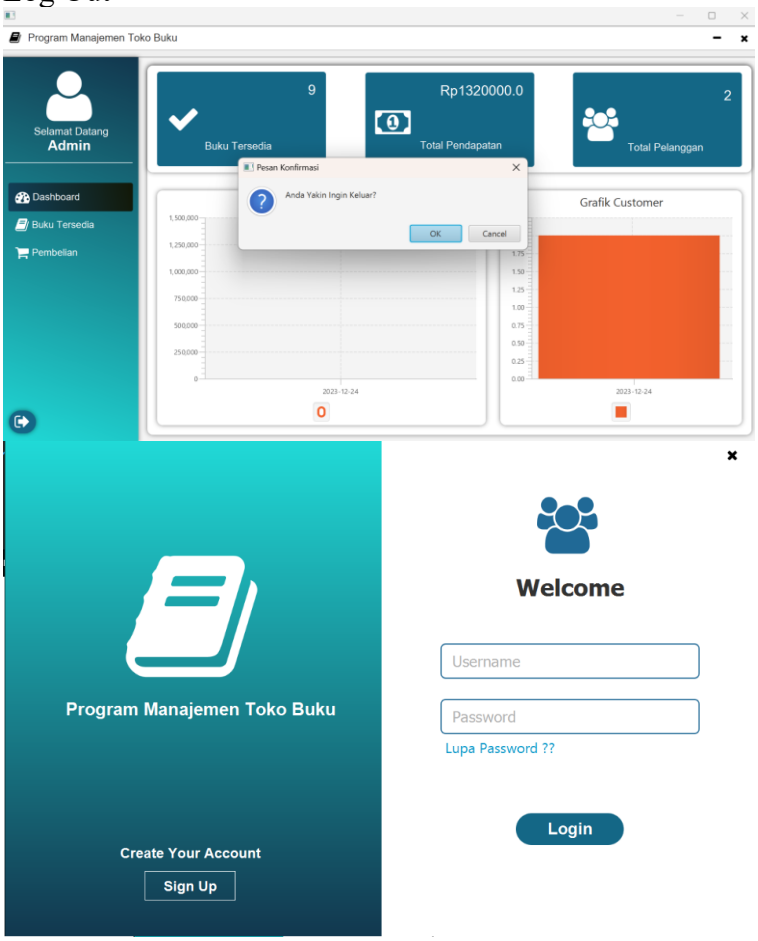
Halaman ini adalah halaman untuk melakukan transakasi pembelian buku yang diinginkan oleh konsumen. Pada halaman ini terdapat fitur tambah yang berfungsi untuk menambah buku apa saja yang ingin dibeli konsumen. Setelah semua buku yang diinginkan ditambah, daftar buku akan keluar pada tabel dibawah, dan total harga juga akan muncul, konsumen hanya perlu menekan bayar untuk menyelesaikan transaksi.





Setelah user menekan bayar akan muncul pop uop konfirmasi dan setelah menekan oke akan muncul kembali pop up yang menyatakan bahwa pembayaran berhasil

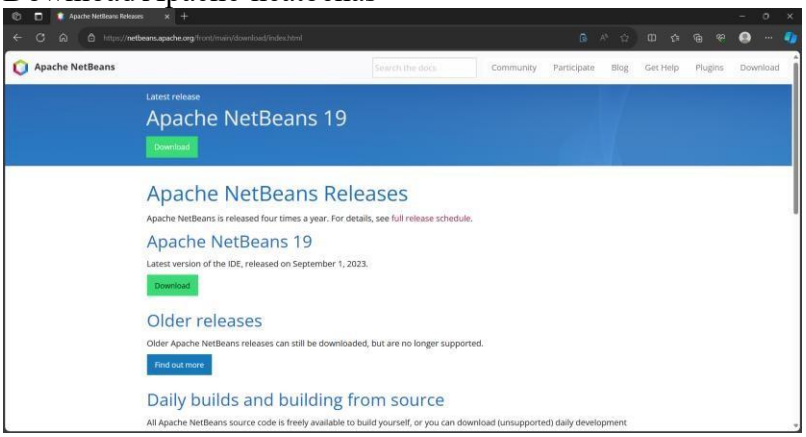
6. Log Out



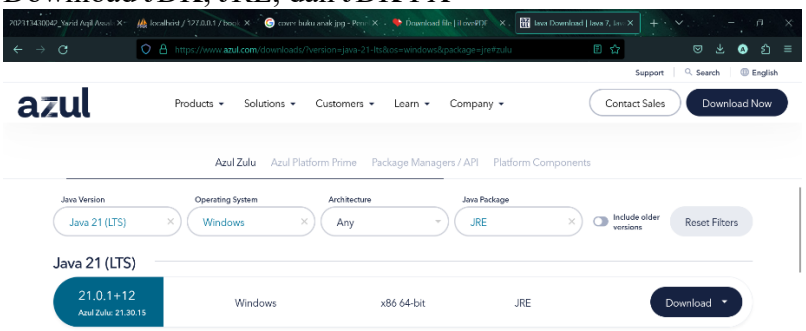
Jika ingin keluar dari halaman aplikasi cukup dengan menekan tombol yang ada pada pojok kiri bawah, lalu akan muncul pesan apakah yakin ingin keluar lalu klik OK, maka kita akan kembali ke halaman login

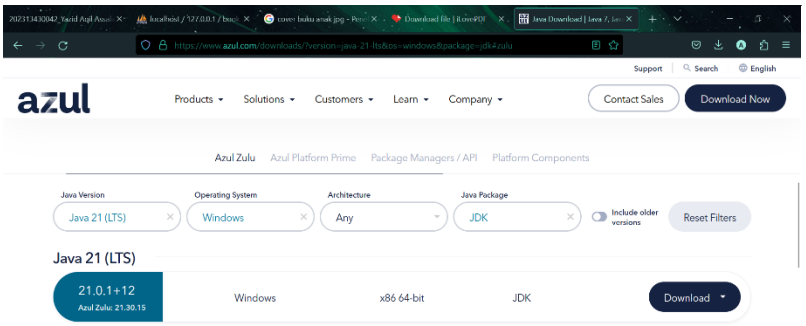
D. Langkah-Langkah Pembuatan Aplikasi

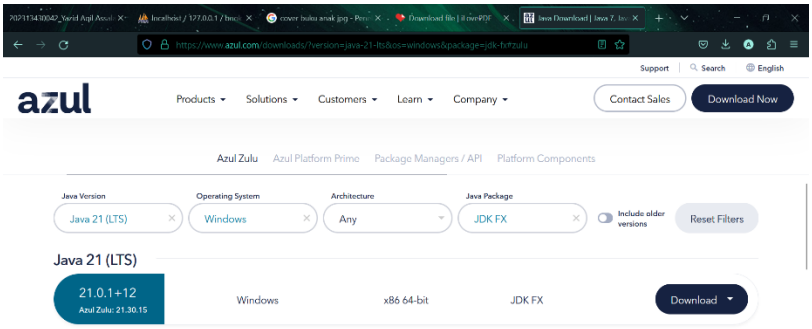
1. Download Apache-neatbenas



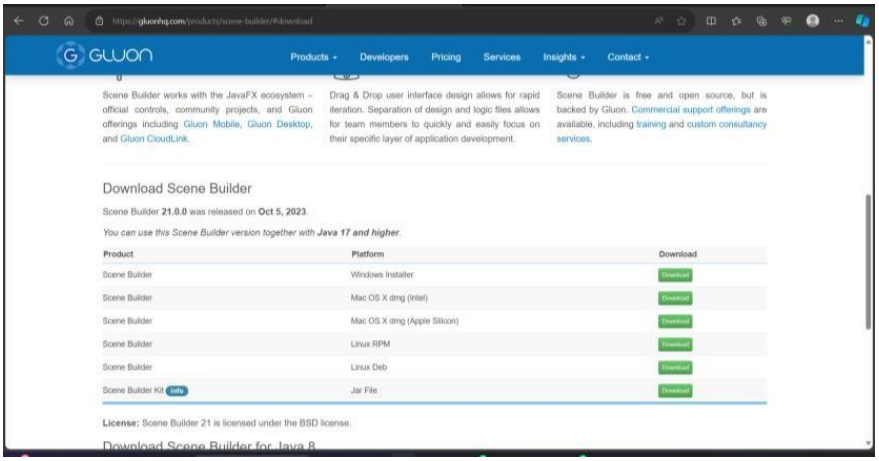
2. Download JDK, JRE, dan JDK FX



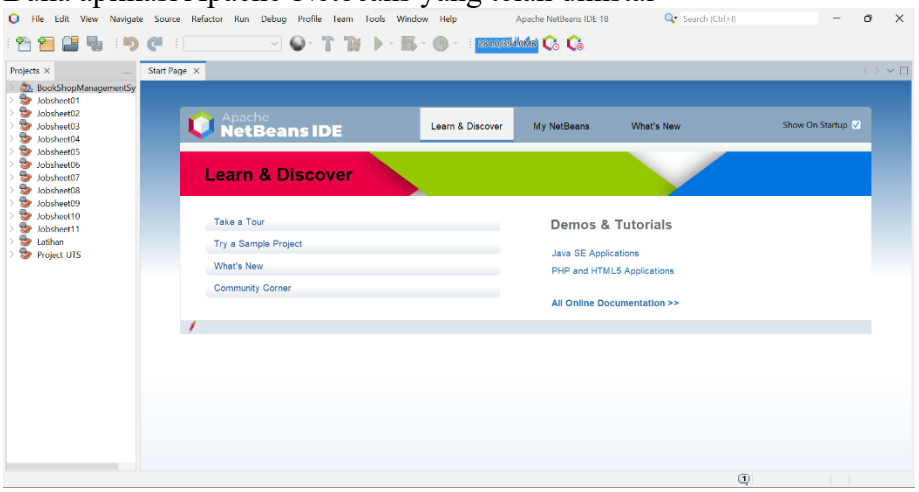




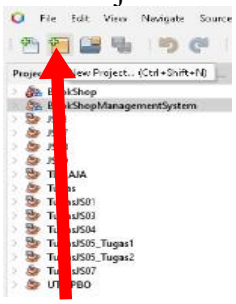
3. Download Scene Builder



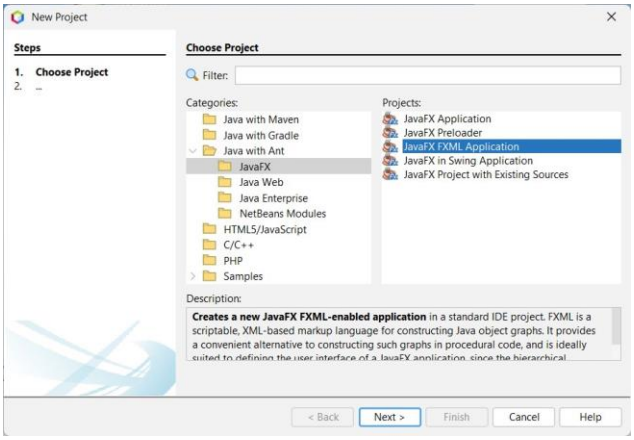
4. Instal semua aplikasi yang telah didownload tadi
5. Buka aplikasi Apache-Netbeans yang telah diinstal



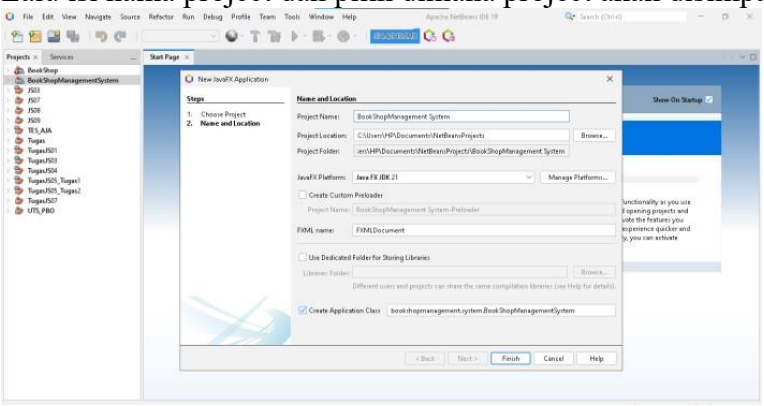
6. Buat Project baru



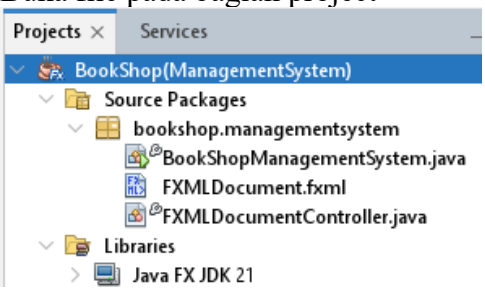
Lalu Pilih Java with Ant > JavaFX > JavaFX FXML Aplication



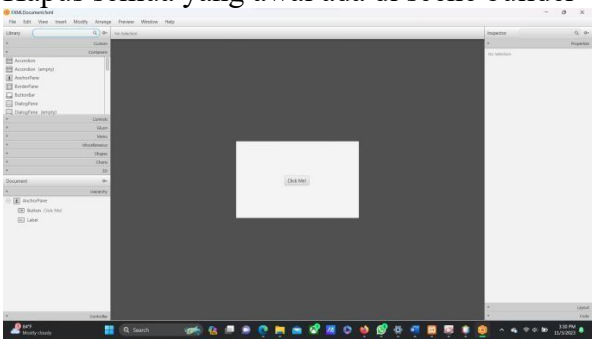
7. Lalu isi nama project dan pilih dimana project akan disimpan



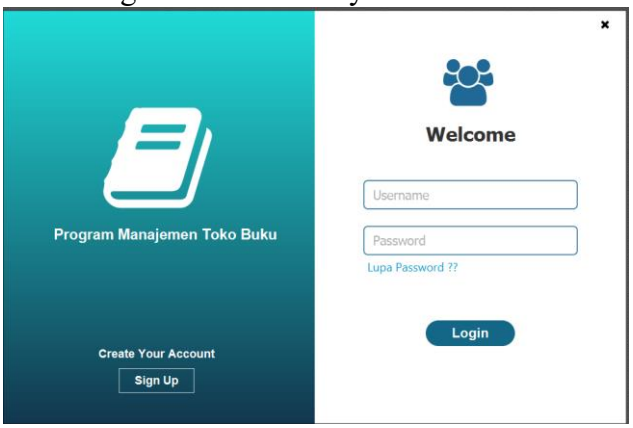
8. Buka file pada bagian project



9. Masuk ke FXMLDocument.fxml dengan klik dua kali pada file
10. Hapus semua yang awal ada di scene builder



11. Selanjutnya mulai dengan membuat bagian “Halama Login”. Dimana proses pengguna baru dapat membuat akun baru. Selamanya proses pendaftaran, pengguna diminta untuk memberikan informasi seperti nama pengguna serta kata sandi. Buat seperti halaman di bawah ini dan hubungkan file desainLogin.css ke dalamnya

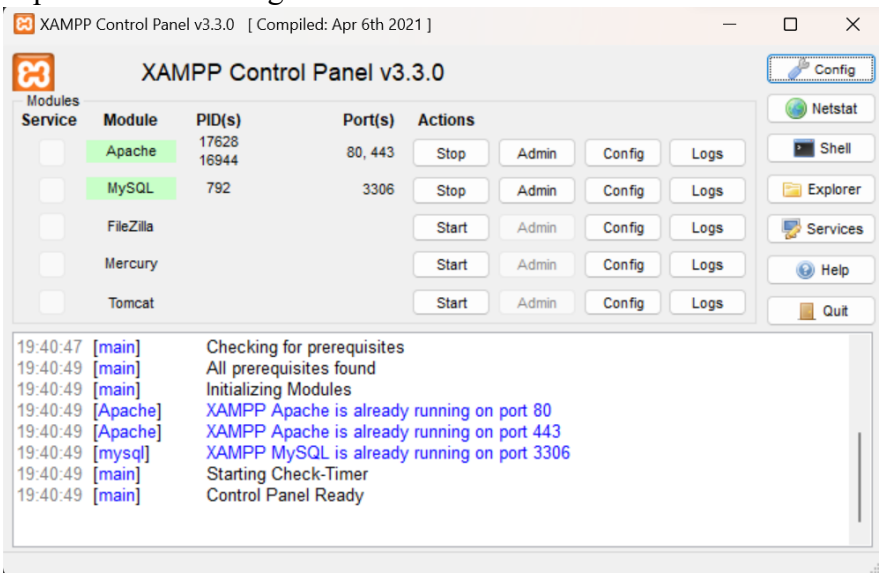


Source code tampilan desainLogin.css

```
.form-kiri{
    -fx-background-color:linear-gradient(to top, #12374e,
#20dbd8);
    -fx-border-color: #000;
    -fx-border-width: .4px 0px .4px .4px;
}
.form-kanan{
    -fx-background-color:#fff;
    -fx-border-color: #000;
    -fx-border-width: .4px .4px .4px 0px;
}
.tombol_close{
    -fx-background-color: transparent;
    -fx-cursor:hand;
}
.tombol_close:hover{
    -fx-background-color:#b10c0c;
}

.label_forgot{
    -fx-border-color: #20dbd8;
    -fx-border-width: 0 0 0 3px;
    -fx-padding: 0 0 0 5px;
}

.teks{
    -fx-background-color: transparent;
    -fx-background-radius:4px;
    -fx-font-size:13px;
    -fx-border-color: linear-gradient(to top, #206996,
#206996);
    -fx-border-width: 1px;
    -fx-border-radius: 4px;
    -fx-font-family: Tahoma;
}
.teks:focused{
    -fx-background-color: #fff;
    -fx-border-color:linear-gradient(to top right, #12374e,
#12374e);
    -fx-border-width: 1px;
}
.tombol_login{
    -fx-background-color:#146786;
    -fx-background-radius: 50px;
    -fx-cursor:hand;
    -fx-text-fill: #fff;
    -fx-font-size: 14px;
    -fx-font-weight: bold;
```

```
    -fx-font-family: "Arial";
}
.tombol_login:hover{
    -fx-background-color:#12374e;
}

.tombol_next{
    -fx-background-color:#146786;
    -fx-cursor:hand;
    -fx-text-fill: #fff;
    -fx-font-size: 14px;
    -fx-font-weight: bold;
    -fx-font-family: "Arial";
}
.tombol_next:hover{
    -fx-background-color:#12374e;
}

.tombol_daftar{
    -fx-background-color:transparent;
    -fx-cursor:hand;
    -fx-text-fill: #fff;
    -fx-border-color: #fff;
    -fx-border-width: .5px;
}

.tombol_daftar:hover{
    -fx-background-color:#ffff;
    -fx-text-fill:#12374e;
}
```

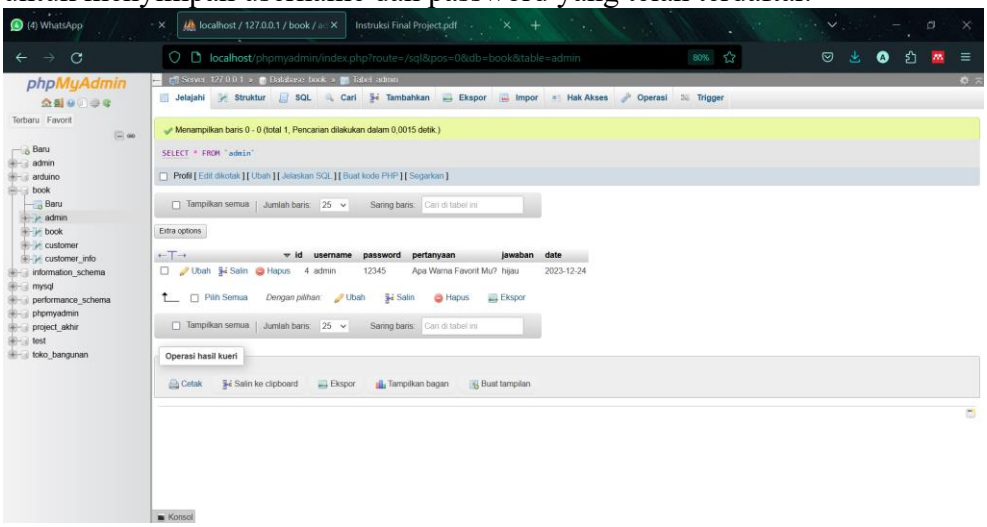
12. Selanjutnya buat file untuk mengkoneksikan database dengan cara membuat file baru dengan diberi nama database.java. Dan hidupkan XAMPP agar file dapat terkoneksi dengan database localhost



Source code database.java

```
package programmanajementokobuku;
import java.sql.Connection;
import java.sql.DriverManager;

public class database {

    public static Connection connectDb(){

        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection connect =
DriverManager.getConnection("jdbc:mysql://localhost/book",
"root", ""); // address, database username, database
password
            return connect;
        }catch(Exception e){e.printStackTrace();}
        return null; // LETS MAKE OUR DATABASE  : ) book is
our database name : )
    }

}
```

Selanjutnya buka phpMyAdmin pada browser lalu buatlah database baru dengan nama “book” dan buatlah tabel baru dengn nama “admin”, tabel ini untuk menyimpan username dan password yang telah terdaftar.



13. Pada file FXMLDocumentController.java masukkan masukkan kode yang terhubung dengan FXMLDocument.fxml agar halaman login dapat dijalankan agar aplikasi dapat dijalankan.

Source code FXMLDocumentController.java

```
package programmanajementokobuku;

import de.jensd.fx.glyphs.fontawesome.FontAwesomeIcon;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.ResourceBundle;
import javafx.animation.TranslateTransition;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Hyperlink;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import javafx.util.Duration;

public class FXMLDocumentController implements Initializable
{

    @FXML
    private Button close;

    @FXML
    private FontAwesomeIcon close_icon;

    @FXML
    private Label edit_label;
```

```
    @FXML
    private Hyperlink forgotPass;

    @FXML
    private AnchorPane gantiPw_form;

    @FXML
    private Button gp_gantiPassBtn;

    @FXML
    private Button gp_kembaliBtn;

    @FXML
    private PasswordField gp_konfirPass;

    @FXML
    private PasswordField gp_passBaru;

    @FXML
    private Button loginBtn;

    @FXML
    private AnchorPane loginForm;

    @FXML
    private Button lp_brktBtn;

    @FXML
    private TextField lp_jawaban;

    @FXML
    private Button lp_kembaliBtn;

    @FXML
    private ComboBox<?> lp_pertanyaan;

    @FXML
    private AnchorPane lupaPassword_form;

    @FXML
    private AnchorPane main_form;

    @FXML
    private PasswordField password;

    @FXML
    private AnchorPane signup_form;

    @FXML
    private TextField su_jawaban;
```

```
    @FXML
    private PasswordField su_password;

    @FXML
    private ComboBox<?> su_pertanyaan;

    @FXML
    private Button su_signupBtn;

    @FXML
    private TextField su_username;

    @FXML
    private AnchorPane sub_form;

    @FXML
    private Button sub_loginBtn;

    @FXML
    private Button sub_signupBtn;

    @FXML
    private TextField username;

    @FXML
    private TextField lp_username;



    private Connection connect;
    private PreparedStatement prepare;
    private ResultSet hasil;
    private Alert alert;

    private double x = 0;
    private double y = 0;

    public void loginAdmin(){

        if (username.getText().isEmpty() ||
password.getText().isEmpty()) {
            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error ");
            alert.setHeaderText(null);
            alert.setContentText("Username/Password Salah");
            alert.showAndWait();
        } else {
```

```
            String selctData = "SELECT username, password
FROM admin WHERE username = ? and password = ?";

            connect = database.connectDb();

            try {

                prepare =
connect.prepareStatement(selctData);
                prepare.setString(1, username.getText());
                prepare.setString(2, password.getText());

                hasil = prepare.executeQuery();
                // IF SUCCESSFULLY LOGIN, THEN PROCEED TO
ANOTHER FORM WHICH IS OUR MAIN FORM
                if (hasil.next()) {
                    // TO GET THE USERNAME THAT USER USED
                    getData.username = username.getText();

                    alert = new
Alert(AlertType.INFORMATION);
                    alert.setTitle("Pesan Informasi");
                    alert.setHeaderText(null);
                    alert.setContentText("Login Berhasil!");
                    alert.showAndWait();

                    // LINK YOUR DASHBOARD FORM
                    Parent root =
FXMLLoader.load(getClass().getResource("Dashboard.fxml"));

                    Stage stage = new Stage();
                    Scene scene = new Scene(root);



                    stage.setScene(scene);
                    stage.show();

                    loginBtn.getScene().getWindow().hide();

                } else { // IF NOT, THEN THE ERROR MESSAGE
WILL APPEAR
                    alert = new Alert(AlertType.ERROR);
                    alert.setTitle("Error ");
                    alert.setHeaderText(null);
                    alert.setContentText("Username/Password
Salah");
                    alert.showAndWait();
                }
```

```
            } catch (Exception e) {
                e.printStackTrace();
            }

        }
    }

    public void daftar() {

        if (su_username.getText().isEmpty() ||
su_password.getText().isEmpty()
                ||
su_pertanyaan.getSelectionModel().getSelectedItem() == null
                || su_jawaban.getText().isEmpty()) {
            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error ");
            alert.setHeaderText(null);
            alert.setContentText("Lengkapi Username,
Password, Pertanyaan dan Jawaban");
            alert.showAndWait();
        } else {

            String regData = "INSERT INTO admin (username,
password, pertanyaan, jawaban, date) "
                    + "VALUES(?,?,?,?,?)";
            connect = database.connectDb();

            try {
                // CHECK IF THE USERNAME IS ALREADY RECORDED
                String checkUsername = "SELECT username FROM
admin WHERE username = '"
                        + su_username.getText() + "'";

                prepare =
connect.prepareStatement(checkUsername);
                hasil = prepare.executeQuery();

                if (hasil.next()) {
                    alert = new Alert(AlertType.ERROR);
                    alert.setTitle("Error ");
                    alert.setHeaderText(null);
                    alert.setContentText(su_username.getText
() + " Username Telah Digunakan");
                    alert.showAndWait();
                } else if (su_password.getText().length() >
8) {
                    alert = new Alert(AlertType.ERROR);
                    alert.setTitle("Error ");
                    alert.setHeaderText(null);
```

```
                    alert.setContentText("Maksimal Panjang
Password 8 Karakter");
                    alert.showAndWait();
                } else {
                    prepare =
connect.prepareStatement(regData);
                    prepare.setString(1,
su_username.getText());
                    prepare.setString(2,
su_password.getText());
                    prepare.setString(3, (String)
su_pertanyaan.getSelectionModel().getSelectedItem());
                    prepare.setString(4,
su_jawaban.getText());

                    Date date = new Date();
                    java.sql.Date sqlDate = new
java.sql.Date(date.getTime());
                    prepare.setString(5,
String.valueOf(sqlDate));
                    prepare.executeUpdate();

                    alert = new
Alert(AlertType.INFORMATION);
                    alert.setTitle("Pesan Informasi");
                    alert.setHeaderText(null);
                    alert.setContentText("Akun Baru Berhasil
Dibuat");
                    alert.showAndWait();

                    su_username.setText("");
                    su_password.setText("");
                    su_pertanyaan.getSelectionModel().clearS
election();
                    su_jawaban.setText("");

                    TranslateTransition slider = new
TranslateTransition();

                    slider.setNode(sub_form);
                    slider.setToX(0);
                    slider.setDuration(Duration.seconds(.5))
;

                    slider.setOnFinished((ActionEvent e) ->
{
                        sub_loginBtn.setVisible(false);
                        sub_signupBtn.setVisible(true);
                    });
```

```
                    slider.play();
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

    }

    private String[] questionList = {"Berapa Tanggal Lahir
Mu?", "Apa Warna Favorit Mu?", "Apa Program Studi Mu?"};

    public void regLquestionList() {
        List<String> listQ = new ArrayList<>();

        for (String data : questionList) {
            listQ.add(data);
        }

        ObservableList listData =
FXCollections.observableArrayList(listQ);
        su_pertanyaan.setItems(listData);
    }

    public void switchForgotPass() {
        lupaPassword_form.setVisible(true);
        loginForm.setVisible(false);

        forgotPassQuestionList();
    }

    public void lanjutBtn() {

        if (lp_username.getText().isEmpty() ||
lp_pertanyaan.getSelectionModel().getSelectedItem() == null
                || lp_jawaban.getText().isEmpty()) {

            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error ");
            alert.setHeaderText(null);
            alert.setContentText("Silahkan Lengkapi Bagian
yang Kosong");
            alert.showAndWait();

        } else {

            String selectData = "SELECT username,
pertanyaan, jawaban FROM admin WHERE username = ? AND
pertanyaan = ? AND jawaban = ?";
```

```
            connect = database.connectDb();

            try {

                prepare =
connect.prepareStatement(selectData);
                prepare.setString(1, lp_username.getText());
                prepare.setString(2, (String)
lp_pertanyaan.getSelectionModel().getSelectedItem());
                prepare.setString(3, lp_jawaban.getText());

                hasil = prepare.executeQuery();

                if (hasil.next()) {
                    gantiPw_form.setVisible(true);
                    lupaPassword_form.setVisible(false);
                } else {
                    alert = new Alert(AlertType.ERROR);
                    alert.setTitle("Error ");
                    alert.setHeaderText(null);
                    alert.setContentText("Jawaban Anda
Salah!!");
                    alert.showAndWait();
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public void gantiPassBtn() {

        if (gp_passBaru.getText().isEmpty() ||
gp_konfirPass.getText().isEmpty()) {
            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error ");
            alert.setHeaderText(null);
            alert.setContentText("Silahkan Lengkapi Bagian
yang Kosong");
            alert.showAndWait();
        } else {

            if
(gp_passBaru.getText().equals(gp_konfirPass.getText())) {
                String getDate = "SELECT date FROM admin
WHERE username = '"
                        + lp_username.getText() + "'";

                connect = database.connectDb();
```

```
                try {

                    prepare =
connect.prepareStatement(getDate);
                    hasil = prepare.executeQuery();

                    String date = "";
                    if (hasil.next()) {
                        date = hasil.getString("date");
                    }

                    String updatePass = "UPDATE admin SET
password = '"
                            + gp_passBaru.getText() + "',
pertanyaan = '"
                            +
lp_pertanyaan.getSelectionModel().getSelectedItem() + "',
jawaban = '"
                            + lp_jawaban.getText() + "',
date = '"
                            + date + "' WHERE username = '"
                            + lp_username.getText() + "'";

                    prepare =
connect.prepareStatement(updatePass);
                    prepare.executeUpdate();

                    alert = new
Alert(AlertType.INFORMATION);
                    alert.setTitle("Pesan Informasi");
                    alert.setHeaderText(null);
                    alert.setContentText("Password Berhasil
Diganti!");
                    alert.showAndWait();

                    loginForm.setVisible(true);
                    gantiPw_form.setVisible(false);

                    // TO CLEAR FIELDS
                    gp_konfirPass.setText("");
                    gp_passBaru.setText("");
                    lp_pertanyaan.getSelectionModel().clearS
election();
                    lp_jawaban.setText("");
                    lp_username.setText("");

                } catch (Exception e) {
                    e.printStackTrace();
                }
```

```
            } else {
                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error ");
                alert.setHeaderText(null);
                alert.setContentText("Tidak Cocok");
                alert.showAndWait();
            }
        }
    }

    public void forgotPassQuestionList() {

        List<String> listQ = new ArrayList<>();

        for (String data : questionList) {
            listQ.add(data);
        }

        ObservableList listData =
FXCollections.observableArrayList(listQ);
        lp_pertanyaan.setItems(listData);

    }

    public void kembaliHalamanLogin(){
        loginForm.setVisible(true);
        lupaPassword_form.setVisible(false);
    }

    public void kembaliHalamanLupapass(){
        lupaPassword_form.setVisible(true);
        gantiPw_form.setVisible(false);
    }

    public void geserDaftar(ActionEvent event) {

        TranslateTransition slider = new
TranslateTransition();

        if (event.getSource() == sub_signupBtn) {
            slider.setNode(sub_form);
            slider.setToX(320);
            slider.setDuration(Duration.seconds(.5));

            slider.setOnFinished((ActionEvent e) -> {
                edit_label.setText("Login Your Account");
                sub_loginBtn.setVisible(true);
                sub_signupBtn.setVisible(false);

                lupaPassword_form.setVisible(false);
```

```
                loginForm.setVisible(true);
                gantiPw_form.setVisible(false);

                regLquestionList();
            });

            slider.play();
        } else if (event.getSource() == sub_loginBtn) {
            slider.setNode(sub_form);
            slider.setToX(0);
            slider.setDuration(Duration.seconds(.5));

            slider.setOnFinished((ActionEvent e) -> {
                edit_label.setText("Create Your Account");
                sub_loginBtn.setVisible(false);
                sub_signupBtn.setVisible(true);

                lupaPassword_form.setVisible(false);
                loginForm.setVisible(true);
                gantiPw_form.setVisible(false);
            });

            slider.play();
        }
    }

    public void keluar(){
        javafx.application.Platform.exit();
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

}
```

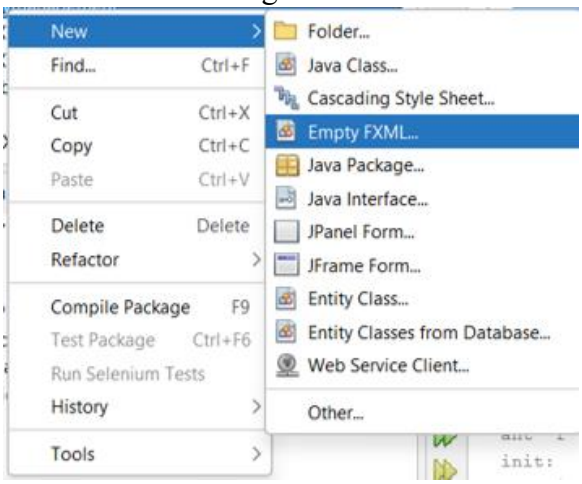
14. Langkah selanjutnya pada file ProgramManajemenTokoBuku.java masukkan kode yang berhubungan dengan controller yang telah dibuat sebelumnya agar aplikasi dapat running

    Source Code ProgramManajemenTokoBuku.java

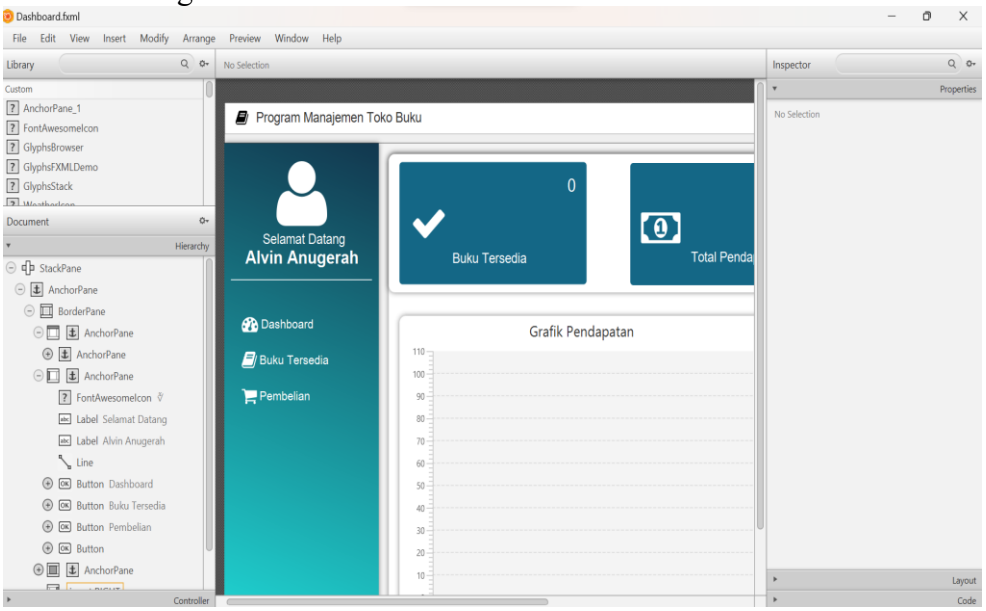
```
package programmanajementokobuku;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
```

```
import javafx.scene.input.MouseEvent;
import javafx.stage.Stage;
import javafx.stage.StageStyle;


public class ProgramManajemenTokoBuku extends Application {
    private double x = 0;
    private double y = 0;

    @Override
    public void start(Stage stage) throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"))
;

        Scene scene = new Scene(root);

        root.setOnMousePressed((MouseEvent event) ->{
            x = event.getSceneX();
            y = event.getSceneY();
        });

        root.setOnMouseDragged((MouseEvent event) ->{
            stage.setX(event.getScreenX() - x);
            stage.setY(event.getScreenY() - y);

            stage.setOpacity(.8);
        });

        root.setOnMouseReleased((MouseEvent event) ->{
            stage.setOpacity(1);
        });

        stage.initStyle(StageStyle.TRANSPARENT);

        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

15. Langkah selanjutnya yaitu membuat halaman dashboard yaitu dengan buatkan file FXML baru dengan nama file Dashboard.fxml



Buatlah halaman dashboard sesuai dengan keinginan dan kebutuhan yang diminta dan tambahkan sedikit kreativitas dan hubungkan dengan file css dasboardDesign.css



16. Selanjutnya buat file css baru dengan nama dashboardDesign.css, agar tampilan halaman dashboard menjadi lebih menarik

Source code dashboardDesign.css

```
.top-form{
    -fx-background-color: #fff;
    -fx-border-color: #000;
    -fx-border-width: .4px .4px .2px .4px;
}
.semi-top-form{
    -fx-background-color: #efefef;
    -fx-border-color: #000;
    -fx-border-width: .2px .4px .4px .4px;
}
```

```
.close{
    -fx-background-color: transparent;
    -fx-cursor:hand;
}
.close:hover{
    -fx-background-color:#b10c0c;
}
.minimize{
    -fx-background-color: transparent;
    -fx-cursor:hand;
}
.minimize:hover{
    -fx-background-color: #ddd;
}
.nav-form{
    -fx-background-color:linear-gradient(to top right,
#20dbd8, #12374e);
}
.nav-btn{
    -fx-background-color:transparent;
    -fx-cursor:hand;
    -fx-font-size: 14px;
    -fx-text-fill: #fff;
    -fx-font-family: Arial;
    -fx-alignment: CENTER-LEFT;
}
.sign-out{
    -fx-background-color: #146786;
    -fx-cursor:hand;
    -fx-background-radius: 20px;
}
.sign-out:hover{
    -fx-background-color: #121522;
}
.shadow{
    -fx-effect: dropshadow(three-pass-box,rgba(0,0,0,0.5),
8,0,0,0);
}
.white-bg{
    -fx-background-color: #fff;
    -fx-background-radius: 8px;
}
.card{
    -fx-background-color: linear-gradient(to top, #146786,
#146786);
    -fx-background-radius: 4px;
}
.add-btn{
    -fx-background-color: #145a85;
    -fx-background-radius: 4px;
```

```
    -fx-cursor:hand;
    -fx-font-size: 14px;
    -fx-font-family: Arial;
    -fx-text-fill: #fff;
}
.add-btn:hover{
    -fx-background-color: #005B41;
}
.update-btn{
    -fx-background-color: #1c2642;
    -fx-background-radius: 4px;
    -fx-cursor:hand;
    -fx-font-size: 14px;
    -fx-font-family: Arial;
    -fx-text-fill: #fff;
}
.update-btn:hover{
    -fx-background-color: #810CA8;
}
.clear-btn{
    -fx-background-color: #12374e;
    -fx-background-radius: 4px;
    -fx-cursor:hand;
    -fx-font-size: 14px;
    -fx-font-family: Arial;
    -fx-text-fill: #fff;
}
.clear-btn:hover{
    -fx-background-color: #8B9A46;
}
.delete-btn{
    -fx-background-color: #121522;
    -fx-background-radius: 4px;
    -fx-cursor:hand;
    -fx-font-size: 14px;
    -fx-font-family: Arial;
    -fx-text-fill: #fff;
}
.delete-btn:hover{
    -fx-background-color: CD1818;
}
.textfield{
    -fx-background-color: linear-gradient(to bottom, #efefef,
#eee);
    -fx-background-radius: 2px;
    -fx-font-family: Tahoma;
    -fx-border-color:#000;
    -fx-border-radius: 2px;
    -fx-border-width: .4px;
}
```

```
.textfield:focused{
    -fx-border-color:linear-gradient(to top right, #3c2c21,
#93773e);
    -fx-background-color: #fff;
    -fx-border-width: 1px;
}
.search{
    -fx-background-color: transparent;
    -fx-font-size: 13px;
    -fx-font-family: Arial;
    -fx-border-color: linear-gradient(to top right, #121522,
#146786);
    -fx-border-radius: 4px;
    -fx-border-width: .8px;
    -fx-padding: 0px 0px 0px 28px;
}
.search:focused{
    -fx-border-width: 1.5px;
}
.table-view{
    -fx-background-color:transparent;
    -fx-border-color: linear-gradient(to top right, #121522,
#146786);
    -fx-border-radius: 8px;
    -fx-border-width: 2px;
    -fx-padding: 0px;
}
.table-view .table-column{
    -fx-alignment: CENTER;
}
.table-view .column-header-background{
    -fx-background-color: linear-gradient(to top right,
#121522, #146786);
    -fx-background-radius: 8px 8px 0px 0px;
    -fx-background-insets: 0 0 0 0;
}
.table-view .column-header, .filter{
    -fx-background-color: transparent;
    -fx-size: 40px;
}
.table-view .column-header .label{
    -fx-text-fill: #fff;
    -fx-font-family: Arial;
}
.info{
    -fx-background-color:linear-gradient(to top right,
#12374e, #12374e);
    -fx-background-radius: 0 8px 8px 0;
}
.info-label{
```

```
    -fx-background-color: #fff;
    -fx-background-radius: 4px;
    -fx-padding: 0 0 0 5px;
}
```

17. Kemudian buatkan file baru dengan nama dashboardController.java, yang berfungsi sebagai kontrol agar halaman dashboard yang telah dibuat sebelumnya dapat muncul ketika di running dan terhubung dengan halaman login

    Source code dashboardController.java

```
package programmanajementokobuku;

import java.io.File;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.time.LocalDate;
import java.util.Date;
import java.util.Optional;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.collections.transformation.FilteredList;
import javafx.collections.transformation.SortedList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.chart.AreaChart;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.XYChart;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.Spinner;
import javafx.scene.control.SpinnerValueFactory;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
```

```
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.stage.FileChooser;
import javafx.stage.FileChooser.ExtensionFilter;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

public class dashboardController implements Initializable{

    @FXML
    private AnchorPane main_form;

    @FXML
    private Button close;

    @FXML
    private Button minimize;

    @FXML
    private Label username;

    @FXML
    private Button dashboard_btn;

    @FXML
    private Button availableBooks_btn;

    @FXML
    private Button purchase_btn;

    @FXML
    private Button logout;

    @FXML
    private AnchorPane dashboard_form;

    @FXML
    private Label dashboard_AB;

    @FXML
    private Label dashboard_TI;

    @FXML
    private Label dashboard_TC;

    @FXML
    private AreaChart<?, ?> dashboard_incomeChart;
```

```
    @FXML
    private BarChart<?, ?> dashboard_customerChart;

    @FXML
    private AnchorPane availableBooks_form;

    @FXML
    private ImageView availableBooks_imageView;

    @FXML
    private Button availableBooks_importBtn;

    @FXML
    private TextField availableBooks_bookID;

    @FXML
    private TextField availableBooks_bookTitle;

    @FXML
    private TextField availableBooks_author;

    @FXML
    private TextField availableBooks_genre;

    @FXML
    private DatePicker availableBooks_date;

    @FXML
    private TextField availableBooks_price;

    @FXML
    private Button availableBooks_addBtn;

    @FXML
    private Button availableBooks_updateBtn;

    @FXML
    private Button availableBooks_clearBtn;

    @FXML
    private Button availableBooks_deleteBtn;

    @FXML
    private TextField availableBooks_search;

    @FXML
    private TableView<DataBuku> availableBooks_tableView;

    @FXML
```

```
    private TableColumn<DataBuku, String>
availableBooks_col_bookID;

    @FXML
    private TableColumn<DataBuku, String>
availableBooks_col_bookTItle;

    @FXML
    private TableColumn<DataBuku, String>
availableBooks_col_author;

    @FXML
    private TableColumn<DataBuku, String>
availableBooks_col_genre;

    @FXML
    private TableColumn<DataBuku, String>
availableBooks_col_date;

    @FXML
    private TableColumn<DataBuku, String>
availableBooks_col_price;

    @FXML
    private AnchorPane purchase_form;

    @FXML
    private ComboBox<?> purchase_bookID;

    @FXML
    private ComboBox<?> purchase_bookTitle;

    @FXML
    private Label purchase_total;

    @FXML
    private Button purchase_addBtn;

    @FXML
    private Label purchase_info_bookID;

    @FXML
    private Label purchase_info_bookTItle;

    @FXML
    private Label purchase_info_author;

    @FXML
    private Label purchase_info_genre;
```

```
    @FXML
    private Label purchase_info_date;

    @FXML
    private Button purchase_payBtn;

    @FXML
    private TableView<DataCustomer> purchase_tableView;

    @FXML
    private Spinner<Integer> purchase_quantity;

    @FXML
    private TableColumn<DataCustomer, String>
purchase_col_bookID;

    @FXML
    private TableColumn<DataCustomer, String>
purchase_col_bookTitle;

    @FXML
    private TableColumn<DataCustomer, String>
purchase_col_author;

    @FXML
    private TableColumn<DataCustomer, String>
purchase_col_genre;

    @FXML
    private TableColumn<DataCustomer, String>
purchase_col_quantity;

    @FXML
    private TableColumn<DataCustomer, String>
purchase_col_price;

    private Connection connect;
    private PreparedStatement prepare;
    private Statement statement;
    private ResultSet hasil;

    private Image gambar;

    public void dashboard_BukuTersedia(){

        String sql = "SELECT COUNT(id) FROM book";

        connect = database.connectDb();
        int countAB = 0;
        try{
```

```
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            if(hasil.next()){
                countAB = hasil.getInt("COUNT(id)");
            }

            dashboard_AB.setText(String.valueOf(countAB));

        }catch(Exception e){e.printStackTrace();}
    }

    public void dashboard_TotalPendapatan(){

        String sql = "SELECT SUM(total) FROM customer_info";

        connect = database.connectDb();
        double sumTotal = 0;
        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            if(hasil.next()){
                sumTotal = hasil.getDouble("SUM(total)");
            }

            dashboard_TI.setText("Rp" +
String.valueOf(sumTotal));

        }catch(Exception e){e.printStackTrace();}
    }

    public void dashboard_TotalCustomer(){
        String sql = "SELECT COUNT(id) FROM customer_info";

        connect = database.connectDb();
        int countTC = 0;
        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            if(hasil.next()){
                countTC = hasil.getInt("COUNT(id)");
            }

            dashboard_TC.setText(String.valueOf(countTC));

        }catch(Exception e){e.printStackTrace();}

    }
```

```
    public void grafikPenghasilan(){

        dashboard_incomeChart.getData().clear();

        String sql = "SELECT date, SUM(total) FROM
customer_info GROUP BY date ORDER BY TIMESTAMP(date) ASC
LIMIT 6";

        connect = database.connectDb();

        try{
            XYChart.Series chart = new XYChart.Series();

            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            while(hasil.next()){
                chart.getData().add(new
XYChart.Data(hasil.getString(1), hasil.getInt(2)));
            }

            dashboard_incomeChart.getData().add(chart);

        }catch(Exception e){e.printStackTrace();}

    }


    public void grafikPelanggan(){

        dashboard_customerChart.getData().clear();

        String sql = "SELECT date, COUNT(id) FROM
customer_info GROUP BY date ORDER BY TIMESTAMP(date) ASC
LIMIT 4";

        connect = database.connectDb();

        try{
            XYChart.Series chart = new XYChart.Series();

            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            while(hasil.next()){
                chart.getData().add(new
XYChart.Data(hasil.getString(1), hasil.getInt(2)));
            }
```

```
            dashboard_customerChart.getData().add(chart);

        }catch(Exception e){e.printStackTrace();}

    }

    public void tambahBuku(){

        String sql = "INSERT INTO book (book_id, title,
author, genre, pub_date, price, image) "
                + "VALUES(?,?,?,?,?,?,?)";

        connect = database.connectDb();

        try{
            Alert alert;

            if(availableBooks_bookID.getText().isEmpty()
                    ||
availableBooks_bookTitle.getText().isEmpty()
                    ||
availableBooks_author.getText().isEmpty()
                    ||
availableBooks_genre.getText().isEmpty()
                    || availableBooks_date.getValue() ==
null
                    ||
availableBooks_price.getText().isEmpty()
                    || getData.path == null || getData.path
== ""){
                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error ");
                alert.setHeaderText(null);
                alert.setContentText("Lengkapi Data Yang
Kosong");
                alert.showAndWait();
            }else{
                // CHECK IF BOOK ID IS ALREADY EXIST
                String checkData = "SELECT book_id FROM book
WHERE book_id = '"
                        +availableBooks_bookID.getText()+"'"
;

                statement = connect.createStatement();
                hasil = statement.executeQuery(checkData);

                if(hasil.next()){
                    alert = new Alert(AlertType.ERROR);
                    alert.setTitle("Error ");
                    alert.setHeaderText(null);
```

```
                    alert.setContentText("ID Buku: " +
availableBooks_bookID.getText() + " Sudah Tersedia!");
                    alert.showAndWait();
                }else{

                    prepare = connect.prepareStatement(sql);
                    prepare.setString(1,
availableBooks_bookID.getText());
                    prepare.setString(2,
availableBooks_bookTitle.getText());
                    prepare.setString(3,
availableBooks_author.getText());
                    prepare.setString(4,
availableBooks_genre.getText());
                    prepare.setString(5,
String.valueOf(availableBooks_date.getValue()));
                    prepare.setString(6,
availableBooks_price.getText());

                    String uri = getData.path;
                    uri = uri.replace("\\", "\\\\");

                    prepare.setString(7, uri);

                    prepare.executeUpdate();

                    alert = new
Alert(AlertType.INFORMATION);
                    alert.setTitle("Pesan Informasi");
                    alert.setHeaderText(null);
                    alert.setContentText("Buku Berhasil
Ditambahkan!");
                    alert.showAndWait();

                    // TO BE UPDATED THE TABLEVIEW
                    tampilDataBuku();
                    // CLEAR FIELDS
                    bersihkan();
                }
            }
        }catch(Exception e){e.printStackTrace();}

    }

    public void updateBuku(){

        String uri = getData.path;
        uri = uri.replace("\\", "\\\\");

        String sql = "UPDATE book SET title = '"
```

```
                +availableBooks_bookTitle.getText()+"',
author = '"
                +availableBooks_author.getText()+"', genre =
'"
                +availableBooks_genre.getText()+"', pub_date
= '"
                +availableBooks_date.getValue()+"', price =
'"
                +availableBooks_price.getText()+"', image =
'"
                +uri+"' WHERE book_id =
'"+availableBooks_bookID.getText()+"'";

        connect = database.connectDb();

        try{
            Alert alert;

            if(availableBooks_bookID.getText().isEmpty()
                    ||
availableBooks_bookTitle.getText().isEmpty()
                    ||
availableBooks_author.getText().isEmpty()
                    ||
availableBooks_genre.getText().isEmpty()
                    || availableBooks_date.getValue() ==
null
                    ||
availableBooks_price.getText().isEmpty()
                    || getData.path == null || getData.path
== ""){
                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error ");
                alert.setHeaderText(null);
                alert.setContentText("Lengkapi Data Yang
Kosong");
                alert.showAndWait();
            }else{
                alert = new Alert(AlertType.CONFIRMATION);
                alert.setTitle("Pesan Konfirmasi");
                alert.setHeaderText(null);
                alert.setContentText("Anda Yakin Ingin
Memperbarui ID Buku: " + availableBooks_bookID.getText() +
"?");
                Optional<ButtonType> option =
alert.showAndWait();

                if(option.get().equals(ButtonType.OK)){
                    statement = connect.createStatement();
                    statement.executeUpdate(sql);
```

```
                    alert = new
Alert(AlertType.INFORMATION);
                    alert.setTitle("Pesan Informasi");
                    alert.setHeaderText(null);
                    alert.setContentText("Data Berhasil
Diperbarui!");
                    alert.showAndWait();

                    // TO BE UPDATED THE TABLEVIEW
                    tampilDataBuku();
                    // CLEAR FIELDS
                    bersihkan();
                }
            }
        }catch(Exception e){e.printStackTrace();}

    }

    public void hapusBuku(){

        String sql = "DELETE FROM book WHERE book_id = '"
                +availableBooks_bookID.getText()+"'";

        connect = database.connectDb();

        try{
            Alert alert;

            if(availableBooks_bookID.getText().isEmpty()
                    ||
availableBooks_bookTitle.getText().isEmpty()
                    ||
availableBooks_author.getText().isEmpty()
                    ||
availableBooks_genre.getText().isEmpty()
                    || availableBooks_date.getValue() ==
null
                    ||
availableBooks_price.getText().isEmpty()
                    || getData.path == null || getData.path
== ""){
                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error ");
                alert.setHeaderText(null);
                alert.setContentText("Lengkapi Data Yang
Kosong");
                alert.showAndWait();
            }else{
                alert = new Alert(AlertType.CONFIRMATION);
```

```
                alert.setTitle("Pesan Konfirmasi");
                alert.setHeaderText(null);
                alert.setContentText("Anda Yakin Ingin
Menghapus Buku: " + availableBooks_bookID.getText() + "?");
                Optional<ButtonType> option =
alert.showAndWait();

                if(option.get().equals(ButtonType.OK)){
                    statement = connect.createStatement();
                    statement.executeUpdate(sql);

                    alert = new
Alert(AlertType.INFORMATION);
                    alert.setTitle("Pesan Informasi");
                    alert.setHeaderText(null);
                    alert.setContentText("Buku Berhasil
Dihapus!");
                    alert.showAndWait();

                    // TO BE UPDATED THE TABLEVIEW
                    tampilDataBuku();
                    // CLEAR FIELDS
                    bersihkan();
                }
            }
        }catch(Exception e){e.printStackTrace();}

    }

    public void bersihkan(){
        availableBooks_bookID.setText("");
        availableBooks_bookTitle.setText("");
        availableBooks_author.setText("");
        availableBooks_genre.setText("");
        availableBooks_date.setValue(null);
        availableBooks_price.setText("");

        getData.path = "";

        availableBooks_imageView.setImage(null);
    }

    public void tambahGambar(){

        FileChooser open = new FileChooser();
        open.setTitle("Buka File Gambar");
        open.getExtensionFilters().add(new
ExtensionFilter("File Gambar", "*jpeg", "*png"));
```

```
        File file =
open.showOpenDialog(main_form.getScene().getWindow());

        if(file != null){
            getData.path = file.getAbsolutePath();

            gambar = new Image(file.toURI().toString(), 112,
137, false, true);
            availableBooks_imageView.setImage(gambar);
        }

    }

    public ObservableList<DataBuku> dataBuku(){

        ObservableList<DataBuku> listData =
FXCollections.observableArrayList();
        String sql = "SELECT * FROM book";

        connect = database.connectDb();

        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            DataBuku bookD;

            while(hasil.next()){
                bookD = new
DataBuku(hasil.getInt("book_id"), hasil.getString("title")
                        , hasil.getString("author"),
hasil.getString("genre")
                        , hasil.getDate("pub_date"),
hasil.getDouble("price")
                        , hasil.getString("image"));

                listData.add(bookD);
            }
        }catch(Exception e){e.printStackTrace();}
        return listData;
    }

    private ObservableList<DataBuku> availableBooksList;
    public void tampilDataBuku(){
        availableBooksList = dataBuku();

        availableBooks_col_bookID.setCellValueFactory(new
PropertyValueFactory<>("bookId"));
        availableBooks_col_bookTItle.setCellValueFactory(new
PropertyValueFactory<>("title"));
```

```
        availableBooks_col_author.setCellValueFactory(new
PropertyValueFactory<>("author"));
        availableBooks_col_genre.setCellValueFactory(new
PropertyValueFactory<>("genre"));
        availableBooks_col_date.setCellValueFactory(new
PropertyValueFactory<>("date"));
        availableBooks_col_price.setCellValueFactory(new
PropertyValueFactory<>("price"));

        availableBooks_tableView.setItems(availableBooksList
);
    }

    public void pilihBuku(){
        DataBuku bookD =
availableBooks_tableView.getSelectionModel().getSelectedItem
();
        int num =
availableBooks_tableView.getSelectionModel().getSelectedInde
x();

        if((num - 1) < -1){ return; }

        availableBooks_bookID.setText(String.valueOf(bookD.g
etBookId()));
        availableBooks_bookTitle.setText(bookD.getTitle());
        availableBooks_author.setText(bookD.getAuthor());
        availableBooks_genre.setText(bookD.getGenre());
        availableBooks_date.setValue(LocalDate.parse(String.
valueOf(bookD.getDate())));
        availableBooks_price.setText(String.valueOf(bookD.ge
tPrice()));

        getData.path = bookD.getImage();

        String uri = "file:" + bookD.getImage();

        gambar = new Image(uri, 112, 137, false, true);

        availableBooks_imageView.setImage(gambar);
    }

    public void cariBuku(){

        FilteredList<DataBuku> filter = new
FilteredList<>(availableBooksList, e -> true);

        availableBooks_search.textProperty().addListener((Ob
servable, oldValue, newValue) ->{
```

```
            filter.setPredicate(predicateBookData -> {

                if(newValue == null || newValue.isEmpty()){
                    return true;
                }

                String searchKey = newValue.toLowerCase();

                if(predicateBookData.getBookId().toString().
contains(searchKey)){
                    return true;
                }else
if(predicateBookData.getTitle().toLowerCase().contains(searc
hKey)){
                    return true;
                }else
if(predicateBookData.getAuthor().toLowerCase().contains(sear
chKey)){
                    return true;
                }else
if(predicateBookData.getGenre().toLowerCase().contains(searc
hKey)){
                    return true;
                }else
if(predicateBookData.getDate().toString().contains(searchKey
)){
                    return true;
                }else
if(predicateBookData.getPrice().toString().contains(searchKe
y)){
                    return true;
                }else return false;
            });
        });

        SortedList<DataBuku> sortList = new
SortedList(filter);
        sortList.comparatorProperty().bind(availableBooks_ta
bleView.comparatorProperty());
        availableBooks_tableView.setItems(sortList);

    }

    private double totalP;
    public void tambahPembayaran(){
        IDPembelianKustomer();

        String sql = "INSERT INTO customer (customer_id,
book_id, title, author, genre, quantity, price, date) "
                + "VALUES(?,?,?,?,?,?,?,?)";
```

```
        connect = database.connectDb();

        try{
            Alert alert;

            if(purchase_bookTitle.getSelectionModel().getSel
ectedItem() == null
                    ||
purchase_bookID.getSelectionModel().getSelectedItem() ==
null){
                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error ");
                alert.setHeaderText(null);
                alert.setContentText("Pilih Buku Terlebih
Dahulu");
                alert.showAndWait();
            }else{

                prepare = connect.prepareStatement(sql);
                prepare.setString(1,
String.valueOf(customerId));
                prepare.setString(2,
purchase_info_bookID.getText());
                prepare.setString(3,
purchase_info_bookTItle.getText());
                prepare.setString(4,
purchase_info_author.getText());
                prepare.setString(5,
purchase_info_genre.getText());
                prepare.setString(6, String.valueOf(qty));

                String checkData = "SELECT title, price FROM
book WHERE title = '"
                        +purchase_bookTitle.getSelectionMode
l().getSelectedItem()+"'";

                double priceD = 0;

                statement = connect.createStatement();
                hasil = statement.executeQuery(checkData);

                if(hasil.next()){
                    priceD = hasil.getDouble("price");
                }

                totalP = (qty * priceD);

                prepare.setString(7,
String.valueOf(totalP));
```

```
                Date date = new Date();
                java.sql.Date sqlDate = new
java.sql.Date(date.getTime());

                prepare.setString(8,
String.valueOf(sqlDate));

                prepare.executeUpdate();

                tampilTotalBayar();
                tampilDataKustomer();
            }
        }catch(Exception e){e.printStackTrace();}
    }

    public void pembelian(){

        String sql = "INSERT INTO customer_info
(customer_id, total, date) "
                + "VALUES(?,?,?)";

        connect = database.connectDb();

        try{
            Alert alert;
            if(displayTotal == 0){
                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error ");
                alert.setHeaderText(null);
                alert.setContentText("Tidak Valid");
                alert.showAndWait();
            }else{
                alert = new Alert(AlertType.CONFIRMATION);
                alert.setTitle("Pesan Konfirmasi");
                alert.setHeaderText(null);
                alert.setContentText("Anda Yakin?");
                Optional<ButtonType> option =
alert.showAndWait();

                if(option.get().equals(ButtonType.OK)){
                    prepare = connect.prepareStatement(sql);
                    prepare.setString(1,
String.valueOf(customerId));
                    prepare.setString(2,
String.valueOf(displayTotal));

                    Date date = new Date();
                    java.sql.Date sqlDate = new
java.sql.Date(date.getTime());
```

```
                    prepare.setString(3,
String.valueOf(sqlDate));

                    prepare.executeUpdate();

                    alert = new
Alert(AlertType.INFORMATION);
                    alert.setTitle("Pesan Informasi");
                    alert.setHeaderText(null);
                    alert.setContentText("Pembayaran
Berhasil!");
                    alert.showAndWait();
                }
            }
        }catch(Exception e){e.printStackTrace();}

    }

    private double displayTotal;
    public void tampilTotalBayar(){
        IDPembelianKustomer();

        String sql = "SELECT SUM(price) FROM customer WHERE
customer_id = '"+customerId+"'";

        connect = database.connectDb();

        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            if(hasil.next()){
                displayTotal =
hasil.getDouble("SUM(price)");
            }

            purchase_total.setText("Rp " +
String.format("%.0f", displayTotal));

        }catch(Exception e){e.printStackTrace();}

    }

    public void pembayaranBuku(){

        String sql = "SELECT book_id FROM book";

        connect = database.connectDb();
```

```
        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            ObservableList listData =
FXCollections.observableArrayList();

            while(hasil.next()){
                listData.add(hasil.getString("book_id"));
            }

            purchase_bookID.setItems(listData);
            judulBuku();
        }catch(Exception e){e.printStackTrace();}

    }

    public void judulBuku(){

        String sql = "SELECT book_id, title FROM book WHERE
book_id = '"
                +purchase_bookID.getSelectionModel().getSele
ctedItem()+"'";

        connect = database.connectDb();

        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            ObservableList listData =
FXCollections.observableArrayList();

            while(hasil.next()){
                listData.add(hasil.getString("title"));
            }

            purchase_bookTitle.setItems(listData);

            infoBayarBuku();

        }catch(Exception e){e.printStackTrace();}

    }

    public void infoBayarBuku(){

        String sql = "SELECT * FROM book WHERE title = '"
                +purchase_bookTitle.getSelectionModel().getS
electedItem()+"'";
```

```
        connect = database.connectDb();

        String bookId = "";
        String title = "";
        String author = "";
        String genre = "";
        String date = "";

        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            if(hasil.next()){
                bookId = hasil.getString("book_id");
                title = hasil.getString("title");
                author = hasil.getString("author");
                genre = hasil.getString("genre");
                date = hasil.getString("pub_date");
            }

            purchase_info_bookID.setText(bookId);
            purchase_info_bookTItle.setText(title);
            purchase_info_author.setText(author);
            purchase_info_genre.setText(genre);
            purchase_info_date.setText(date);


        }catch(Exception e){e.printStackTrace();}

    }

    public ObservableList<DataCustomer> dataPembelian(){
        IDPembelianKustomer();
        String sql = "SELECT * FROM customer WHERE
customer_id = '"+customerId+"'";

        ObservableList<DataCustomer> listData =
FXCollections.observableArrayList();

        connect = database.connectDb();

        try{
            prepare  = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            DataCustomer customerD;

            while(hasil.next()){
```

```
                customerD = new
DataCustomer(hasil.getInt("customer_id")
                        , hasil.getInt("book_id")
                        , hasil.getString("title")
                        , hasil.getString("author")
                        , hasil.getString("genre")
                        , hasil.getInt("quantity")
                        , hasil.getDouble("price")
                        , hasil.getDate("date"));

                listData.add(customerD);
            }

        }catch(Exception e){e.printStackTrace();}
        return listData;
    }

    private ObservableList<DataCustomer>
purchaseCustomerList;
    public void tampilDataKustomer(){
        purchaseCustomerList = dataPembelian();

        purchase_col_bookID.setCellValueFactory(new
PropertyValueFactory<>("bookId"));
        purchase_col_bookTitle.setCellValueFactory(new
PropertyValueFactory<>("title"));
        purchase_col_author.setCellValueFactory(new
PropertyValueFactory<>("author"));
        purchase_col_genre.setCellValueFactory(new
PropertyValueFactory<>("genre"));
        purchase_col_quantity.setCellValueFactory(new
PropertyValueFactory<>("quantity"));
        purchase_col_price.setCellValueFactory(new
PropertyValueFactory<>("price"));

        purchase_tableView.setItems(purchaseCustomerList);

    }

    private SpinnerValueFactory<Integer> spinner;

    public void tampilPembelianQTY(){
        spinner = new
SpinnerValueFactory.IntegerSpinnerValueFactory(0, 10, 0);
        purchase_quantity.setValueFactory(spinner);
    }
    private int qty;
    public void pembelianQTY(){
        qty = purchase_quantity.getValue();
    }
```

```
    private int customerId;
    public void IDPembelianKustomer(){

        String sql = "SELECT MAX(customer_id) FROM
customer";
        int checkCID = 0 ;
        connect = database.connectDb();

        try{
            prepare = connect.prepareStatement(sql);
            hasil = prepare.executeQuery();

            if(hasil.next()){
                customerId =
hasil.getInt("MAX(customer_id)");
            }

            String checkData = "SELECT MAX(customer_id) FROM
customer_info";

            prepare = connect.prepareStatement(checkData);
            hasil = prepare.executeQuery();

            if(hasil.next()){
                checkCID = hasil.getInt("MAX(customer_id)");
            }

            if(customerId == 0){
                customerId += 1;
            }else if(checkCID == customerId){
                customerId = checkCID + 1;
            }

        }catch(Exception e){e.printStackTrace();}

    }

    public void tampilUsername(){
        String user = getData.username;
        user = user.substring(0, 1).toUpperCase() +
user.substring(1);
        username.setText(user);
    }

    public void peralihan(ActionEvent event){

        if(event.getSource() == dashboard_btn){
            dashboard_form.setVisible(true);
            availableBooks_form.setVisible(false);
```

```
            purchase_form.setVisible(false);

            dashboard_btn.setStyle("-fx-background-
color:linear-gradient(to top right, #12415b, #121807);");
            availableBooks_btn.setStyle("-fx-background-
color: transparent");
            purchase_btn.setStyle("-fx-background-color:
transparent");

            dashboard_BukuTersedia();
            dashboard_TotalPendapatan();
            dashboard_TotalCustomer();
            grafikPenghasilan();
            grafikPelanggan();

        }else if(event.getSource() == availableBooks_btn){
            dashboard_form.setVisible(false);
            availableBooks_form.setVisible(true);
            purchase_form.setVisible(false);

            availableBooks_btn.setStyle("-fx-background-
color:linear-gradient(to top right, #12415b, #121807);");
            dashboard_btn.setStyle("-fx-background-color:
transparent");
            purchase_btn.setStyle("-fx-background-color:
transparent");

            tampilDataBuku();
            cariBuku();

        }else if(event.getSource() == purchase_btn){
            dashboard_form.setVisible(false);
            availableBooks_form.setVisible(false);
            purchase_form.setVisible(true);

            purchase_btn.setStyle("-fx-background-
color:linear-gradient(to top right, #12415b, #121807);");
            availableBooks_btn.setStyle("-fx-background-
color: transparent");
            dashboard_btn.setStyle("-fx-background-color:
transparent");

            judulBuku();
            pembayaranBuku();
            tampilDataKustomer();
            tampilPembelianQTY();
            tampilTotalBayar();

        }
    }
```

```
    private double x = 0;
    private double y = 0;
    public void keluar(){
        try{
            Alert alert = new Alert(AlertType.CONFIRMATION);
            alert.setTitle("Pesan Konfirmasi");
            alert.setHeaderText(null);
            alert.setContentText("Anda Yakin Ingin
Keluar?");
            Optional<ButtonType> option =
alert.showAndWait();

            if(option.get().equals(ButtonType.OK)){

                // HIDE YOUR DASHBOARD
                logout.getScene().getWindow().hide();
                // LINK YOUR LOGIN FORM
                Parent root =
FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"))
;
                Stage stage = new Stage();
                Scene scene = new Scene(root);

                root.setOnMousePressed((MouseEvent event) -
>{
                    x = event.getSceneX();
                    y = event.getSceneY();
                });

                root.setOnMouseDragged((MouseEvent event) -
>{
                    stage.setX(event.getScreenX() - x);
                    stage.setY(event.getScreenY() - y);

                    stage.setOpacity(.8);
                });

                root.setOnMouseReleased((MouseEvent event) -
>{
                    stage.setOpacity(1);
                });

                stage.initStyle(StageStyle.TRANSPARENT);

                stage.setScene(scene);
                stage.show();
            }

        }catch(Exception e){e.printStackTrace();}
```

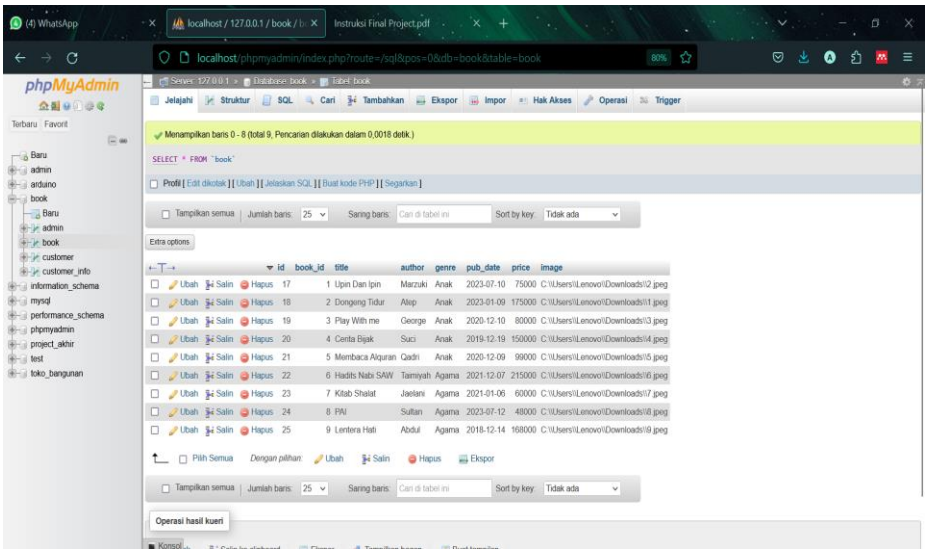
```
    }

    public void close(){
        System.exit(0);
    }

    public void minimize(){
        Stage stage =
(Stage)main_form.getScene().getWindow();
        stage.setIconified(true);
    }

    @Override
    public void initialize(URL location, ResourceBundle
resources) {
        tampilUsername();

        dashboard_BukuTersedia();
        dashboard_TotalPendapatan();
        dashboard_TotalCustomer();
        grafikPenghasilan();
        grafikPelanggan();

        // TO SHOW THE DATA ON TABLEVIEW (AVAILABLE BOOKS)
        tampilDataBuku();

        pembayaranBuku();
        judulBuku();
        tampilDataKustomer();
        tampilPembelianQTY();
        tampilTotalBayar();

    }

}
```

18. Kemudian buatlah file baru dengan nama getData.java. yang berfungsi untuk mengambil data buku yang telah diinputkan

    Source code getData.java

```
package programmanajementokobuku;

public class getData {
    public static String username;
    public static String path;
}
```

19. Kemudian buat tabel baru pada database yang telah dibuat sebelunya dengan nama “book”



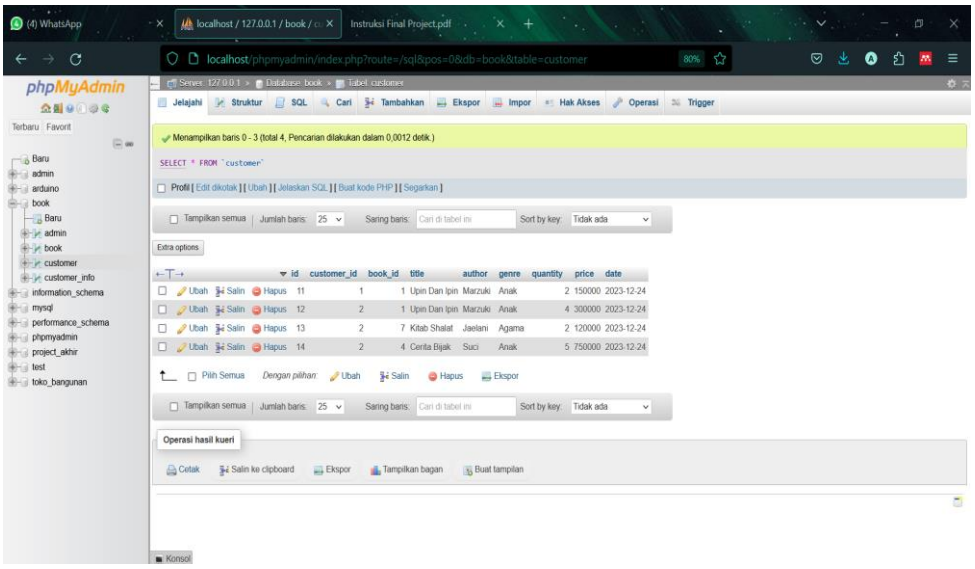
20. Selanjutnya buatlah file baru dengan nama DataBuku.java untuk menyimpan data pada tabel “book” di database yang telah dibuat sebelumnya

Source code DataBuku.java

```
package programmanajementokobuku;

import java.sql.Date;

public class DataBuku {
    private Integer bookId;
    private String title;
    private String author;
    private String genre;
    private Date date;
    private Double price;
    private String image;
    // MAKE SURE YOU FOLLOWED THE PARAMETERS THAT I PUT
    public DataBuku(Integer bookId, String title, String
author, String genre, Date date, Double price, String
image){
        this.bookId = bookId;
        this.title = title;
        this.author = author;
        this.genre = genre;
        this.date = date;
        this.price = price;
        this.image = image;
    }
    public Integer getBookId(){
        return bookId;
    }
    public String getTitle(){
```

```
        return title;
    }
    public String getAuthor(){
        return author;
    }
    public String getGenre(){
        return genre;
    }
    public Date getDate(){
        return date;
    }
    public Double getPrice(){
        return price;
    }
    public String getImage(){
        return image;
    }
}
```

21. Kemudian buat tabel baru pada database yang telah dibuat sebelunya dengan nama “customer”



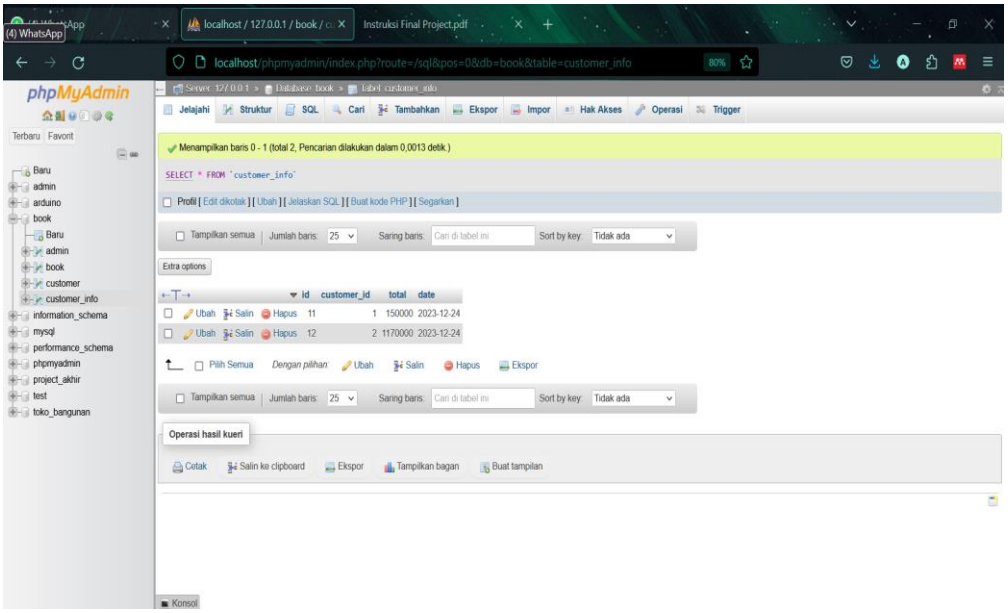
22. Selanjutnya buatlah file baru dengan nama DataCustomer.java untuk menyimpan data customer pada tabel “customer” di database yang telah dibuat sebelumnya

Source code DataCustomer.java

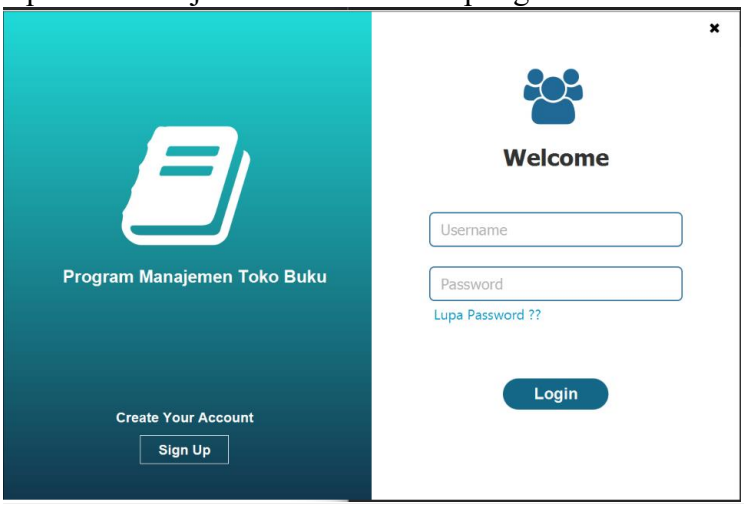
```
package programmanajementokobuku;
import java.sql.Date;

public class DataCustomer {
    private Integer customerId;
    private Integer bookId;
    private String title;
```

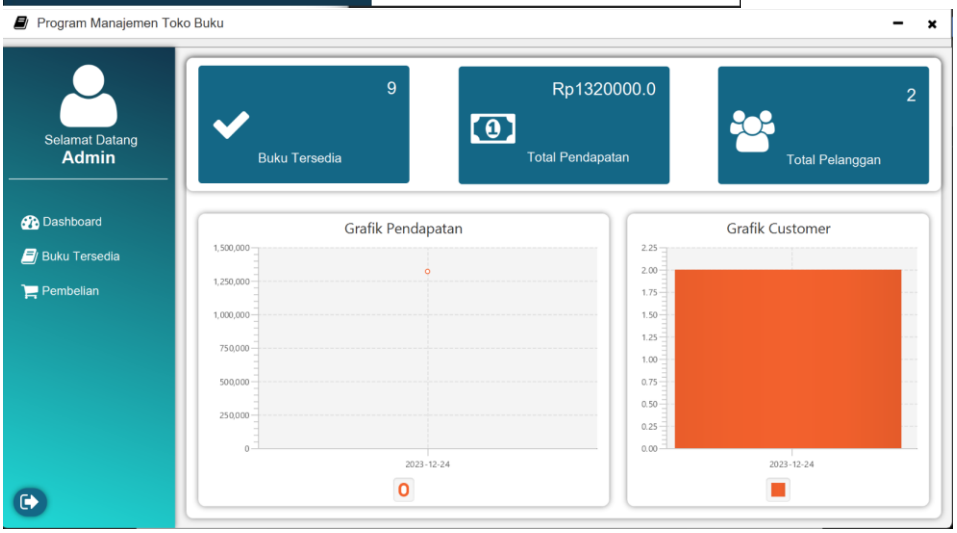
```
    private String author;
    private String genre;
    private Integer quantity;
    private Double price;
    private Date date;

    public DataCustomer(Integer customerId, Integer bookId,
String title, String author
            , String genre, Integer quantity, Double price,
Date date){
        this.customerId = customerId;
        this.bookId = bookId;
        this.title = title;
        this.author = author;
        this.genre = genre;
        this.quantity = quantity;
        this.price = price;
        this.date = date;
    }
    public Integer getCustomerId(){
        return customerId;
    }
    public Integer getBookId(){
        return bookId;
    }
    public String getTitle(){
        return title;
    }
    public String getAuthor(){
        return author;
    }
    public String getGenre(){
        return genre;
    }
    public Integer getQuantity(){
        return quantity;
    }
    public Double getPrice(){
        return price;
    }
    public Date getDate(){
        return date;
    }
}
```

23. Kemudian buat tabel baru pada database yang telah dibuat sebelumnya dengan nama “customer_info”. Untuk menyimpan info pembelian dan pembayaran dari customer



24. Setelah mengikuti langkah-langkah seperti diatas dengan baik dan benar, Aplikasi Manajemen Toko Buku siap digunakan.





E. Referensi

[1] F. Maulana Syahputra *et al.*, “RANCANGAN APLIKASI PENJUALAN PADA TOKO BUKU SYAHPUTRA BERBASIS DESKTOP,” 2022.

[2] A. Fajri Ali, “RANCANG BANGUN APLIKASI PENJUALAN BARANG BERBASIS JAVA PROGRAMMING,” *Jurnal SIMTIKA*, vol. 2, no. 1, 2019.