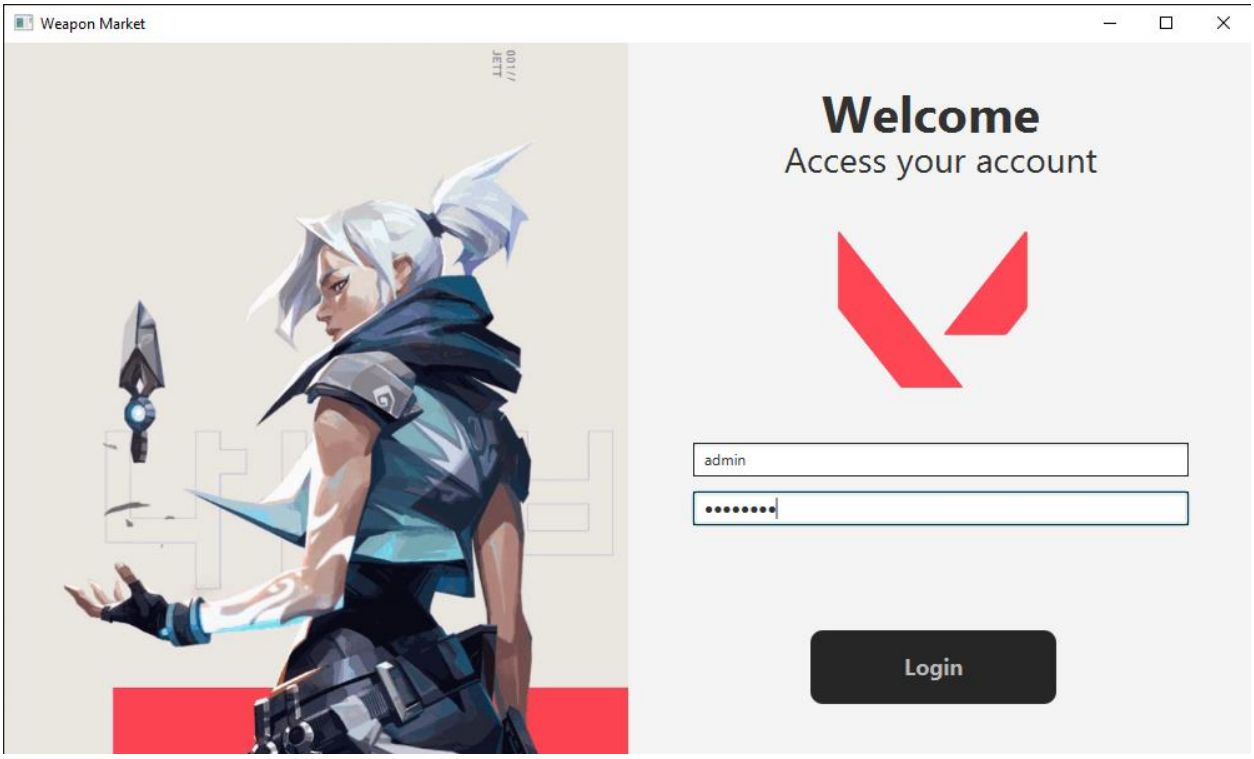
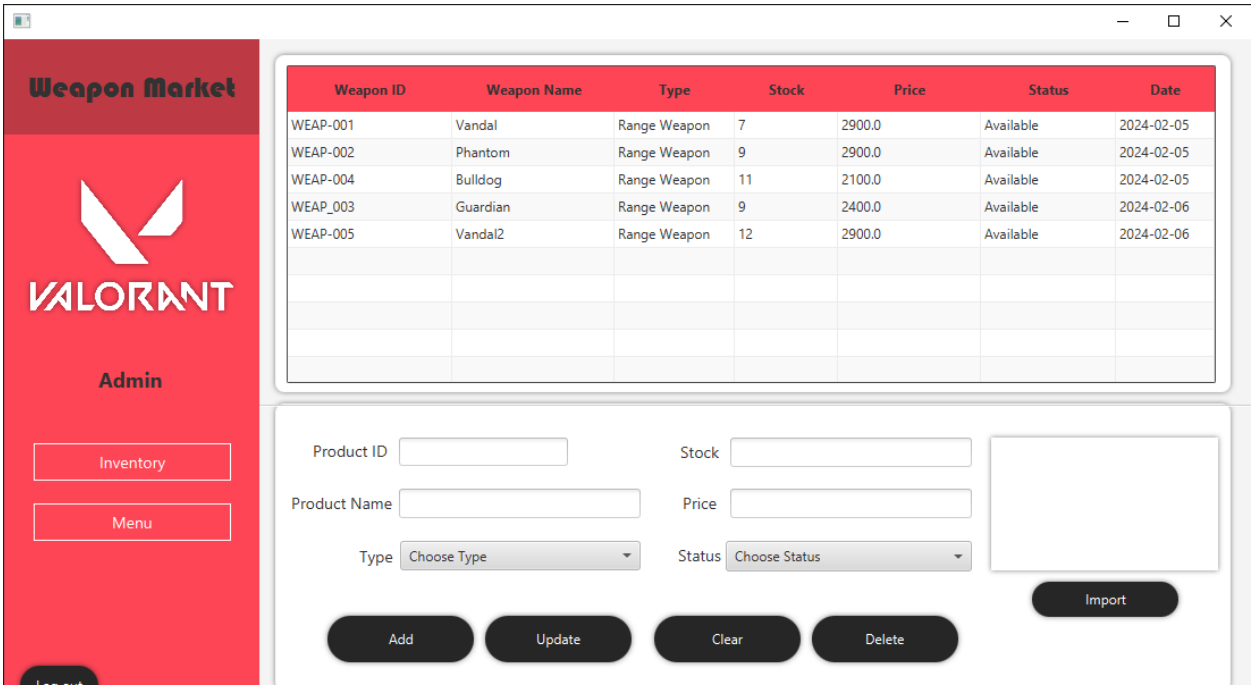
MUHAMAD DAFFA NAHRULLAH
301230015

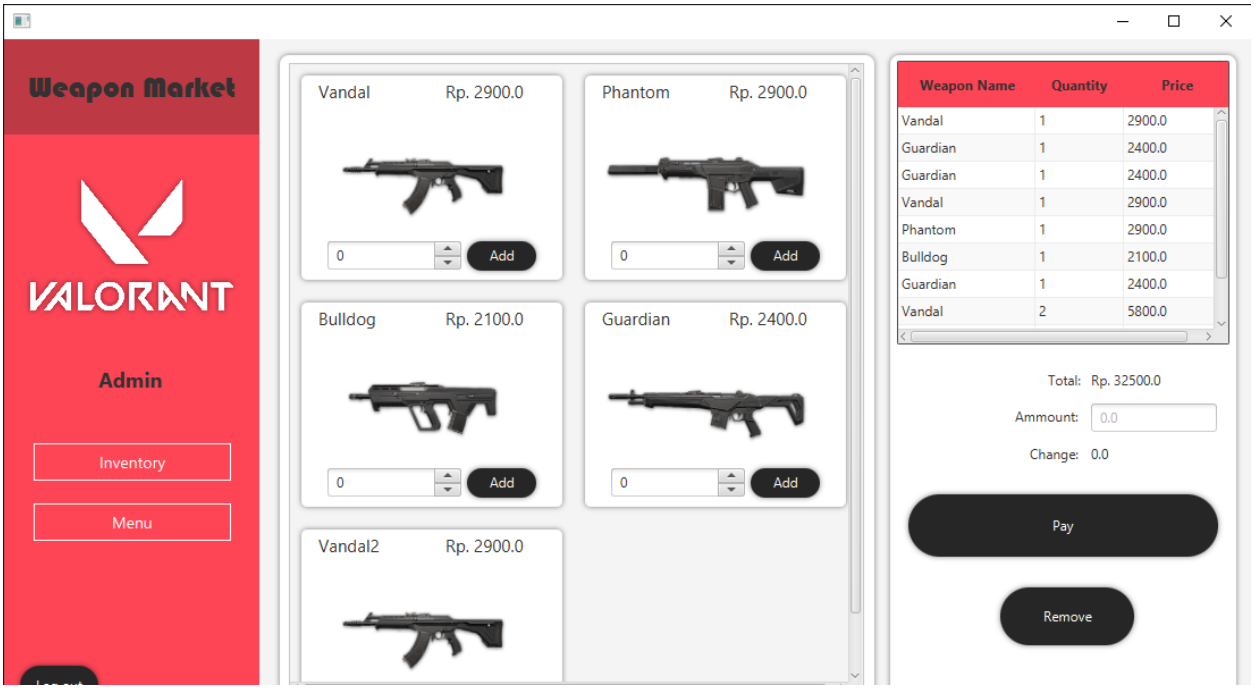
IF 1A

Valorant Weapon Market

Aplikasi ini memungkinkan kita untuk membeli senjata dari game Valorant




Weapon Market
VALORANT
Admin
Inventory
Menu
Weapon ID
Weapon Name
Type
Stock
Price
Status
Date
WEAP-001 Vandal Range Weapon 7 2900.0 Available 2024-02-05
WEAP-002 Phantom Range Weapon 9 2900.0 Available 2024-02-05
WEAP-004 Bulldog Range Weapon 11 2100.0 Available 2024-02-05
WEAP_003 Guardian Range Weapon 9 2400.0 Available 2024-02-06
WEAP-005 Vandal2 Range Weapon 12 2900.0 Available 2024-02-06
Product ID
Stock
Product Name
Price
Type
Choose Type
Status
Choose Status
Import
Add
Update
Clear
Delete



Main.java

```
package org.valo.system;

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

```
public class Main extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("/org/valo/view/loginView.fxml"));

        Scene scene = new Scene(root);

        stage.setTitle("Weapon Market");

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

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

LoginViewController.java

```
package org.valo.controller;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Pos;
import javafx.scene.Node;
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.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import javafx.util.Duration;
import org.valo.database.Database;


public class LoginViewController implements Initializable{
    @FXML
   private ImageView imgLogo;

   @FXML
   private Button loginbtn;

   @FXML
   private PasswordField txtPassword;

   @FXML
   private TextField txtUsername;
```

```
private Connection connect;
private PreparedStatement prepare;
private ResultSet result;
private Alert alert;
private String username;



@Override
public void initialize(URL location, ResourceBundle resources) {
    Image img = new Image("/org/valo/images/jett.gif");
    imgLogo.setImage(img);
}
 @FXML
private void login(ActionEvent event) {
    if (txtUsername.getText().isEmpty() || txtPassword.getText().isEmpty()) {
        alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error Message");
        alert.setHeaderText(null);
        alert.setContentText("Incorrect Username/Password");
        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, txtUsername.getText());
            prepare.setString(2, txtPassword.getText());

            result = prepare.executeQuery();

            if (result.next()) {

                data.username = txtUsername.getText();

                username = txtUsername.getText();

                alert = new Alert(AlertType.INFORMATION);
                alert.setTitle("Information Message");
                alert.setHeaderText(null);
                alert.setContentText("Successfully Login!");
```

```
        alert.showAndWait();

        Parent root = FXMLLoader.load(getClass().getResource("/org/valo/view/MainView.fxml"));

        Stage stage = new Stage();
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();

        loginbtn.getScene().getWindow().hide();

    } else {
        alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error Message");
        alert.setHeaderText(null);
        alert.setContentText("Incorrect Username/Password");
        alert.showAndWait();
    }

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

```
        }

    }




}
```

MainViewController.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package org.valo.controller;

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.util.ArrayList;
```

```
import java.util.Date;
import java.util.List;
import java.util.Optional;
import java.util.ResourceBundle;
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.geometry.Insets;
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.ButtonType;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
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.layout.AnchorPane;
import javafx.scene.layout.GridPane;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import static org.valo.controller.data.cID;
import org.valo.database.Database;

/**
 *
 * @author USER
 */
public class MainViewController implements Initializable {

    @FXML
    private Button chart_btn;

    @FXML
    private Button inventory_addBtn;

    @FXML
```

```
private Button inventory_btn;

@FXML
private Button inventory_clearBtn;

@FXML
private TableView<productData> inventory_tableView;

@FXML
private TableColumn<productData, String> inventory_col_productID;

@FXML
private TableColumn<productData, String> inventory_col_productName;

@FXML
private TableColumn<productData, String> inventory_col_type;

@FXML
private TableColumn<productData, String> inventory_col_stock;

@FXML
private TableColumn<productData, String> inventory_col_price;
```

```
@FXML
private TableColumn<productData, String> inventory_col_status;

@FXML
private TableColumn<productData, String> inventory_col_date;

@FXML
private Button inventory_deleteBtn;

@FXML
private AnchorPane inventory_form;

@FXML
private ImageView inventory_imageView;

@FXML
private Button inventory_importBtn;

@FXML
private Button inventory_updateBtn;

@FXML
private AnchorPane main_form;
```

```
@FXML
private Button menu_btn;

@FXML
private Label username;

@FXML
private Button logoutBtn;

@FXML
private TextField inventory_price;

@FXML
private TextField inventory_productID;

@FXML
private TextField inventory_productName;

@FXML
private TextField inventory_stock;

@FXML
```

```
private ComboBox<?> inventory_status;

@FXML
private ComboBox<?> inventory_type;


@FXML
private TextField menu_amount;


@FXML
private Label menu_change;

@FXML
private TableColumn<?, ?> menu_col_price;

@FXML
private TableColumn<?, ?> menu_col_productName;

@FXML
private TableColumn<?, ?> menu_col_quantity;

@FXML
```

```
private AnchorPane menu_form;

@FXML
private GridPane menu_gridPane;

@FXML
private Button menu_payBtn;

@FXML
private Button menu_receiptBtn;

@FXML
private Button menu_removeBtn;

@FXML
private ScrollPane menu_scrollPane;

@FXML
private TableView<productData> menu_tableView;

@FXML
private Label menu_total;
```

```
private Alert alert;

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

private Image image;

private ObservableList<productData> cardListData = FXCollections.observableArrayList();

public void inventoryAddBtn() {

    if (inventory_productID.getText().isEmpty()
            || inventory_productName.getText().isEmpty()
            || inventory_type.getSelectionModel().getSelectedItem() == null
            || inventory_stock.getText().isEmpty()
            || inventory_price.getText().isEmpty()
            || inventory_status.getSelectionModel().getSelectedItem() == null
            || data.path == null) {

        alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error Message");
```

```
    alert.setHeaderText(null);
    alert.setContentText("Please fill all blank fields");
    alert.showAndWait();

} else {

    String checkProdID = "SELECT prod_id FROM product WHERE prod_id = '"
            + inventory_productID.getText() + "'";

    connect = Database.connectdb();
    try {

        statement = connect.createStatement();
        result = statement.executeQuery(checkProdID);

        if (result.next()) {
            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error Message");
            alert.setHeaderText(null);
            alert.setContentText(inventory_productID.getText() + " is already taken");
            alert.showAndWait();
        } else {
            String insertData = "INSERT INTO product"
```

```
        + "(prod_id, prod_name, type, stock, price, status, image, date) "
        + "VALUES(?,?,?,?,?,?,?,?)";

prepare = connect.prepareStatement(insertData);
prepare.setString(1, inventory_productID.getText());
prepare.setString(2, inventory_productName.getText());
prepare.setString(3, (String) inventory_type.getSelectionModel().getSelectedItem());
prepare.setString(4, inventory_stock.getText());
prepare.setString(5, inventory_price.getText());
prepare.setString(6, (String) inventory_status.getSelectionModel().getSelectedItem());

String path = data.path;
path = path.replace("\\", "\\\\");

prepare.setString(7, path);

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

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

prepare.executeUpdate();
```

```
            alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Error Message");
            alert.setHeaderText(null);
            alert.setContentText("Successfully Added!");
            alert.showAndWait();

            inventoryShowData();
            inventoryClearBtn();

        }

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

public void inventoryUpdateBtn() {

  if (inventory_productID.getText().isEmpty()
        || inventory_productName.getText().isEmpty()
        || inventory_type.getSelectionModel().getSelectedItem() == null
        || inventory_stock.getText().isEmpty()
```

```
        || inventory_price.getText().isEmpty()
        || inventory_status.getSelectionModel().getSelectedItem() == null
        || data.path == null || data.id == 0) {

    alert = new Alert(AlertType.ERROR);
    alert.setTitle("Error Message");
    alert.setHeaderText(null);
    alert.setContentText("Please fill all blank fields");
    alert.showAndWait();

} else {

    String path = data.path;
    path = path.replace("\\", "\\\\");

    String updateData = "UPDATE product SET "
            + "prod_id = '" + inventory_productID.getText() + "', prod_name = '"
            + inventory_productName.getText() + "', type = '"
            + inventory_type.getSelectionModel().getSelectedItem() + "', stock = '"
            + inventory_stock.getText() + "', price = '"
            + inventory_price.getText() + "', status = '"
            + inventory_status.getSelectionModel().getSelectedItem() + "', image = '"
            + data.path + "', date = '"
```

```
            + data.date + "' WHERE id = " + data.id;

        connect = Database.connectdb();

        try {

            alert = new Alert(AlertType.CONFIRMATION);
            alert.setTitle("Error Message");
            alert.setHeaderText(null);
            alert.setContentText("Are you sure you want to UPDATE PRoduct ID: " +
inventory_productID.getText() + "?");
            Optional<ButtonType> option = alert.showAndWait();

            if (option.get().equals(ButtonType.OK)) {
                prepare = connect.prepareStatement(updateData);
                prepare.executeUpdate();

                alert = new Alert(AlertType.INFORMATION);
                alert.setTitle("Error Message");
                alert.setHeaderText(null);
                alert.setContentText("Successfully Updated!");
                alert.showAndWait();
```

```
            inventoryShowData();


            inventoryClearBtn();
        } else {
            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error Message");
            alert.setHeaderText(null);
            alert.setContentText("Cancelled.");
            alert.showAndWait();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}

public void inventoryDeleteBtn() {
  if (data.id == 0) {

    alert = new Alert(AlertType.ERROR);
    alert.setTitle("Error Message");
    alert.setHeaderText(null);
    alert.setContentText("Please fill all blank fields");
```

```
        alert.showAndWait();


    } else {
        alert = new Alert(AlertType.CONFIRMATION);
        alert.setTitle("Error Message");
        alert.setHeaderText(null);
        alert.setContentText("Are you sure you want to DELETE Product ID: " +
inventory_productID.getText() + "?");
        Optional<ButtonType> option = alert.showAndWait();


        if (option.get().equals(ButtonType.OK)) {
            String deleteData = "DELETE FROM product WHERE id = " + data.id;
            try {
                prepare = connect.prepareStatement(deleteData);
                prepare.executeUpdate();


                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error Message");
                alert.setHeaderText(null);
                alert.setContentText("successfully Deleted!");
                alert.showAndWait();


                inventoryShowData();
```

```
            inventoryClearBtn();

        } catch (Exception e) {
            e.printStackTrace();
        }
    } else {
        alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error Message");
        alert.setHeaderText(null);
        alert.setContentText("Cancelled");
        alert.showAndWait();
    }
  }
}

public void inventoryClearBtn() {

  inventory_productID.setText("");
  inventory_productName.setText("");
  inventory_type.getSelectionModel().clearSelection();
  inventory_stock.setText("");
  inventory_price.setText("");
```

```
        inventory_status.getSelectionModel().clearSelection();
        data.path = "";
        inventory_imageView.setImage(null);

    }

    public void inventoryImportBtn() {

        FileChooser openFile = new FileChooser();
        openFile.getExtensionFilters().add(new FileChooser.ExtensionFilter("Open Image File", "*png",
"*jpg"));

        File file = openFile.showOpenDialog(main_form.getScene().getWindow());

        if (file != null) {

            data.path = file.getAbsolutePath();
            image = new Image(file.toURI().toString(), 200, 117, false, true);

            inventory_imageView.setImage(image);
        }
    }
```

```
public ObservableList<productData> inventoryDataList() {

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

    String sql = "SELECT * FROM product";

    connect = Database.connectdb();

    try {

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

        productData prodData;

        while (result.next()) {

            prodData = new productData(result.getInt("id"),
                    result.getString("prod_id"),
                    result.getString("prod_name"),
                    result.getString("type"),
                    result.getInt("stock"),
                    result.getDouble("price"),
```

```
                result.getString("status"),
                result.getString("image"),
                result.getDate("date"));

            listData.add(prodData);

        }

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

private ObservableList<productData> inventoryListData;

public void inventoryShowData() {
    inventoryListData = inventoryDataList();

    inventory_col_productID.setCellValueFactory(new PropertyValueFactory<>("productId"));
    inventory_col_productName.setCellValueFactory(new PropertyValueFactory<>("productName"));
    inventory_col_type.setCellValueFactory(new PropertyValueFactory<>("type"));
    inventory_col_stock.setCellValueFactory(new PropertyValueFactory<>("stock"));
```

```
    inventory_col_price.setCellValueFactory(new PropertyValueFactory<>("price"));
    inventory_col_status.setCellValueFactory(new PropertyValueFactory<>("status"));
    inventory_col_date.setCellValueFactory(new PropertyValueFactory<>("date"));

    inventory_tableView.setItems(inventoryListData);

}

public void inventorySelectData() {

    productData prodData = inventory_tableView.getSelectionModel().getSelectedItem();
    int num = inventory_tableView.getSelectionModel().getSelectedIndex();

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

    inventory_productID.setText(prodData.getProductId());
    inventory_productName.setText(prodData.getProductName());
    inventory_stock.setText(String.valueOf(prodData.getStock()));
    inventory_price.setText(String.valueOf(prodData.getPrice()));

    data.path = prodData.getImage();
```

```
    String path = "File:" + prodData.getImage();

    data.date = String.valueOf(prodData.getDate());
    data.id = prodData.getId();

    image = new Image(data.path, 200, 117, false, true);
    inventory_imageView.setImage(image);
}

private String[] typeList = {"Range Weapon", "Melee Weapon"};

public void inventoryTypeList() {

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

    for (String data : typeList) {
        typeL.add(data);
    }

    ObservableList listData = FXCollections.observableArrayList(typeL);
    inventory_type.setItems(listData);
}
```

```
private String[] statusList = {"Available", "Unavaliable"};

public void inventoryStatusList() {

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

    for (String data : statusList) {
        statusL.add(data);
    }

    ObservableList listData = FXCollections.observableArrayList(statusL);
    inventory_status.setItems(listData);
}

public ObservableList<productData> MenuGetData() {
    String sql = "SELECT * FROM product";

    ObservableList<productData> listData = FXCollections.observableArrayList();
    connect = Database.connectdb();

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

```
        result = prepare.executeQuery();

        productData prod;

        while (result.next()) {
            prod = new productData(result.getInt("id"),
                    result.getString("prod_id"),
                    result.getString("prod_name"),
                    result.getString("type"),
                    result.getInt("stock"),
                    result.getDouble("price"),
                    result.getString("image"),
                    result.getDate("date"));

            listData.add(prod);
        }

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

```
public void menuDisplayCard() {

    cardListData.clear();
    cardListData.addAll(MenuGetData());

    int row = 0;
    int column = 0;

    menu_gridPane.getChildren().clear();
    menu_gridPane.getRowConstraints().clear();
    menu_gridPane.getColumnConstraints().clear();

    for (int i = 0; i < cardListData.size(); i++) {

        try {
            FXMLLoader load = new FXMLLoader();
            load.setLocation(getClass().getResource("/org/valo/view/cardProduct.fxml"));
            AnchorPane pane = load.load();
            cardProductController cardC = load.getController();
            cardC.setData(cardListData.get(i));

            if (column == 2) {
                column = 0;
```

```
            row += 1;
        }

        menu_gridPane.add(pane, column++, row);
        GridPane.setMargin(pane, new Insets(10));

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

public ObservableList<productData> menuGetOrder() {
  CustomerID();
  ObservableList<productData> listData = FXCollections.observableArrayList();
  String sql = "SELECT * FROM customer WHERE customer_id = " + cID;
  connect = Database.connectdb();

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

    productData prod;
```

```
        while (result.next()) {
            prod = new productData(result.getInt("id"),
                    result.getString("prod_id"),
                    result.getString("prod_name"),
                    result.getString("type"),
                    result.getInt("quantity"),
                    result.getDouble("price"),
                    result.getString("image"),
                    result.getDate("date"));
            listData.add(prod);
        }

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

private ObservableList<productData> menuOrderListData;

public void menuShowOrderData() {
    menuOrderListData = menuGetOrder();
```

```
    menu_col_productName.setCellValueFactory(new PropertyValueFactory<>("productName"));
    menu_col_quantity.setCellValueFactory(new PropertyValueFactory<>("quantity"));
    menu_col_price.setCellValueFactory(new PropertyValueFactory<>("price"));

    menu_tableView.setItems(menuOrderListData);
}

private int getid;
public void menuSelectOrder() {
    productData prod = (productData) menu_tableView.getSelectionModel().getSelectedItem();
    int num = menu_tableView.getSelectionModel().getSelectedIndex();

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

    getid = prod.getId();

}
private double totalP;

public void menuGetTotal() {
    CustomerID();
```

```
    String total = "SELECT SUM(price) FROM customer WHERE customer_id = " + cID;
    connect = Database.connectdb();

    try {

        prepare = connect.prepareStatement(total);
        result = prepare.executeQuery();

        if (result.next()) {
            totalP = result.getDouble("SUM(price)");

        }

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

public void menuDisplayTotal() {
    menuGetTotal();
    menu_total.setText("Rp. " + totalP);
}
private double amount;
```

```
private double change;

public void menuAmount() {
    menuGetTotal();
    if (menu_amount.getText().isEmpty() || totalP == 0) {
        alert = new Alert(AlertType.ERROR);
        alert.setTitle("Error Message");
        alert.setHeaderText(null);
        alert.setContentText("Invalid");
        alert.showAndWait();
    } else {
        amount = Double.parseDouble(menu_amount.getText());

        if (amount < totalP) {
            menu_amount.setText("");
        } else {
            change = (amount - totalP);
            menu_change.setText("Rp. " + change);
        }
    }
}

public void menuPayBtn() {
```

```
if (totalP == 0) {
    alert = new Alert(AlertType.ERROR);
    alert.setTitle("Error Message");
    alert.setHeaderText(null);
    alert.setContentText("Silakan pilih pesanan Anda terlebih dahulu!");
    alert.showAndWait();
} else {
    menuGetTotal();
    String insertPay = "INSERT INTO receipt (customer_id, total, date, em_username) "
            + "VALUES(?,?,?,?)";

    connect = Database.connectdb();

    try {

        if (amount == 0) {
            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error Messaged");
            alert.setHeaderText(null);
            alert.setContentText("Ada yang salah");
            alert.showAndWait();
        } else {
```

```
alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Message");
alert.setHeaderText(null);
alert.setContentText("Apa Anda yakin?");
Optional<ButtonType> option = alert.showAndWait();

if (option.get().equals(ButtonType.OK)) {
    CustomerID();
    menuGetTotal();
    prepare = connect.prepareStatement(insertPay);
    prepare.setString(1, String.valueOf(cID));
    prepare.setString(2, String.valueOf(totalP));

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

    prepare.setString(3, String.valueOf(sqlDate));
    prepare.setString(4, data.username);

    prepare.executeUpdate();

    alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Infomation Message");
```

```
                alert.setHeaderText(null);
                alert.setContentText("Sukses.");
                alert.showAndWait();

                menuShowOrderData();

            } else {
                alert = new Alert(AlertType.WARNING);
                alert.setTitle("Infomation Message");
                alert.setHeaderText(null);
                alert.setContentText("Dibatalkan.");
                alert.showAndWait();
            }
        }

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

}

public void menuRemoveBtn() {
```

```
if (getid == 0) {
    alert = new Alert(AlertType.ERROR);
    alert.setTitle("Error Message");
    alert.setHeaderText(null);
    alert.setContentText("Silakan pilih pesanan yang ingin Anda hapus");
    alert.showAndWait();
} else {
    String deleteData = "DELETE FROM customer WHERE id = " + getid;
    connect = Database.connectdb();
    try {
        alert = new Alert(AlertType.CONFIRMATION);
        alert.setTitle("Confirmation Message");
        alert.setHeaderText(null);
        alert.setContentText("Apakah Anda yakin ingin menghapus pesanan ini?");
        Optional<ButtonType> option = alert.showAndWait();

        if (option.get().equals(ButtonType.OK)) {
            prepare = connect.prepareStatement(deleteData);
            prepare.executeUpdate();
        }

        menuShowOrderData();
```

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

    }
}

private int cID;

public void CustomerID() {
    String sql = "SELECT MAX(customer_id) FROM customer";
    connect = Database.connectdb();

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

        if (result.next()) {
            cID = result.getInt("MAX(customer_id)");
        }

        String checkCID = "SELECT MAX(customer_id) FROM receipt";
        prepare = connect.prepareStatement(checkCID);
```

```
        result = prepare.executeQuery();
        int checkID = 0;
        if (result.next()) {
            checkID = result.getInt("MAX(customer_id)");
        }

        if (cID == 0) {
            cID += 1;
        } else if (cID == checkID) {
            cID += 1;
        }

        data.cID = cID;

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

public ObservableList<customerData> customersDataList() {

    ObservableList<customerData> listData = FXCollections.observableArrayList();
    String sql = "SELECT * FROM receipt";
```

```
    connect = Database.connectdb();

    try {

        prepare = connect.prepareStatement(sql);
        result = prepare.executeQuery();
        customerData cData;

        while (result.next()) {
            cData = new customerData(result.getInt("id"),
                    result.getInt("customer_id"),
                    result.getDouble("total"),
                    result.getDate("date"),
                    result.getString("em_username"));

            listData.add(cData);
        }

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

```
private ObservableList<customerData> customersListData;

public void customersShowData() {
    menuOrderListData = menuGetOrder();

    menu_col_productName.setCellValueFactory(new PropertyValueFactory<>("productName"));
    menu_col_quantity.setCellValueFactory(new PropertyValueFactory<>("quantity"));
    menu_col_price.setCellValueFactory(new PropertyValueFactory<>("price"));

    menu_tableView.setItems(menuOrderListData);
}


public void switchForm(ActionEvent event) {

    if (event.getSource() == inventory_btn) {
        inventory_form.setVisible(true);
        menu_form.setVisible(false);

        inventoryTypeList();
        inventoryStatusList();
        inventoryShowData();
```

```
    } else if(event.getSource() == menu_btn) {
        inventory_form.setVisible(false);
        menu_form.setVisible(true);

        menuDisplayCard();
        menuGetOrder();
        menuDisplayTotal();
        menuShowOrderData();



    }

}

public void logout() {

    try {

        alert = new Alert(AlertType.CONFIRMATION);
        alert.setTitle("Error Message");
        alert.setContentText("Are you sure want to logout?");
        Optional<ButtonType> option = alert.showAndWait();
```

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

            logoutBtn.getScene().getWindow().hide();

            Parent root = FXMLLoader.load(getClass().getResource("/org/valo/view/loginView.fxml"));

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

            stage.setTitle("weapon market");

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

        }

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

}

public void displayUsername() {
```

```
    String user = data.username;
    user = user.substring(0, 1).toUpperCase() + user.substring(1);

    username.setText(user);
}

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

    displayUsername();

    inventoryTypeList();
    inventoryStatusList();
    inventoryShowData();

    menuDisplayCard();
    menuGetOrder();
    menuDisplayTotal();
    menuShowOrderData();
    customersShowData();

}
```

}

cardProduct.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package org.valo.controller;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.net.URL;
import java.sql.SQLException;
import java.util.Date;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
```

```
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Spinner;
import javafx.scene.control.SpinnerValueFactory;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import org.valo.database.Database;


/**
 *
 * @author HD
 */
public class cardProductController implements Initializable {

    @FXML
    private AnchorPane card_form;

    @FXML
    private Button prod_addBtn;

    @FXML
```

```
private ImageView prod_imageView;

@FXML
private Label prod_name;

@FXML
private Label prod_price;

@FXML
private Spinner<Integer> prod_spinner;

private String prod_ID;
private productData prodData;
private Image image;
private SpinnerValueFactory<Integer> spin;
private Connection connect;
private ResultSet result;
private Statement statement;
private PreparedStatement prepare;
private Alert alert;
private String prodID;
private String type;
private String prod_image;
```

```
private String prod_date;

public void setData(productData prodData) {
    this.prodData = prodData;
    prod_date = String.valueOf(prodData.getDate());
    prod_image = prodData.getImage();
    type = prodData.getType();
    prodID = prodData.getProductId();
    prod_name.setText(prodData.getProductName());
    prod_price.setText("Rp. " + String.valueOf(prodData.getPrice()));
    String path = "File:" + prodData.getImage();
    image = new Image(path, 200, 117, false, true);
    prod_imageView.setImage(image);
    pr = prodData.getPrice();
}
private int qty;

public void setQuantity() {
    spin = new SpinnerValueFactory.IntegerSpinnerValueFactory(0, 100, 0);
    prod_spinner.setValueFactory(spin);
}
private double totalP;
private double pr;
```

```
public void addBtn() throws SQLException {

    MainViewController mForm = new MainViewController();
    mForm.CustomerID();

    qty = prod_spinner.getValue();
    String check = "";
    String checkAvailable = "SELECT status FROM product WHERE prod_id = '"
            + prodID + "'";

    connect = Database.connectdb();

    try {
        int checkStck = 0;

        String checkStock = "SELECT stock FROM product WHERE prod_id = '"
                + prodID + "'";
        prepare = connect.prepareStatement(checkStock);
        result = prepare.executeQuery();

        if (result.next()) {
            checkStck = result.getInt("stock");
```

```
}

if (checkStck == 0) {

    String updateStock = "UPDATE product SET prod_name = '"
            + prod_name.getText() + "', type = '"
            + type + "', stock = 0, price = " + pr
            + ", status = 'Unavailable', image = '"
            + prod_image + "', date = '"
            + prod_date + "' WHERE prod_id = '"
            + prodID + "'";
    prepare = connect.prepareStatement(updateStock);
    prepare.executeUpdate();

}

prepare = connect.prepareStatement(checkAvailable);
result = prepare.executeQuery();

if (result.next()) {
    check = result.getString("status");
}
```

```
        if (!check.equals("Available") || qty == 0) {
            alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error Message");
            alert.setHeaderText(null);
            alert.setContentText("Something Wrong :3");
            alert.showAndWait();
        } else {

            if (checkStck < qty) {

                alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error Message");
                alert.setHeaderText(null);
                alert.setContentText("Gagal. Barang tidak tersedia!");
                alert.showAndWait();

            } else {
                String insertData = "INSERT INTO customer"
                        + "(customer_id, prod_id, prod_name, type, quantity, price, date, image,
em_username)"
                        + "VALUES(?,?,?,?,?,?,?,?,?)";
                prepare = connect.prepareStatement(insertData);
```

```
prepare.setString(1, String.valueOf(data.cID));
prepare.setString(2, prodID);
prepare.setString(3, prod_name.getText());
prepare.setString(4, type);
prepare.setString(5, String.valueOf(qty));

totalP = (qty * pr);
prepare.setString(6, String.valueOf(totalP));

Date date = new Date();
java.sql.Date sqlDate = new java.sql.Date(date.getTime());
prepare.setString(7, String.valueOf(sqlDate));

prepare.setString(8, prod_image);
prepare.setString(9, data.username);

prepare.executeUpdate();

int upStock = checkStck - qty;
prod_image = prod_image.replace("\\", "\\\\");

System.out.println("Date: " + prod_date);
System.out.println("Image: " + prod_image);
```

```
            String updateStock = "UPDATE product SET prod_name = '"
                    + prod_name.getText() + "', type = '"
                    + type + "', stock = " + upStock + ", price = " + pr
                    + ", status = '"
                    + check + "', image = '"
                    + prod_image + "', date = '"
                    + prod_date + "' WHERE prod_id = '"
                    + prodID + "'";

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

            alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Information Message");
            alert.setHeaderText(null);
            alert.setContentText("Successfully Added!");
            alert.showAndWait();

            //mForm.menuGetTotal();
        }

    }
} catch (Exception e) {
```

```
            e.printStackTrace();
        }

    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        setQuantity();
    }

}
```

ccustomerData.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package org.valo.controller;

import java.sql.Date;

/**
```

```
 *
 * @author WINDOWS 10
 */
public class customerData {

    private Integer id;
    private Integer customerID;
    private Double total;
    private Date date;
    private String emUsername;

    public customerData(Integer id, Integer customerID, Double total,
            Date date, String emUsername) {
        this.id = id;
        this.customerID = customerID;
        this.total = total;
        this.date = date;
        this.emUsername = emUsername;
    }

    public Integer getId() {
        return id;
    }
```

```
    public Integer getCustomerID() {
        return customerID;
    }

    public Double getTotal() {
        return total;
    }

    public Date getDate() {
        return date;
    }

    public String getEmUsername() {
        return emUsername;
    }

}
```

data.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
```

```
 */
package org.valo.controller;

/**
 *
 * @author USER
 */
public class data {

    public static String username;
    public static String path;
    public static String date;
    public static Integer id;
    public static Integer cID;

}
```

productData.java

```
package org.valo.controller;
import java.sql.Date;

public class productData {
```

```
private Integer id;
private String productId;
private String productName;
private String type;
private Integer stock;
private Double price;
private String status;
private String image;
private Date date;
private Integer quantity;

public productData(Integer id, String productId,
        String productName, String type, Integer stock,
        Double price, String status, String image, Date date) {
    this.id = id;
    this.productId = productId;
    this.productName = productName;
    this.type = type;
    this.stock = stock;
    this.price = price;
    this.status = status;
    this.image = image;
```

```
        this.date = date;
    }

    public productData(Integer id, String productId, String productName,
            String type, Integer quantity, Double price, String image, Date date){
        this.id = id;
        this.productId = productId;
        this.productName = productName;
        this.type = type;
        this.price = price;
        this.image = image;
        this.date = date;
        this.quantity = quantity;
    }

    public productData(Integer id, String productId,String productName,Double price, String status, String
image) {
        this.id = id;
        this.productId = productId;
        this.productName = productName;
        this.type = type;
        this.stock = stock;
        this.price = price;
```

```
    this.status = status;
    this.image = image;
    this.date = date;
}

public Integer getId() {
    return id;
}

public String getProductId() {
    return productId;
}

public String getProductName() {
    return productName;
}

public String getType(){
    return type;
}

public Integer getStock() {
    return stock;
```

```
    }

    public Double getPrice() {
        return price;
    }

    public String getStatus() {
        return status;
    }

    public String getImage() {
        return image;
    }

    public Date getDate() {
        return date;
    }

    public Integer getQuantity(){
        return quantity;
    }
}
```