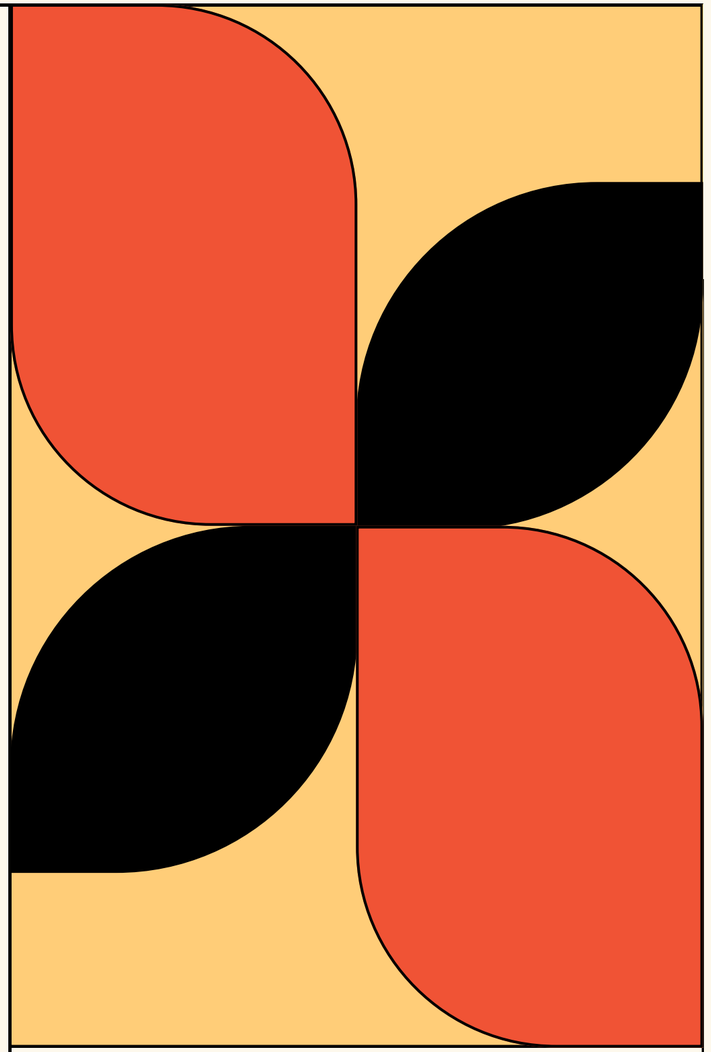
WaMaBase

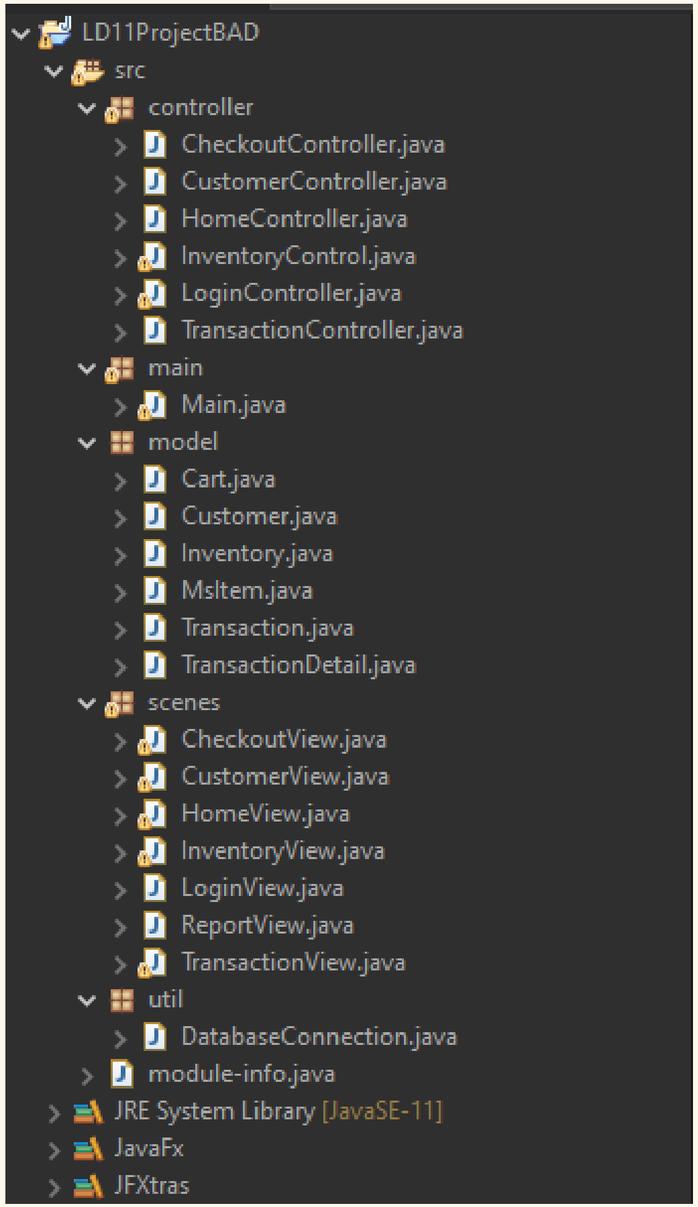
Arya Wira Raja - 2702302984
Avanindra Ikhsan Hafizha - 2702365530
Faris Bayhaqi - 2702378395
Muhamad Fadhil Saputra - 2702365354
Wan Ahmad Ilhamzakky - 2702370202
Bryan Winston Tingginehe - 2702247815
Muhammad Rakhien Althaf - 2702378483



WaMaBase

Aplikasi ini bertujuan membantu pemilik warung dalam mengelola operasional sehari-hari dengan lebih efisien, termasuk dalam hal pelacakan stok barang, koordinasi dengan pemasok, serta pencatatan transaksi. Dengan menggunakan WaMaBase, pemilik warung dapat meminimalkan kerugian akibat kehilangan stok, memantau pemasok dengan lebih terorganisir, dan menyederhanakan manajemen staf. Sistem ini memberikan kemudahan dalam pengelolaan sehingga pemilik warung dapat lebih fokus pada pelayanan pelanggan dan pengembangan usaha mereka.

Application Structure



SQL Structure

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	inventory	Browse Structure Search Insert Empty Drop	17	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	mscustomer	Browse Structure Search Insert Empty Drop	25	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	msitem	Browse Structure Search Insert Empty Drop	17	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	msstaff	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	mssupplier	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	paymentmethod	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	transactiondetail	Browse Structure Search Insert Empty Drop	84	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	transactionheader	Browse Structure Search Insert Empty Drop	43	InnoDB	utf8mb4_general_ci	64.0 KiB	-
☐	transactionsupplydetail	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	transactionsupplyheader	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
	10 tables	Sum	205	InnoDB	utf8mb4_general_ci	272.0 KiB	0 B

DatabaseConnection.java

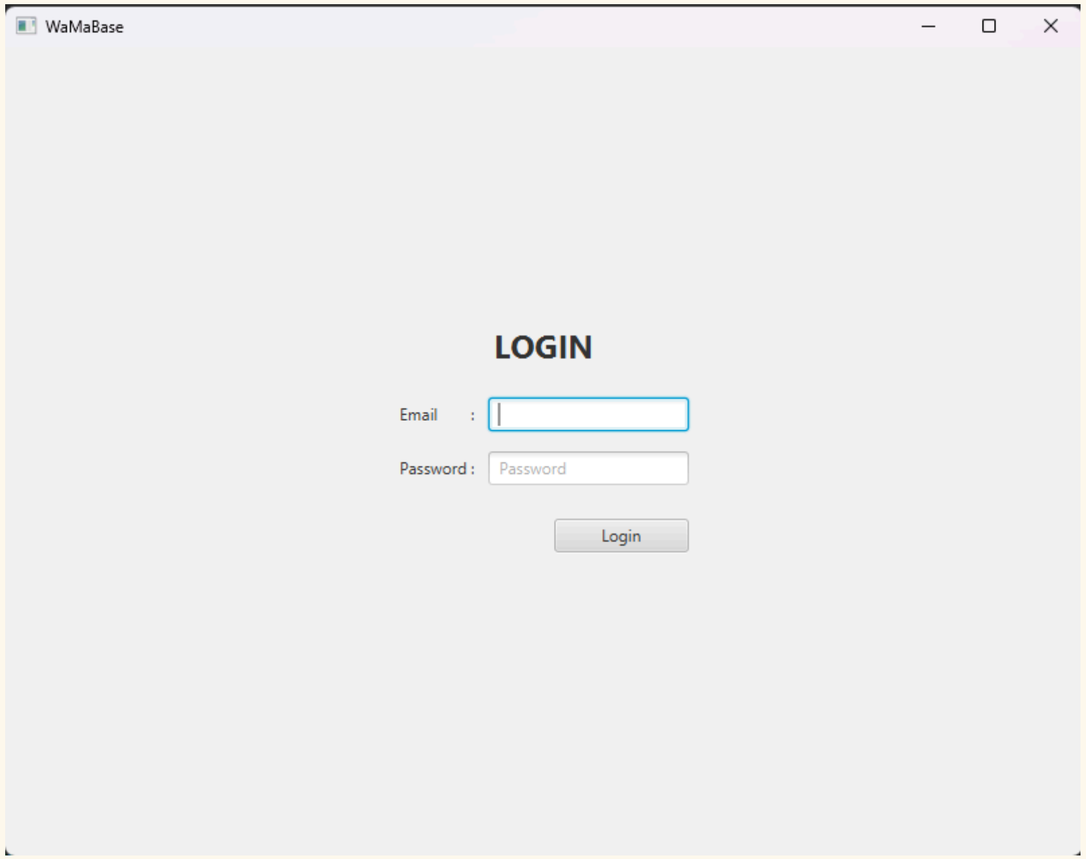
```java
package util;

import java.sql.Connection;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/wamabase";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    public static Connection getConnection() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
//            System.out.println("Connected to database: " + conn.getCatalog());
            return conn;
        } catch (ClassNotFoundException e) {
            throw new SQLException("MySQL JDBC Driver not found.", e);
        }
    }

    public static void main(String[] args) {
        try {
            Connection conn = getConnection();
//            System.out.println("Connection successful!");
            conn.close();
        } catch (SQLException e) {
            System.out.println("Connection failed. Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Main.java

Menyimpan launch(args), psv, dan sebagai connector antar page.

```java
public class Main extends Application{
    private static Stage primaryStage;

    @Override
    public void start(Stage primaryStage) throws Exception {
        this.primaryStage = primaryStage;
        showLoginPage();
        primaryStage.setResizable(false);
        primaryStage.setTitle("KingsHcut");
        primaryStage.show();
    }


    public static void showLoginPage() {
        LoginPage loginPage = new LoginPage();
        primaryStage.setScene(loginPage.getScene());
    }

    public static void showRegisterPage() {
        System.out.println("Register");
        RegisterPage regPage = new RegisterPage();
        primaryStage.setScene(regPage.getScene());
    }

    public static void showReserveServicePage(String username, String userID) {
        System.out.println("ReserveService");
        ReserveServicePage reserverServicePage = new ReserveServicePage(username, userID);
        primaryStage.setScene(reserverServicePage.getScene());
    }

    public static void showCustomerReservationPage(String username, String userID) {
        System.out.println("CustomerReservation");
        CustomerReservationPage custReservationPage = new CustomerReservationPage(username, userID);
        primaryStage.setScene(custReservationPage.getScene());
    }

    public static void showAdminServiceManagementPage(String username, String userID) {
        AdminServiceManagementPage adServiceManagementPage = new AdminServiceManagementPage(username, userID);
        primaryStage.setScene(adServiceManagementPage.getScene());
    }

    public static void showAdminReservationManagementPage(String username, String userID) {
        AdminReservationManagementPage adReservationManagementPage = new AdminReservationManagementPage(username, userID);
        primaryStage.setScene(adReservationManagementPage.getScene());
    }

    public static void showAlert(AlertType alertType, String title, String header, String content) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(header);
        alert.setContentText(content);
        alert.show();
    }

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

LOGIN



LoginView.java

Meng-inisialisasi page Login dan inisialisasi event handling

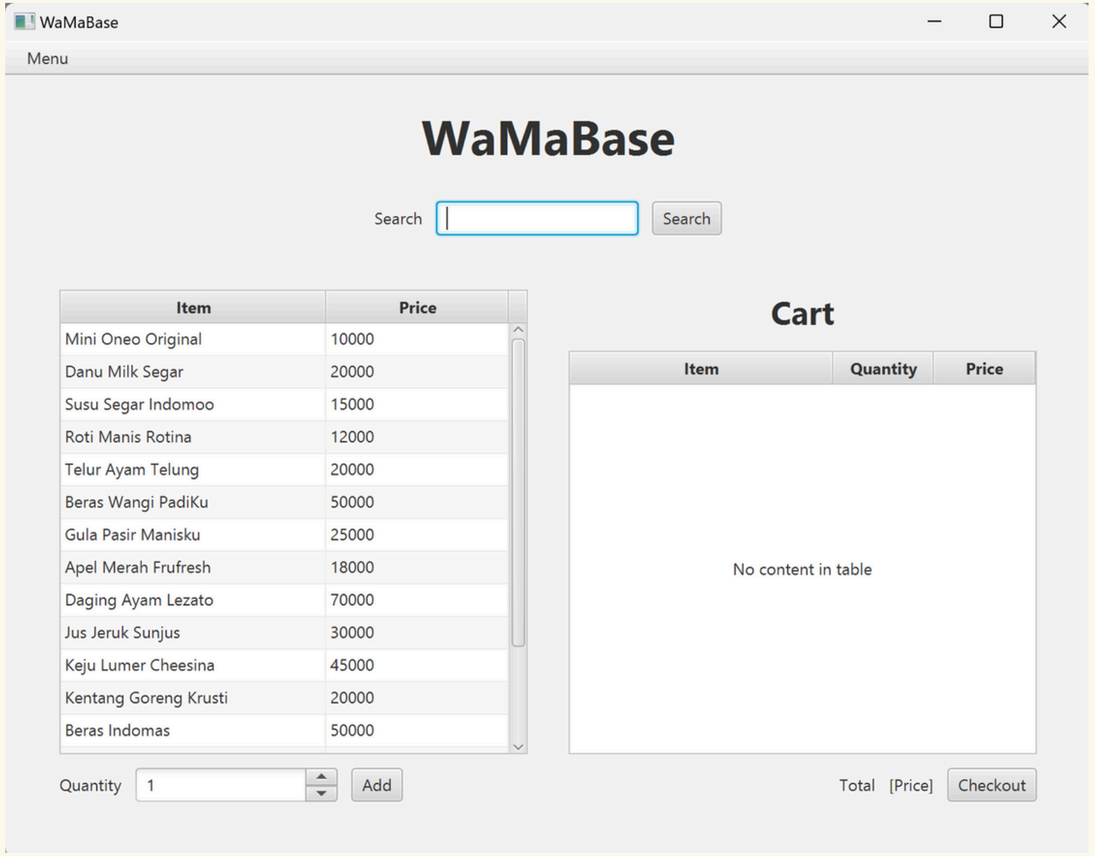
```java
package scenes;

import controller.LoginController;

public class LoginView {

    private Scene scene;
    private Label loginLbl, emailLbl, passwordLbl;
    private TextField emailFld;
    private PasswordField passwordFld;
    private Button loginBtn;
    private VBox mainContainer, buttonContainer;
    private GridPane formContainer;
    private BorderPane baseContainer;

    public LoginView() {
        initialize();
        styling();
        action();
    }

    private void initialize() {
        baseContainer = new BorderPane();
        scene = new Scene(baseContainer, 800, 600);

        loginLbl = new Label("LOGIN");

        // Form elements
        formContainer = new GridPane();
        emailLbl = new Label("Email\t:");
        emailFld = new TextField();
        passwordLbl = new Label("Password\t:");
        passwordFld = new PasswordField();

        formContainer.addRow(0, emailLbl, emailFld);
        formContainer.addRow(1, passwordLbl, passwordFld);

        // Login button
        buttonContainer = new VBox();
        loginBtn = new Button("Login");
        buttonContainer.getChildren().add(loginBtn);

        formContainer.add(buttonContainer, 1, 2);

        // Main container
        mainContainer = new VBox();
        mainContainer.getChildren().addAll(loginLbl, formContainer);
        baseContainer.setCenter(mainContainer);
    }

    private void styling() {
        loginLbl.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");
        emailFld.setPromptText("Email");
        passwordFld.setPromptText("Password");
        loginBtn.setPrefWidth(100);

        formContainer.setVgap(15);
        formContainer.setHgap(10);
        formContainer.setPadding(new Insets(20));
        formContainer.setAlignment(Pos.CENTER);

        buttonContainer.setAlignment(Pos.CENTER_RIGHT);
        buttonContainer.setPadding(new Insets(10, 0, 0, 0));

        mainContainer.setPadding(new Insets(30));
        mainContainer.setAlignment(Pos.CENTER);
        BorderPane.setAlignment(mainContainer, Pos.CENTER);
    }

    private void action() {
        loginBtn.setOnAction(e->{
            String email = emailFld.getText();
            String password = passwordFld.getText();
            if(email.isEmpty() || password.isEmpty()) {
                Main.showAlert(AlertType.ERROR, "Error", "Validation Error", "Email or Password can't be empty!");
                return ;
            }
            if(LoginController.login(emailFld.getText(),passwordFld.getText())) {
                Main.showHomePage();
            } else {
                Main.showAlert(AlertType.ERROR, "Error", "Validation Error", "Email and/or Password does not match");
            }
        });
    }

    public Scene getScene() {
        return scene;
    }

}
```

LoginController.java

Meng-handle backend dari Login Page

```java
package controller;

import java.sql.Connection;

public class LoginController {

    public static boolean login(String email, String password) {

        try(Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT * FROM msstaff WHERE StaffEmail = ? AND StaffPassword = ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, email);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                return true;
            }
        }
        catch (SQLException e) {
            e.printStackTrace();
            Main.showAlert(AlertType.ERROR, "Error","Database Error", "An error has ocurred in the database error");
        }

        return false;
    }

}
```

HOME


WaMaBase
Menu
WaMaBase
Search
Search
Item
Price
Mini Oneo Original
10000
Danu Milk Segar
20000
Susu Segar Indomoo
15000
Roti Manis Rotina
12000
Telur Ayam Telung
20000
Beras Wangi PadiKu
50000
Gula Pasir Manisku
25000
Apel Merah Frufresh
18000
Daging Ayam Lezato
70000
Jus Jeruk Sunjus
30000
Keju Lumer Cheesina
45000
Kentang Goreng Krusti
20000
Beras Indomas
50000
Cart
Item
Quantity
Price
No content in table
Quantity
1
Add
Total
[Price]
Checkout

HomeView.java

Meng-inisialisasi page Login dan inisialisasi event handling

```java
package scenes;

import controller.HomeController;

public class HomeView {

    private static Scene scene;
    private MenuBar menuBar;
    private Menu menuOptions;
    private MenuItem home, inventory, customer, report, logout;
    private TableView<MsItem> leftTable;
    private TableView<Cart> rightTable;
    private TableColumn<MsItem, String> itemColumn;
    private TableColumn<Cart, String> cartItemColumn;
    private TableColumn<Cart, Integer> cartQuantityColumn;
    private TableColumn<Cart, Integer> cartPriceColumn;
    private TableColumn<MsItem, Integer> priceColumn;
    private TextField searchField;
    private Button searchButton, addButton, checkoutButton;
    private Spinner<Integer> quantitySpinner;
    private Label totalLabel, priceLabel;
    private HBox searchLayout, quantityLayout, checkoutLayout;
    private VBox leftTableLayout, rightTableLayout;
    private HBox tableLayout;
    private BorderPane root;

    private ObservableList<MsItem> itemList = FXCollections.observableArrayList();
    public static ObservableList<Cart> cartList = FXCollections.observableArrayList();
    private MsItem selectedItem;

    public HomeView() {
        initialize();
        populateTables();
        setTableColumns();
        setTableListener();
        action();
    }
```

```java
    private void initialize() {
        // Create the MenuBar
        menuBar = new MenuBar();

        // Create Menus and MenuItems
        menuOptions = new Menu("Menu");
        home = new MenuItem("Home");
        home.setDisable(true);
        inventory = new MenuItem("Inventory");
        customer = new MenuItem("Customer");
        report = new MenuItem("Report");
        logout = new MenuItem("Log Out");

        // Add MenuItems to the Menu
        menuOptions.getItems().addAll(home, inventory, customer, report, new SeparatorMenuItem(), logout);
        menuBar.getMenus().add(menuOptions);

        // Create Left Table (Items and Prices)
        leftTable = new TableView<MsItem>();
        leftTable.setPlaceholder(new Label("No content in table"));
        itemColumn = new TableColumn<MsItem, String>("Item");
        priceColumn = new TableColumn<MsItem, Integer>("Price");
        leftTable.getColumns().addAll(itemColumn, priceColumn);
        leftTable.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

        leftTable.setPrefWidth(350);
        leftTable.setPrefHeight(400);

        // Create Right Table (Cart: Item, Quantity, Price)
        rightTable = new TableView<Cart>();
        rightTable.setPlaceholder(new Label("No content in table"));
        cartItemColumn = new TableColumn<Cart, String>("Item");
        cartQuantityColumn = new TableColumn<Cart, Integer>("Quantity");
        cartPriceColumn = new TableColumn<Cart, Integer>("Price");
        rightTable.getColumns().addAll(cartItemColumn, cartQuantityColumn, cartPriceColumn);
        rightTable.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
        cartItemColumn.setPrefWidth(200);
        rightTable.setPrefWidth(350);
        rightTable.setPrefHeight(400);

        // Add Cart Label
        Label cartLabel = new Label("Cart");
        cartLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

        // Add WaMaBase Label
        Label titleLabel = new Label("WaMaBase");
        titleLabel.setStyle("-fx-font-size: 36px; -fx-font-weight: bold;");
        titleLabel.setAlignment(Pos.CENTER);

        // Search and Quantity Inputs
        searchField = new TextField();
        searchField.setPromptText("Search");
        searchButton = new Button("Search");
        quantitySpinner = new Spinner<>(1, 100, 1);
        addButton = new Button("Add");
```

```
        // Layout for Search
        searchLayout = new HBox(10, new Label("Search"), searchField, searchButton);
        searchLayout.setAlignment(Pos.CENTER);
        searchLayout.setPadding(new Insets(0, 20, 0, 20));

        // Layout for Quantity
        quantityLayout = new HBox(10, new Label("Quantity"), quantitySpinner, addButton);
        quantityLayout.setAlignment(Pos.CENTER_LEFT);

        // Cart Total and Checkout
        totalLabel = new Label("Total");
        priceLabel = new Label("[Price]");
        checkoutButton = new Button("Checkout");

        checkoutLayout = new HBox(10, totalLabel, priceLabel, checkoutButton);
        checkoutLayout.setAlignment(Pos.CENTER_RIGHT);

        // Layout for Tables
        leftTableLayout = new VBox(10, leftTable, quantityLayout);
        rightTableLayout = new VBox(10, cartLabel, rightTable, checkoutLayout);
        rightTableLayout.setAlignment(Pos.CENTER);

        // Main layout for the tables
        tableLayout = new HBox(30, leftTableLayout, rightTableLayout);
        tableLayout.setPadding(new Insets(20, 20, 20, 20));
        tableLayout.setAlignment(Pos.CENTER);

        VBox mainLayout = new VBox(20, titleLabel, searchLayout, tableLayout);
        mainLayout.setPadding(new Insets(20));
        mainLayout.setAlignment(Pos.CENTER);

        // Main container
        BorderPane root = new BorderPane();
        root.setTop(menuBar);
        root.setCenter(mainLayout);
        root.setTop(new VBox(menuBar));

        // Set the scene
        scene = new Scene(root, 800, 600);
    }

    private void populateTables() {
        itemList = HomeController.populateTable();
        leftTable.setItems(FXCollections.observableArrayList(itemList));

        rightTable.setItems(FXCollections.observableArrayList(cartList));
    }

    private void setTableColumns() {
        itemColumn.setCellValueFactory(new PropertyValueFactory<MsItem, String>("ItemName"));
        priceColumn.setCellValueFactory(new PropertyValueFactory<MsItem, Integer>("ItemPrice"));

        cartItemColumn.setCellValueFactory(new PropertyValueFactory<Cart, String>("name"));
        cartQuantityColumn.setCellValueFactory(new PropertyValueFactory<Cart, Integer>("quantity"));
        cartPriceColumn.setCellValueFactory(new PropertyValueFactory<Cart, Integer>("total"));
    }

```

```
    private void setTableListener() {
        leftTable.getSelectionModel().selectedItemProperty().addListener((obs, oldValue, selectedDonut) -> {
            if (selectedDonut != null) {
                this.selectedItem = selectedDonut;
            }
        });
    }

    private void updateCartTotal() {
        int total = 0;
        for(Cart c : cartList) {
            total += c.getTotal();
        }
        priceLabel.setText(String.valueOf(total));
    }
```

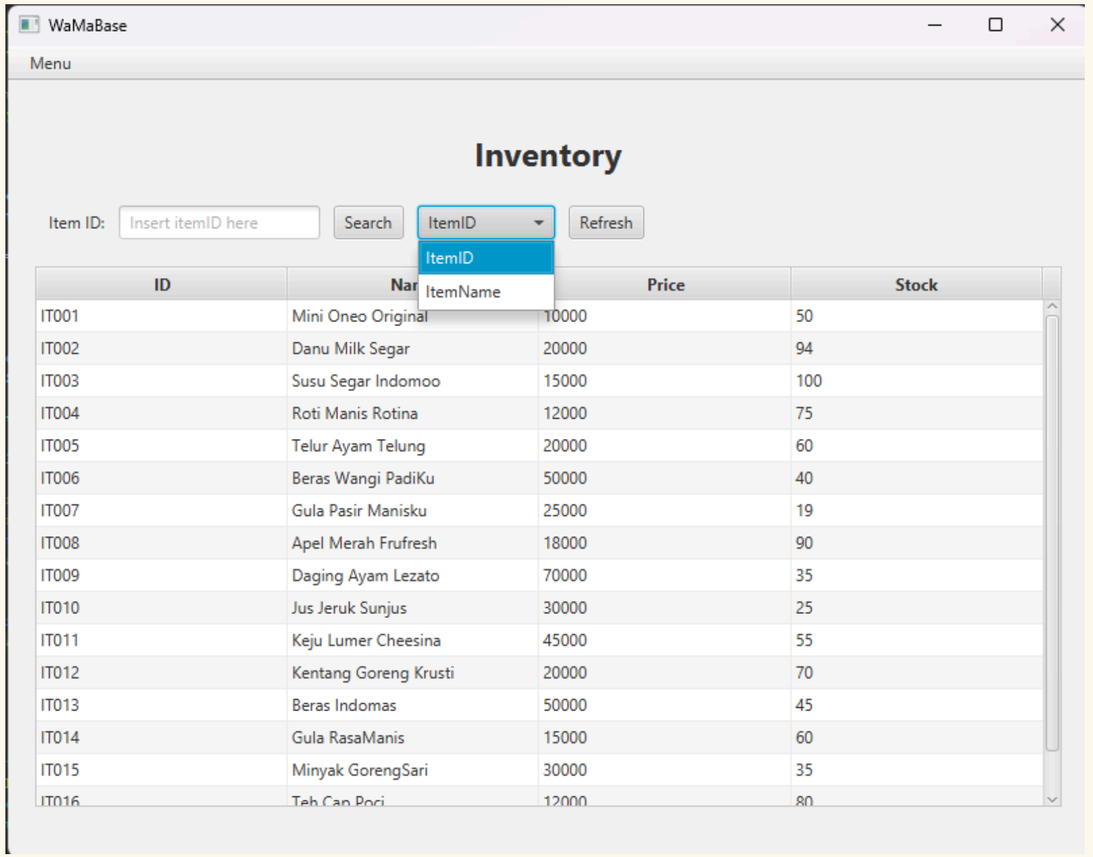
```
    private void action() {
        inventory.setOnAction(e -> {
            Main.showInventoryPage();
        });
        customer.setOnAction(e -> {
            Main.showCustomerPage();
        });
        report.setOnAction(e -> {
            Main.showReportPage();
        });
        logout.setOnAction(e -> {
            Main.showLoginPage();
        });

        addButton.setOnAction(e -> {
            if (selectedItem == null) {
                Main.showAlert(AlertType.ERROR, "Error", "Selection Error", "Please select an item first.");
                return;
            }

            // Get the item details from the selected MsItem
            String itemID = selectedItem.getItemID(); // Use getItemID() from MsItem
            String name = selectedItem.getItemName();
            int quantity = quantitySpinner.getValue();
            int price = quantity * selectedItem.getItemPrice();

            // Add the item to the cart
            cartList.add(new Cart(itemID, name, quantity, price));

            // Reset spinner, update total, and refresh tables
            quantitySpinner.getValueFactory().setValue(1);
            updateCartTotal();
            populateTables();
        });


        searchButton.setOnAction(e -> {
            String search = searchField.getText();
            if(search.isEmpty()) {
                Main.showAlert(AlertType.ERROR, "Error", "Validation Error", "Type in something before searching!");
                return;
            }
            itemList = HomeController.search(search);
            leftTable.setItems(FXCollections.observableArrayList(itemList));
            searchField.clear();
        });

        checkoutButton.setOnAction(e->{
            if(cartList.isEmpty()) {
                Main.showAlert(AlertType.ERROR, "Error", "Validation Error", "Cart is empty!");
                return;
            }
            Main.showCheckoutPage();
        });
    }

    public Scene getScene() {
        return scene;
    }
```

HomeController.java

Meng-handle backend dari Home Page

```java
package controller;

import java.sql.Connection;

public class HomeController {

    public static ObservableList<MsItem> populateTable() {
        ObservableList<MsItem> itemList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT * FROM msitem";
            PreparedStatement ps = conn.prepareStatement(query);
            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
                itemList.add(new MsItem(rs.getString("ItemID"), rs.getString("ItemName"), rs.getInt("ItemPrice")));
            }

        } catch (SQLException e) {

            e.printStackTrace();
        }
        return itemList;
    }

    public static ObservableList<MsItem> search(String search) {
        ObservableList<MsItem> itemList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM msitem\r\n"
                    + "WHERE ItemName REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                itemList.add(new MsItem(rs.getString("ItemID"), rs.getString("ItemName"), rs.getInt("ItemPrice")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return itemList;
    }

}
```

INVENTORY


WaMaBase
Menu
Inventory
Item ID:
Insert itemID here
Search
ItemID
Refresh
ItemID
ItemName
ID
Price
Stock
IT001 Mini Oneo Original 10000 50
IT002 Danu Milk Segar 20000 94
IT003 Susu Segar Indomoo 15000 100
IT004 Roti Manis Rotina 12000 75
IT005 Telur Ayam Telung 20000 60
IT006 Beras Wangi PadiKu 50000 40
IT007 Gula Pasir Manisku 25000 19
IT008 Apel Merah Frufresh 18000 90
IT009 Daging Ayam Lezato 70000 35
IT010 Jus Jeruk Sunjus 30000 25
IT011 Keju Lumer Cheesina 45000 55
IT012 Kentang Goreng Krusti 20000 70
IT013 Beras Indomas 50000 45
IT014 Gula RasaManis 15000 60
IT015 Minyak GorengSari 30000 35

InventoryView.java

Meng-inisialisasi page Inventory dan inisialisasi event handling

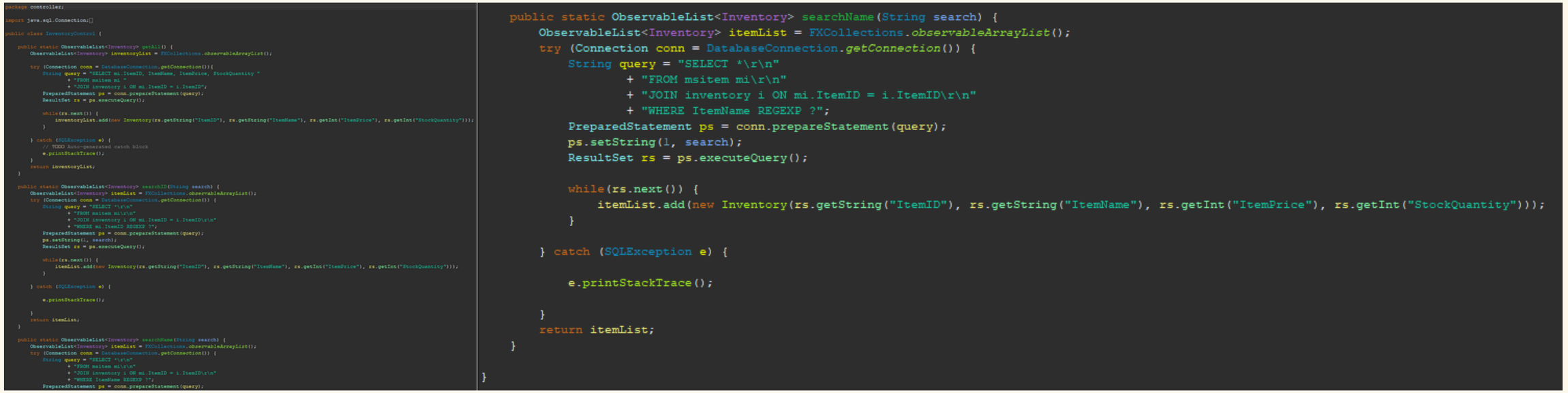
```java
package scenes;

import controller.InventoryControl;

public class InventoryView {

    private static Scene scene;
    private MenuBar menuBar;
    private Menu menuOptions;
    private MenuItem home, inventory, customer, report, logout;
    private Label inv;
    private TableView<Inventory> inventoryTable;
    private TableColumn<Inventory, String> idColumn, nameColumn;
    private TableColumn<Inventory, Integer> priceColumn, stockColumn;
    private ComboBox<String> searchByCB;
    private TextField searchField;
    private Button searchButton, refreshButton;
    private BorderPane root;

    private ObservableList<Inventory> inventoryList = FXCollections.observableArrayList();

    public InventoryView() {
        initialize();
        styling();
        populateTables();
        setTableColumns();
        action();
    }

    @SuppressWarnings({ "unchecked", "deprecation" })
    private void initialize() {
        // Create the MenuBar
        menuBar = new MenuBar();

        // Create Menus and MenuItems
        menuOptions = new Menu("Menu");
        home = new MenuItem("Home");
        inventory = new MenuItem("Inventory");
        inventory.setDisable(true);
        customer = new MenuItem("Customer");
        report = new MenuItem("Report");
        logout = new MenuItem("Log Out");

        // Add MenuItems to the Menu
        menuOptions.getItems().addAll(home, inventory, customer, report, new SeparatorMenuItem(), logout);
        menuBar.getMenus().add(menuOptions);

        // Create Inventory Table
        inventoryTable = new TableView<>();
        inventoryTable.setPlaceholder(new Label("No content in table"));

        idColumn = new TableColumn<Inventory, String>("ID");
        nameColumn = new TableColumn<>("Name");
        priceColumn = new TableColumn<>("Price");
        stockColumn = new TableColumn<>("Stock");

        inventoryTable.getColumns().addAll(idColumn, nameColumn, priceColumn, stockColumn);
        inventoryTable.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
```

```java
        inventoryTable.getColumns().addAll(idColumn, nameColumn, priceColumn, stockColumn);
        inventoryTable.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

        // Create Search Field and Button
        searchField = new TextField();
        searchField.setPromptText("Insert itemID here");
        searchButton = new Button("Search");
        refreshButton = new Button("Refresh");
        searchByCB = new ComboBox<String>();
        searchByCB.getItems().addAll("ItemID", "ItemName");
        searchByCB.setValue("ItemID");

        // Layout for Search
        HBox searchLayout = new HBox(10, new Label("Item ID:"), searchField, searchButton, searchByCB,  refreshButton);
        searchLayout.setAlignment(Pos.CENTER_LEFT);
        searchLayout.setPadding(new Insets(10));

        inv = new Label("Inventory");
        inv.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

        // Main Layout
        VBox mainLayout = new VBox(10, inv, searchLayout, inventoryTable);
        mainLayout.setPadding(new Insets(20));
        mainLayout.setAlignment(Pos.CENTER);

        // Main container
        root = new BorderPane();
        root.setTop(menuBar);
        root.setCenter(mainLayout);

        // Set the scene
        scene = new Scene(root, 800, 600);
    }

    private void styling() {
        // Add styling if needed (CSS or inline styles)
    }

    private void populateTables() {
        inventoryList = InventoryControl.getAll();
        inventoryTable.setItems(FXCollections.observableArrayList(inventoryList));
    }

    private void setTableColumns() {
        idColumn.setCellValueFactory(new PropertyValueFactory<Inventory, String>("id"));
        nameColumn.setCellValueFactory(new PropertyValueFactory<Inventory, String>("name"));
        priceColumn.setCellValueFactory(new PropertyValueFactory<Inventory, Integer>("price"));
        stockColumn.setCellValueFactory(new PropertyValueFactory<Inventory, Integer>("stock"));
    }

    private void action() {
        home.setOnAction(e->{
            Main.showHomePage();
        });
        customer.setOnAction(e->{
            Main.showCustomerPage();
        });
```

```java
private void action() {
    home.setOnAction(e->{
        Main.showHomePage();
    });
    customer.setOnAction(e->{
        Main.showCustomerPage();
    });
    report.setOnAction(e->{
        Main.showReportPage();
    });
    logout.setOnAction(e->{
        Main.showLoginPage();
    });
    searchButton.setOnAction(e -> {
        String search = searchField.getText();
        String searchBy = searchByCB.getValue();
        if(searchBy.equalsIgnoreCase("ItemID")) {
            inventoryList = InventoryControl.searchID(search);

        } else if (searchBy.equalsIgnoreCase("ItemName")) {
            inventoryList = InventoryControl.searchName(search);

        }
        inventoryTable.setItems(FXCollections.observableArrayList(inventoryList));
    });
    refreshButton.setOnAction(e -> {
        populateTables();
    });
}

public Scene getScene() {
    return scene;
}
```

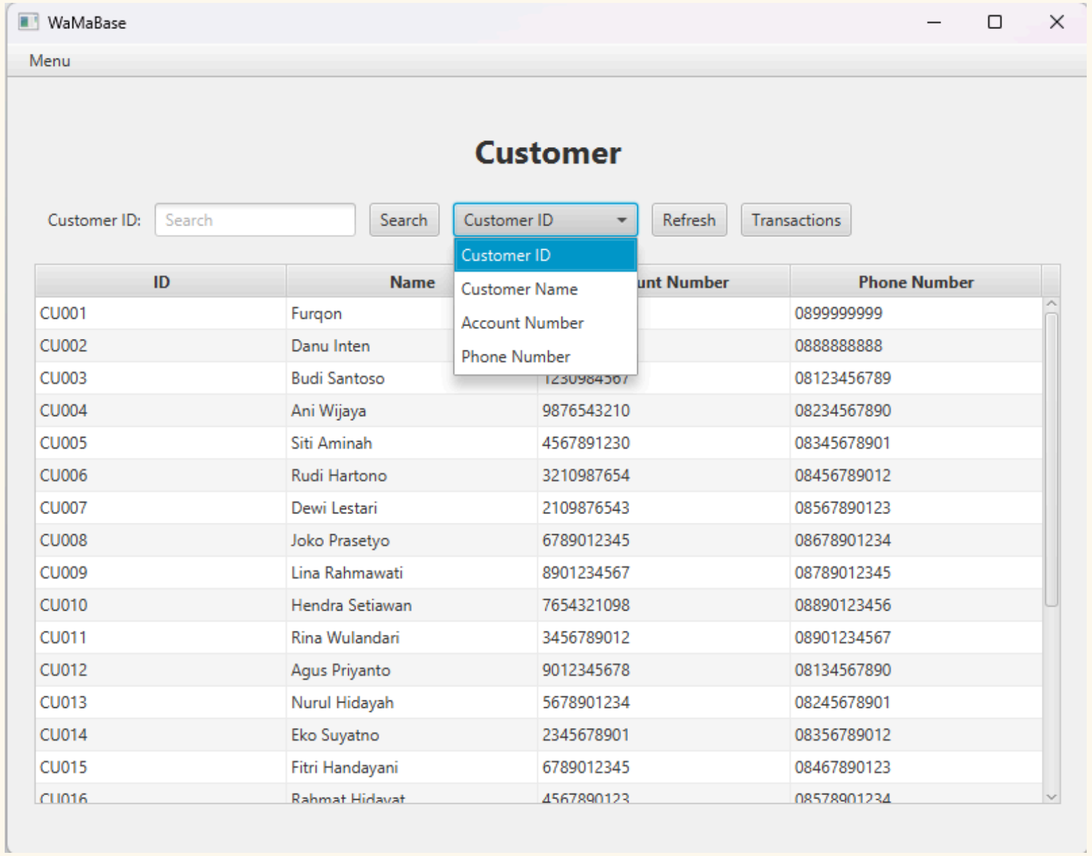
InventoryControl.java

Meng-handle backend dari Inventory Page



```java
package controller;

import java.sql.Connection;

public class InventoryControl {

    public static ObservableList<Inventory> getAll() {
        ObservableList<Inventory> inventoryList = FXCollections.observableArrayList();

        try (Connection conn = DatabaseConnection.getConnection()){
            String query = "SELECT mi.ItemID, ItemName, ItemPrice, StockQuantity "
                    + "FROM msitem mi "
                    + "JOIN inventory i ON mi.ItemID = i.ItemID";
            PreparedStatement ps = conn.prepareStatement(query);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                inventoryList.add(new Inventory(rs.getString("ItemID"), rs.getString("ItemName"), rs.getInt("ItemPrice"), rs.getInt("StockQuantity")));
            }

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return inventoryList;
    }

    public static ObservableList<Inventory> searchID(String search) {
        ObservableList<Inventory> itemList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM msitem mi\r\n"
                    + "JOIN inventory i ON mi.ItemID = i.ItemID\r\n"
                    + "WHERE mi.ItemID REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                itemList.add(new Inventory(rs.getString("ItemID"), rs.getString("ItemName"), rs.getInt("ItemPrice"), rs.getInt("StockQuantity")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return itemList;
    }

    public static ObservableList<Inventory> searchName(String search) {
        ObservableList<Inventory> itemList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM msitem mi\r\n"
                    + "JOIN inventory i ON mi.ItemID = i.ItemID\r\n"
                    + "WHERE ItemName REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                itemList.add(new Inventory(rs.getString("ItemID"), rs.getString("ItemName"), rs.getInt("ItemPrice"), rs.getInt("StockQuantity")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return itemList;
    }
}
```

CUSTOMER


WaMaBase
Menu
Customer
Customer ID:
Search
Search
Customer ID
Refresh
Transactions
Customer ID
Customer Name
Account Number
Phone Number
ID
Name
unt Number
Phone Number
CU001 Furqon 0899999999
CU002 Danu Inten 0888888888
CU003 Budi Santoso 08123456789
CU004 Ani Wijaya 9876543210 08234567890
CU005 Siti Aminah 4567891230 08345678901
CU006 Rudi Hartono 3210987654 08456789012
CU007 Dewi Lestari 2109876543 08567890123
CU008 Joko Prasetyo 6789012345 08678901234
CU009 Lina Rahmawati 8901234567 08789012345
CU010 Hendra Setiawan 7654321098 08890123456
CU011 Rina Wulandari 3456789012 08901234567
CU012 Agus Priyanto 9012345678 08134567890
CU013 Nurul Hidayah 5678901234 08245678901
CU014 Eko Suyatno 2345678901 08356789012
CU015 Fitri Handayani 6789012345 08467890123
CU016 Rahmat Hidayat 4567890123 08578901234

CustomerView.java

Meng-inisialisasi page Customer dan inisialisasi event handling

```java
package scenes;

import controller.CustomerController;

public class CustomerView {

    private static Scene scene;
    private MenuBar menuBar;
    private Menu menuOptions;
    private MenuItem home, inventory, customer, report, logout;
    private TableView<Customer> customerTable;
    private TableColumn<Customer, String> idColumn, nameColumn, accountNumberColumn, phoneNumberColumn;
    private ComboBox<String> searchByCB;
    private TextField searchField;
    private Button searchButton, refreshButton, showTransactionButton;
    private BorderPane root;

    public static Customer selectedCustomer;
    private ObservableList<Customer> customerList = FXCollections.observableArrayList();

    public CustomerView() {
        initialize();
        styling();
        populateTable();
        setTableColumns();
        setTableListener();
        action();
    }

    private void initialize() {
        // Create the MenuBar
        menuBar = new MenuBar();

        // Create Menus and MenuItems
        menuOptions = new Menu("Menu");
        home = new MenuItem("Home");
        inventory = new MenuItem("Inventory");
        customer = new MenuItem("Customer");
        customer.setDisable(true);
        report = new MenuItem("Report");
        logout = new MenuItem("Log Out");

        // Add MenuItems to the Menu
        menuOptions.getItems().addAll(home, inventory, customer, report, new SeparatorMenuItem(), logout);
        menuBar.getMenus().add(menuOptions);

        // Create Customer Table
        customerTable = new TableView<Customer>();
        customerTable.setPlaceholder(new Label("No content in table"));

        idColumn = new TableColumn<Customer, String>("ID");
        nameColumn = new TableColumn<Customer, String>("Name");
        accountNumberColumn = new TableColumn<Customer, String>("Account Number");
        phoneNumberColumn = new TableColumn<Customer, String>("Phone Number");

        customerTable.getColumns().addAll(idColumn, nameColumn, accountNumberColumn, phoneNumberColumn);
        customerTable.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
```

```java
        // Create Search Field and Button
        searchField = new TextField();
        searchField.setPromptText("Search");
        searchButton = new Button("Search");
        refreshButton = new Button("Refresh");
        searchByCB = new ComboBox<String>();
        searchByCB.getItems().addAll("Customer ID", "Customer Name", "Account Number", "Phone Number");
        searchByCB.setValue("Customer ID");
        showTransactionButton = new Button("Transactions");

        // Layout for Search
        HBox searchLayout = new HBox(10, new Label("Customer ID:"), searchField, searchButton, searchByCB, refreshButton, showTransactionButton);
        searchLayout.setAlignment(Pos.CENTER_LEFT);
        searchLayout.setPadding(new Insets(10));

        Label cust = new Label("Customer");
        cust.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

        // Main Layout
        VBox mainLayout = new VBox(10, cust, searchLayout, customerTable);
        mainLayout.setPadding(new Insets(20));
        mainLayout.setAlignment(Pos.CENTER);

        // Main container
        root = new BorderPane();
        root.setTop(menuBar);
        root.setCenter(mainLayout);

        // Set the scene
        scene = new Scene(root, 800, 600);
    }

    private void styling() {
        // Add styling if needed (CSS or inline styles)
    }

    private void populateTable() {
        customerList = CustomerController.getAll();
        customerTable.setItems(FXCollections.observableArrayList(customerList));
    }

    private void setTableColumns() {
        idColumn.setCellValueFactory(new PropertyValueFactory<Customer, String>("id"));
        nameColumn.setCellValueFactory(new PropertyValueFactory<Customer, String>("name"));
        accountNumberColumn.setCellValueFactory(new PropertyValueFactory<Customer, String>("accountNumber"));
        phoneNumberColumn.setCellValueFactory(new PropertyValueFactory<Customer, String>("phoneNumber"));
    }
    private void setTableListener() {
        customerTable.getSelectionModel().selectedItemProperty().addListener((obs, oldValue, selectedCustomer) -> {
            if(selectedCustomer != null) {
                CustomerView.selectedCustomer = selectedCustomer;
            }
        });
    }
```

CustomerView.java

Meng-inisialisasi page Customer dan inisialisasi event handling

```
    private void action() {
        home.setOnAction(e->{
            Main.showHomePage();
        });
        inventory.setOnAction(e->{
            Main.showInventoryPage();
        });
        report.setOnAction(e->{
            Main.showReportPage();
        });
        logout.setOnAction(e->{
            Main.showLoginPage();
        });
        showTransactionButton.setOnAction(e->{
            if(selectedCustomer == null) {
                Main.showAlert(AlertType.ERROR, "Error", "Validation Error", "Please select a customer!");
                return;
            }
            Main.showTransactionPage();
        });
        searchButton.setOnAction(e->{
            String search = searchField.getText();
            String searchBy = searchByCB.getValue();

            if(searchBy.equalsIgnoreCase("Customer ID")) {
                customerList = CustomerController.searchID(search);
            } else if(searchBy.equalsIgnoreCase("Customer Name")) {
                customerList = CustomerController.searchName(search);
            } else if(searchBy.equalsIgnoreCase("Account Number")) {
                customerList = CustomerController.searchAccountNumber(search);
            } else if(searchBy.equalsIgnoreCase("Phone Number")) {
                customerList = CustomerController.searchPhoneNumber(search);
            }
            customerTable.setItems(FXCollections.observableArrayList(customerList));
        });
        refreshButton.setOnAction(e ->{
            populateTable();
        });
    }

    public Scene getScene() {
        return scene;
    }

}
```

CustomerController.java

Meng-handle backend page customer

```java
package controller;

import java.sql.Connection;

public class CustomerController {

    public static ObservableList<Customer> getAll() {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT * FROM mscustomer";
            PreparedStatement ps = conn.prepareStatement(query);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }

    public static ObservableList<Customer> searchID(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE CustomerID REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }

    public static ObservableList<Customer> searchName(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE CustomerName REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();
```

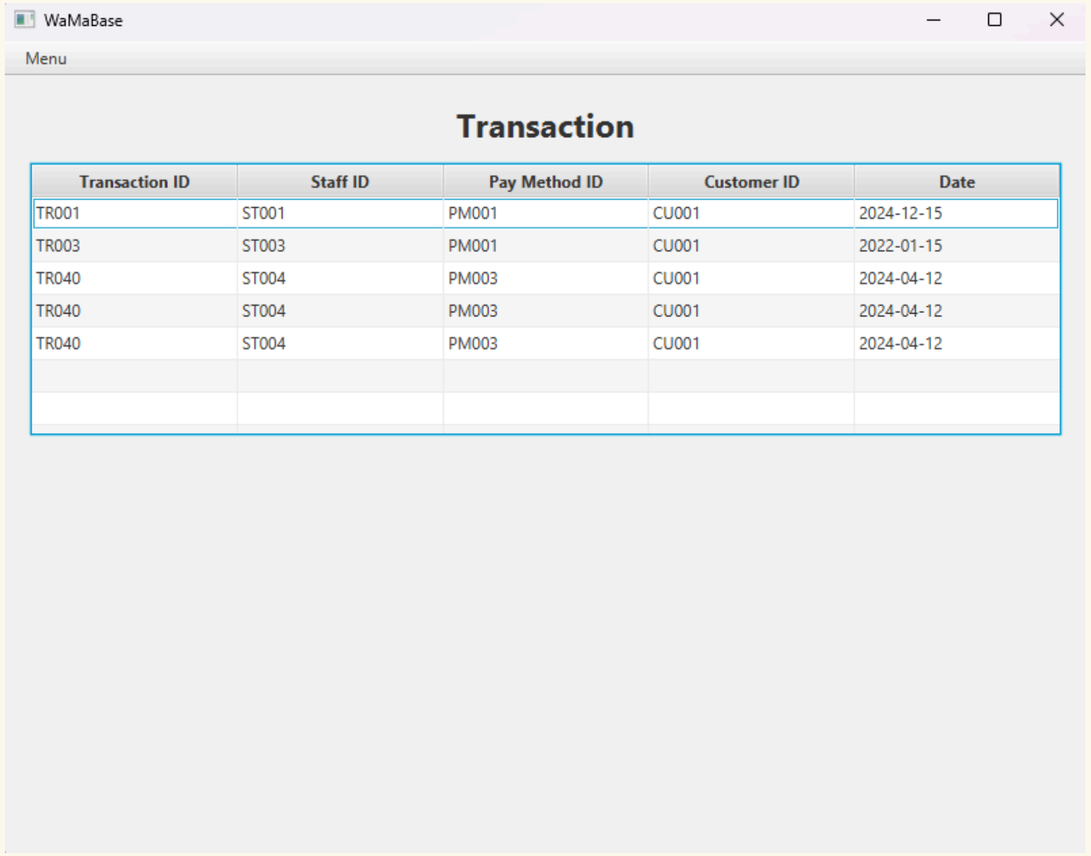
```java
    public static ObservableList<Customer> searchName(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE CustomerName REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }

    public static ObservableList<Customer> searchAccountNumber(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE AccountNumber REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }
```

CustomerController.java

Meng-handle backend page customer

```
    public static ObservableList<Customer> searchPhoneNumber(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE PhoneNumber REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }

}
```

TRANSACTION



TransactionView.java

Meng-inisialisasi page Customer dan inisialisasi event handling

```java
package scenes;

import controller.TransactionController;

public class TransactionView {

    private static Scene scene;
    private MenuBar menuBar;
    private Menu menuOptions;
    private MenuItem back, logout;
    private TableView<Transaction> transactionTable;
    private TableColumn<Transaction, String> transactionIdColumn, staffIdColumn, payIdColumn, customerIdColumn, dateColumn;
    private TableView<TransactionDetail> transactionDetailTable;
    private TableColumn<TransactionDetail, String> itemNameColumn;
    private TableColumn<TransactionDetail, Integer> quantityColumn, totalColumn;
    private TextField transactionIdField;
    private Button searchButton;
    private BorderPane root;
    private HBox detailLayout;
    private Label transactionIdLabel, staffNameLabel, customerNameLabel, dateLabel;

    private Transaction selectedTransaction;
    private ObservableList<Transaction> transactionList = FXCollections.observableArrayList();
    private ObservableList<TransactionDetail> tDetailList = FXCollections.observableArrayList();

    public TransactionView() {
        initialize();
        styling();
        populateTable();
        setTableColumns();
        setTableListener();
        action();
    }
```

```java
    @SuppressWarnings({ "unchecked", "deprecation" })
    private void initialize() {
        // Create the MenuBar
        menuBar = new MenuBar();

        // Create Menus and MenuItems
        menuOptions = new Menu("Menu");
        back = new MenuItem("Back");
        logout = new MenuItem("Log Out");

        // Add MenuItems to the Menu
        menuOptions.getItems().addAll(back, logout);
        menuBar.getMenus().add(menuOptions);

        // Create Transaction Tables
        transactionTable = new TableView<Transaction>();
        transactionTable.setPlaceholder(new Label("No content in table"));
        transactionTable.setPrefHeight(200);

        transactionIdColumn = new TableColumn<Transaction, String>("Transaction ID");
        staffIdColumn = new TableColumn<Transaction, String>("Staff ID");
        payIdColumn = new TableColumn<Transaction, String>("Pay Method ID");
        customerIdColumn = new TableColumn<Transaction, String>("Customer ID");
        dateColumn = new TableColumn<Transaction, String>("Date");

        transactionTable.getColumns().addAll(transactionIdColumn, staffIdColumn, payIdColumn, customerIdColumn, dateColumn);
        transactionTable.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

        transactionDetailTable = new TableView<TransactionDetail>();
        transactionDetailTable.setPlaceholder(new Label("No content in table"));
        transactionDetailTable.setPrefHeight(150);

        itemNameColumn = new TableColumn<TransactionDetail, String>("Item Name");
        quantityColumn = new TableColumn<TransactionDetail, Integer>("Quantity");
        totalColumn = new TableColumn<TransactionDetail, Integer>("Total");

        transactionDetailTable.getColumns().addAll(itemNameColumn, quantityColumn, totalColumn);
        transactionDetailTable.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);

        // Set Transaction Detail Labels
        transactionIdLabel = new Label("Transaction ID  :");
        staffNameLabel = new Label("Staff Name      :");
        customerNameLabel = new Label("Customer Name    :");
        dateLabel = new Label("Date            :");

        // Create Search Field and Button
        transactionIdField = new TextField();
        transactionIdField.setPromptText("Insert TransactionID here");
        searchButton = new Button("Search");

          VBox leftDetailLayout = new VBox(10, transactionIdLabel, staffNameLabel, customerNameLabel, dateLabel);
          leftDetailLayout.setPadding(new Insets(20));
          leftDetailLayout.setAlignment(Pos.TOP_LEFT);

          detailLayout = new HBox(30, leftDetailLayout, transactionDetailTable);
          detailLayout.setAlignment(Pos.TOP_CENTER);
          detailLayout.setPadding(new Insets(40));

          Label trans = new Label("Transaction");
          trans.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

          // Main Layout
          VBox mainLayout = new VBox(10, trans, transactionTable);
          mainLayout.setPadding(new Insets(20));
          mainLayout.setAlignment(Pos.TOP_CENTER);

          // Main container
          root = new BorderPane();
          root.setTop(menuBar);
          root.setCenter(mainLayout);

          // Set the scene
          scene = new Scene(root, 800, 600);
      }
```

```
131    private void populateTable() {
132        transactionList = TransactionController.getAll(CustomerView.selectedCustomer.getId());
133        transactionTable.setItems(FXCollections.observableArrayList(transactionList));
134    }
135
136    private void setTableColumns() {
137        transactionIdColumn.setCellValueFactory(new PropertyValueFactory<Transaction, String>("transactionID"));
138        staffIdColumn.setCellValueFactory(new PropertyValueFactory<Transaction, String>("staffID"));
139        payIdColumn.setCellValueFactory(new PropertyValueFactory<Transaction, String>("payID"));
140        customerIdColumn.setCellValueFactory(new PropertyValueFactory<Transaction, String>("customerID"));
141        dateColumn.setCellValueFactory(new PropertyValueFactory<Transaction, String>("date"));
142
143        itemNameColumn.setCellValueFactory(new PropertyValueFactory<TransactionDetail, String>("ItemName"));
144        quantityColumn.setCellValueFactory(new PropertyValueFactory<TransactionDetail, Integer>("quantity"));
145        totalColumn.setCellValueFactory(new PropertyValueFactory<TransactionDetail, Integer>("total"));
146    }
147
148    private void setTableListener() {
149        transactionTable.getSelectionModel().selectedItemProperty().addListener((obs, oldValue, selectedTransaction) -> {
150            this.selectedTransaction = selectedTransaction;
151            tDetailList = TransactionController.getDetail(selectedTransaction.getTransactionID());
152            transactionDetailTable.setItems(FXCollections.observableArrayList(tDetailList));
153            transactionIdLabel.setText("Transaction ID  :" + selectedTransaction.getTransactionID());
154            staffNameLabel.setText("Staff Name      :" + TransactionController.getStaffName(selectedTransaction.getTransactionID()));
155            customerNameLabel.setText("Customer Name    :" + TransactionController.getCustomerName(selectedTransaction.getTransactionID()));
156            dateLabel.setText("Date             :" + selectedTransaction.getDate());
157            root.setBottom(detailLayout);
158        });
159    }
160
161    private void action() {
162        back.setOnAction(e->{
163            Main.showCustomerPage();
164        });
165        logout.setOnAction(e->{
166            Main.showLoginPage();
167        });
168    }
169
170    public Scene getScene() {
171        return scene;
172    }
173 }
```

TransactionController.java

Meng-handle backend page customer

```java
package controller;

import java.sql.Connection;

public class CustomerController {

    public static ObservableList<Customer> getAll() {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT * FROM mscustomer";
            PreparedStatement ps = conn.prepareStatement(query);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }

    public static ObservableList<Customer> searchID(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE CustomerID REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }

    public static ObservableList<Customer> searchName(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE CustomerName REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();
```

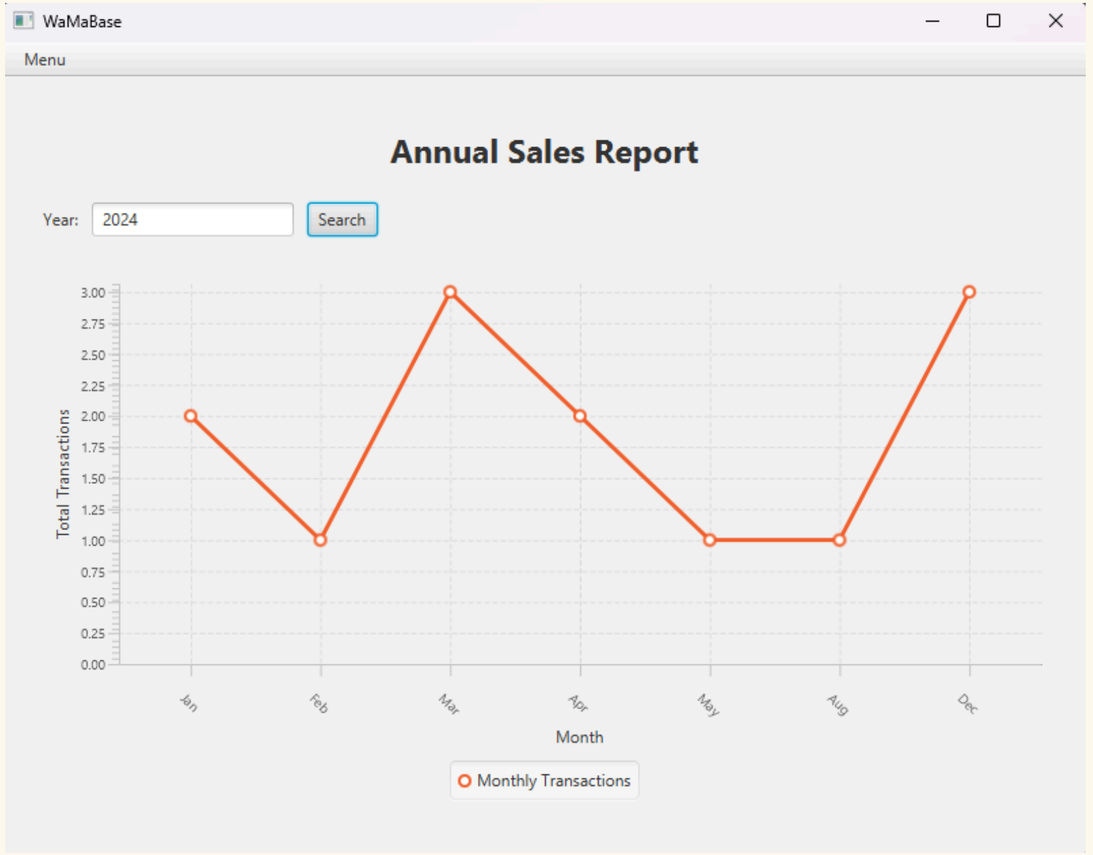
```java
public static ObservableList<Customer> searchName(String search) {
    ObservableList<Customer> customerList = FXCollections.observableArrayList();
    try (Connection conn = DatabaseConnection.getConnection()) {
        String query = "SELECT *\r\n"
                + "FROM mscustomer\r\n"
                + "WHERE CustomerName REGEXP ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, search);
        ResultSet rs = ps.executeQuery();

        while(rs.next()) {
            customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                    rs.getString("PhoneNumber")));
        }

    } catch (SQLException e) {

        e.printStackTrace();

    }
    return customerList;
}

public static ObservableList<Customer> searchAccountNumber(String search) {
    ObservableList<Customer> customerList = FXCollections.observableArrayList();
    try (Connection conn = DatabaseConnection.getConnection()) {
        String query = "SELECT *\r\n"
                + "FROM mscustomer\r\n"
                + "WHERE AccountNumber REGEXP ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, search);
        ResultSet rs = ps.executeQuery();

        while(rs.next()) {
            customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                    rs.getString("PhoneNumber")));
        }

    } catch (SQLException e) {

        e.printStackTrace();

    }
    return customerList;
}

public static ObservableList<Customer> searchPhoneNumber(String search) {
    ObservableList<Customer> customerList = FXCollections.observableArrayList();
    try (Connection conn = DatabaseConnection.getConnection()) {
        String query = "SELECT *\r\n"
                + "FROM mscustomer\r\n"
                + "WHERE PhoneNumber REGEXP ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, search);
        ResultSet rs = ps.executeQuery();

        while(rs.next()) {
            customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
```

TransactionController.java

Meng-handle backend page customer

```
    public static ObservableList<Customer> searchPhoneNumber(String search) {
        ObservableList<Customer> customerList = FXCollections.observableArrayList();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String query = "SELECT *\r\n"
                    + "FROM mscustomer\r\n"
                    + "WHERE PhoneNumber REGEXP ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, search);
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                customerList.add(new Customer(rs.getString("CustomerID"), rs.getString("CustomerName"), rs.getString("AccountNumber"),
                        rs.getString("PhoneNumber")));
            }

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return customerList;
    }

}
```

REPORT


WaMaBase
Menu
Annual Sales Report
Year:
2024
Search
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Total Transactions
Jan
Feb
Mar
Apr
May
Aug
Dec
Month
Monthly Transactions

Report.java

Meng-handle inisialisasi view dari page dan juga backend dari Inventory Page

```java
package scenes;

import javafx.collections.FXCollections;

public class ReportView {

    private static Scene scene;
    private MenuBar menuBar;
    private Menu menuOptions;
    private MenuItem home, inventory, customer, report, logout;
    private TextField yearField;
    private Button searchButton;
    private LineChart<String, Number> salesChart;
    private BorderPane root;

    public ReportView() {
        initialize();
        styling();
        action();
    }

    private void initialize() {
        // Create the MenuBar
        menuBar = new MenuBar();

        // Create Menus and MenuItems
        menuOptions = new Menu("Menu");
        home = new MenuItem("Home");
        inventory = new MenuItem("Inventory");
        customer = new MenuItem("Customer");
        report = new MenuItem("Report");
        report.setDisable(true);
        logout = new MenuItem("Log Out");

        // Add MenuItems to the Menu
        menuOptions.getItems().addAll(home, inventory, customer, report, new SeparatorMenuItem(), logout);
        menuBar.getMenus().add(menuOptions);

        // Create Search Controls
        yearField = new TextField();
        yearField.setPromptText("Enter Year");

        searchButton = new Button("Search");

        // Layout for Search
        HBox searchLayout = new HBox(10, new Label("Year:"), yearField, searchButton);
        searchLayout.setAlignment(Pos.CENTER_LEFT);
        searchLayout.setPadding(new Insets(10));

        // Create Sales Chart
        CategoryAxis xAxis = new CategoryAxis();
        xAxis.setLabel("Month");
        xAxis.setTickLabelRotation(45); // Rotasi 45 derajat untuk mengurangi tumpang tindih
        xAxis.setTickLabelGap(10);

        NumberAxis yAxis = new NumberAxis();
        yAxis.setLabel("Total Transactions");

        salesChart = new LineChart<>(xAxis, yAxis);

        Label titleLabel = new Label("Annual Sales Report");
        titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

        // Main Layout
        VBox mainLayout = new VBox(10, titleLabel, searchLayout, salesChart);
        mainLayout.setPadding(new Insets(20));
        mainLayout.setAlignment(Pos.CENTER);

        // Main container
        root = new BorderPane();
        root.setTop(menuBar);
        root.setCenter(mainLayout);

        // Set the scene
        scene = new Scene(root, 800, 600);
    }

    private void styling() {
        // Add styling if needed (CSS or inline styles)
    }

    private void action() {
        home.setOnAction(e -> {
            Main.showHomePage();
        });
        inventory.setOnAction(e -> {
            Main.showInventoryPage();
        });
        customer.setOnAction(e -> {
            Main.showCustomerPage();
        });
        logout.setOnAction(e -> {
            Main.showLoginPage();
        });

        searchButton.setOnAction(e -> {
            String year = yearField.getText();
            if (!year.isEmpty()) {
                updateChart(year);
            } else {
                System.out.println("Please enter a valid year.");
            }
        });
    }

    private void updateChart(String year) {
        ObservableList<XYChart.Data<String, Number>> chartData = fetchTransactionData(year);
        XYChart.Series<String, Number> series = new XYChart.Series<>();

        if (chartData.isEmpty()) {
            System.out.println("No data available for the selected year.");
            showAlert("Error", "Tidak ada data untuk tahun ini");
            return;
        }

        series.setName("Monthly Transactions");

        series.getData().addAll(chartData);

        salesChart.getData().clear();
        salesChart.getData().add(series);

        // Force layout refresh
        salesChart.layout();
    }


    private ObservableList<XYChart.Data<String, Number>> fetchTransactionData(String year) {
        ObservableList<XYChart.Data<String, Number>> data = FXCollections.observableArrayList();

        String query = "SELECT MONTH(TransactionDate) as month, COUNT(TransactionID) as count " +
                       "FROM transactionHeader " +
                       "WHERE YEAR(TransactionDate) = ? " +
                       "GROUP BY MONTH(TransactionDate)";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setString(1, year);

            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    int month = rs.getInt("month");
                    int count = rs.getInt("count");
                    String monthName = getMonthName(month); // Convert month number to name
                    data.add(new XYChart.Data<>(monthName, count));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return data;
    }

    private String getMonthName(int month) {
        String[] months = {
            "Jan", "Feb", "Mar", "Apr", "May", "Jun",
            "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
        };
        return months[month - 1];
    }
```