

# Flora Artistry: JavaFX Project Documentation

This document explains the structure, design patterns, and components of the **Flora Artistry** project. The project is organized into several packages, each serving a distinct purpose, and adheres to clean code principles like **MVC**, **DAO**, and **Singleton**.

---

## Project Structure

The project is organized into the following packages:

1. **controller**: Contains classes for managing user interactions and updating the view/model.
  2. **dao**: Encapsulates database operations using the **DAO pattern**.
  3. **database**: Manages the connection to the database with a singleton class.
  4. **main**: Contains the entry point of the application.
  5. **model**: Defines the data structure of entities.
  6. **resources**: Stores resources.
  7. **view**: Contains the UI components and logic for rendering pages.
  8. **view.partials**: Contains reusable components like navigation menus and custom alerts.
- 

## Package Details

### 1. controller

Manages interactions between the view and the model.

- **Abstract Class: Controller**
  - A generic class that holds a reference to a specific DAO and the MainController.
  - Provides shared functionality for all controllers.
- **Concrete Controllers:**

- **CartController**: Manages cart-related actions (e.g., adding/removing items).
  - **FlowerController**: Handles flower-related operations like fetching flower data.
  - **LoginController**: Manages user authentication logic.
  - **RegisterController**: Handles new user registration.
  - **PageController**: Responsible for navigating between views (e.g., showLoginView).
  - **TransactionController**: Manages transactions (header and detail operations).
  - **MainController**:
    - Singleton class that holds all other controllers as single instances.
    - Provides **getter methods** to access specific controllers.
- 

## 2. dao

Encapsulates database operations using the **DAO Design Pattern** for abstraction and reusability.

- **Abstract Class: DAO**
    - Superclass that provides shared logic for DAOs.
    - Contains a reference to Connect instance for connecting to the database.
  - **Concrete DAOs:**
    - **CartDAO**: Handles CRUD operations for the Cart model.
    - **FlowerDAO**: Manages Flower data.
    - **TransactionDAO**: Operates on TransactionHeader and TransactionDetail models.
    - **UserDAO**: Manages user data for authentication and registration.
- 

## 3. database

Handles database connectivity.

- **Class: Connect**
    - Implements the **Singleton Design Pattern** to ensure a single database connection.
-

## 4. main

Contains the application entry point.

- **Class: Main**
    - Starts the application and initializes the MainController.
    - Displays the LoginView using PageController.
- 

## 5. model

Defines the application's data entities.

- **Classes:**
    - Cart → userId, flowerId, quantity.
    - User → userId, username, email, password, address, phoneNumber, role.
    - Flower → flowerId, name, type, price.
    - TransactionHeader → transactionId, userId.
    - TransactionDetail → transactionId, flowerId, quantity.
    - Role → Enum for user roles (CUSTOMER, ADMIN).
- 

## 6. resources

Contains external resources for the application.

- **File: styles.css**
    - Used for styling the JavaFX UI components.
- 

## 7. view

Defines the UI structure of the application.

- **Abstract Class: View**
  - Superclass for all views.

Holds the MainController and defines the abstract method to get the Pane.

- **Concrete Views:**

- BuyFlowerView → Displays flowers available for purchase.

The screenshot displays the BuyFlowerView application. At the top, a dark blue header bar contains the word "Page". Below this, a light gray section titled "Product List" displays a welcome message: "Welcome, Kennedy". Underneath is a table with three columns: "Name", "Type", and "Price". The table lists several flowers, with "Daffodil" highlighted in blue. Below the table, the details for the selected "Daffodil" are shown: "Name: Daffodil", "Type: Bulb", and "Price: 1100". A blue "Add to Cart" button is positioned below these details. At the bottom of the screen, an "Add to Cart" popup is visible. This popup has a dark blue header and contains a label "Name: Tulip", a "Quantity:" label, a text input field with the value "3", and two buttons: "Add to Cart" and "Cancel".

Name	Type	Price
Tulip	Bulb	1500
Lily	Perennial	1200
Daffodil	Bulb	1100
Sunflower	Annual	1300
Orchid	Epiphyte	2000
Lavender	Perennial	1400

Name: Daffodil  
Type: Bulb  
Price: 1100

Add to Cart

Add to Cart

Name: Tulip  
Quantity:  
3  
Add to Cart  
Cancel

- Label: product list, greeting, product detail, product name, quantity.
  - TableView: product table.
  - Spinner: quantity.
  - Button: add to cart, add to cart popup, cancel.
  - Window (jfxtras): add to cart popup
- CartView → Shows the cart and allows management of items.

Page

Your Cart List

Name	Type	Price	Quantity	Total
Tulip	Bulb	1500	1	1500
Sunflower	Annual	1300	3	3900

Name: Tulip  
 Type: Bulb  
 Price: 1500  
 Subtotal: 1500  
**Total: 5400**

Checkout

Checkout All Items

- Label: cart, item detail, total.
  - TableView: cart table.
  - Button: checkout, checkout all items.
- LoginView → Provides a login form.

Page

Login

Email

Password

Login

- Label: login, email, password.
  - TextField: email.
  - PasswordField: password.
  - Button: login.
- RegisterView → Contains a registration form for new users.

**Register**

Email

Username

Password

Confirm Password

Address

Phone Number

Role

☐ I agree to the terms and conditions

**Register**

- Label: register, email, username, password, confirm password, address, phone number, role.
  - TextField: email, username, phone number.
  - PasswordField: password, confirm password.
  - TextArea: address.
  - ComboBox: role.
  - CheckBox: terms and conditions.
  - Button: register.
- ManageFlowerView → Allows admins to add or manage flowers.

**Account****Flower List**

Welcome, Admin

ID	Name	Type	Price
FL002	Tulip	Bulb	1500
FL003	Lily	Perennial	1200
FL004	Daffodil	Bulb	1100
FL005	Sunflower	Annual	1300
FL006	Orchid	Epiphyte	2000
FL008	Lavender	Perennial	1400

Name

Type

Price

**Add****Cancel**

Account

Flower List

Welcome, Admin

ID	Name	Type	Price
FL002	Tulip	Bulb	1500
FL003	Lily	Perennial	1200
FL004	Daffodil	Bulb	1100
FL005	Sunflower	Annual	1300
FL006	Orchid	Epiphyte	2000
FL008	Lavender	Perennial	1400

Name

Lily

Type

Perennial

Price

1200

Update

Delete

Cancel

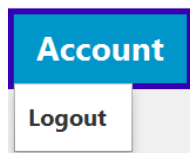
- Label: flower list, greeting, name, type, price.
- TableView: flower table.
- TextField: flower name, flower type, flower price.
- Button: add, update, delete, cancel.

## 8. view.partials

Reusable UI components and utility classes.

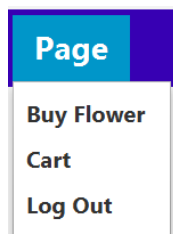
- **Classes:**

- **AdminNavigation:**



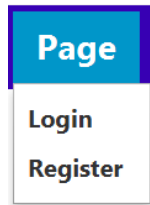
- Navigation menu for admins.
- Options: Logout.

- **CustomerNavigation:**



- Navigation menu for customers.
- Options: Buy Flowers, Cart, Logout.

- **AuthNavigation:**



- Navigation for the login and registration page (switch between them).
  - **CustomAlert:**
    - Utility class for creating JavaFX alert dialogs without instantiating new objects repeatedly.
  - **Abstract Class: Navigation**
    - Superclass for all navigation components.
    - Holds the `MainController` and defines the abstract method to get the `MenuBar`.
- 

## Architecture and Design Patterns Used

1. **MVC:** Separates the application into Model, View, and Controller layers for modularity and scalability.
  2. **DAO:** Encapsulates database operations, making them reusable and maintainable.
  3. **Singleton:** Ensures a single instance of the database connection using the `Connect` class.
- 

## Conclusion

The **Flora Artistry** project is designed for clarity, modularity, and maintainability. By leveraging **MVC**, **DAO**, and **Singleton**, the project ensures separation of concerns and a clean codebase. The use of partial views, abstract superclasses, and dynamic navigation menus enhances reusability and flexibility across the application.