

Ecouture

Team #1

Kate Paduganao, Jason Le, Gabriela Alvarez

# **Software Design Specification & Project Delivery Report**

**Version: (1)**

**Date: (12/11/2024)**

## Table of Contents

1 Introduction.....	4
1.1 Goals and objectives.....	4
1.2 Statement of system scope.....	4
2 Architectural design.....	6
2.1 System Architecture.....	6
High-Level Subsystems and Their Responsibilities.....	7
Subsystem Collaboration.....	7
2.2 Design Rational.....	8
3 Key Functionality design.....	9
3.1 Function 1: Closet.....	9
3.1.1 Closet Use Cases.....	9
3.1.2 Processing sequence for Closet.....	10
3.1.3 Structural Design for Closet.....	11
3.1.4 Key Activities.....	12
3.1.5 Software Interface to other components.....	12
3.2 [Function 2: Dashboard].....	13
3.2.1 [Dashboard] Use Cases.....	13
3.2.2 Processing sequence for [Dashboard].....	14
3.2.3 Structural Design for [Dashboard].....	15
3.2.4 Key Activities.....	16
3.2.5 Software Interface to other components.....	16
3.3 [Function 3: Forum].....	17
3.3.1 [Forum] Use Cases.....	17
3.3.2 Processing sequence for [Forum].....	18
3.3.3 Structural Design for [Forum].....	19
3.3.4 Key Activities.....	20
3.3.5 Software Interface to other components.....	20
3.4 [Function 4: Brand Review].....	21
3.4.1 [Brand Review] Use Cases.....	21
3.4.2 Processing sequence for [Brand Review].....	22
3.4.3 Structural Design for [Brand Review].....	23
3.4.4 Key Activities.....	24
3.4.5 Software Interface to other components.....	24
4 User interface design.....	25
4.1 Interface design rules.....	25
4.2 Description of the user interface.....	25
4.2.1 User Management: Login and Signup.....	25

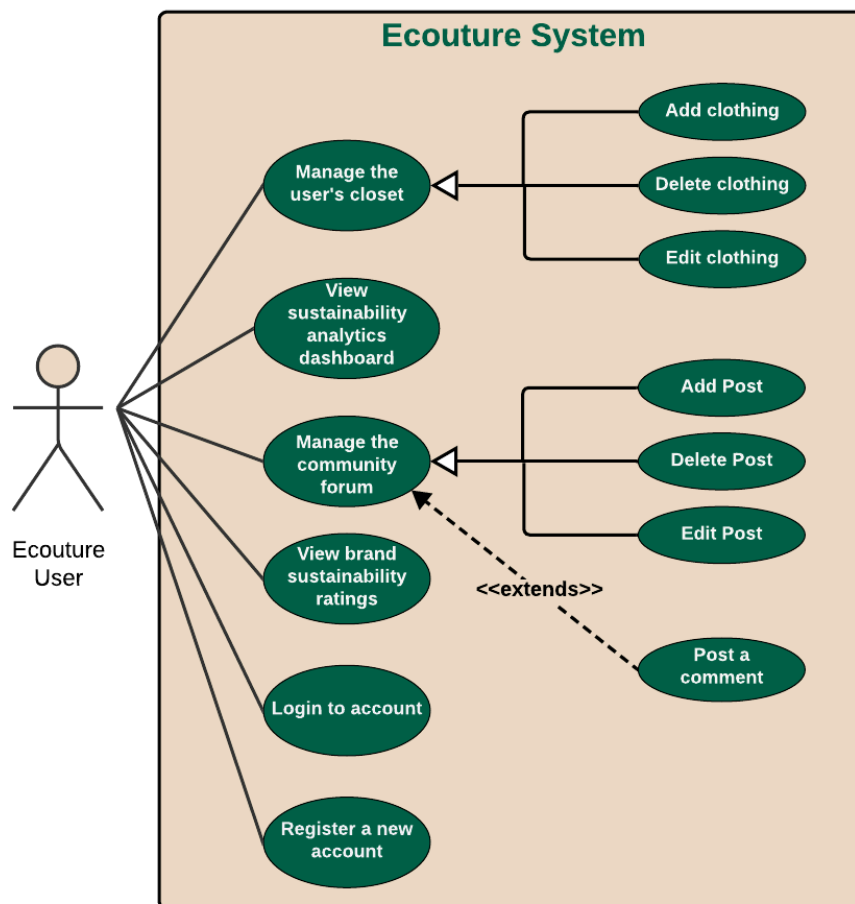
4.2.1 Start Page.....	27
4.2.2 Forum Page.....	28
4.2.3 Closet Page.....	29
4.2.4 Dashboard Page.....	30
4.2.5 Brand Review Page.....	31
5 Restrictions, Limitations, and Constraints.....	33
6 Testing Issues (SLO #2.v).....	33
6.1 Types of tests.....	33
6.2 List of Test Cases.....	34
6.3 Test Coverage.....	35
7 Appendices.....	37
7.1 Packaging and installation issues.....	37
7.2 User Manual.....	37
7.3 Open Issues.....	40
7.4 Lessons Learned.....	40
7.4.1 Project Management & Task Allocations (SLO #2.i).....	41
7.4.2 Implementation (SLO #2.iv).....	41
7.4.3 Design Patterns.....	43
7.4.4 Team Communications.....	43
7.4.4 Technologies Practiced (SLO #7).....	44
7.4.5 Desirable Changes.....	44
7.4.6 Challenges Faced.....	44

# 1 Introduction

## 1.1 Goals and objectives

The goal of our project is to educate people about sustainability, with a focus on sustainable fashion. Our objective is to provide users with insights into the environmental impact of their closet, along with resources and features to help them make more informed choices. Through a review page that assesses popular brands and a dashboard displaying personalized statistics on the sustainability of their closet, our project empowers users to better understand and reduce their fashion footprint.

## 1.2 Statement of system scope



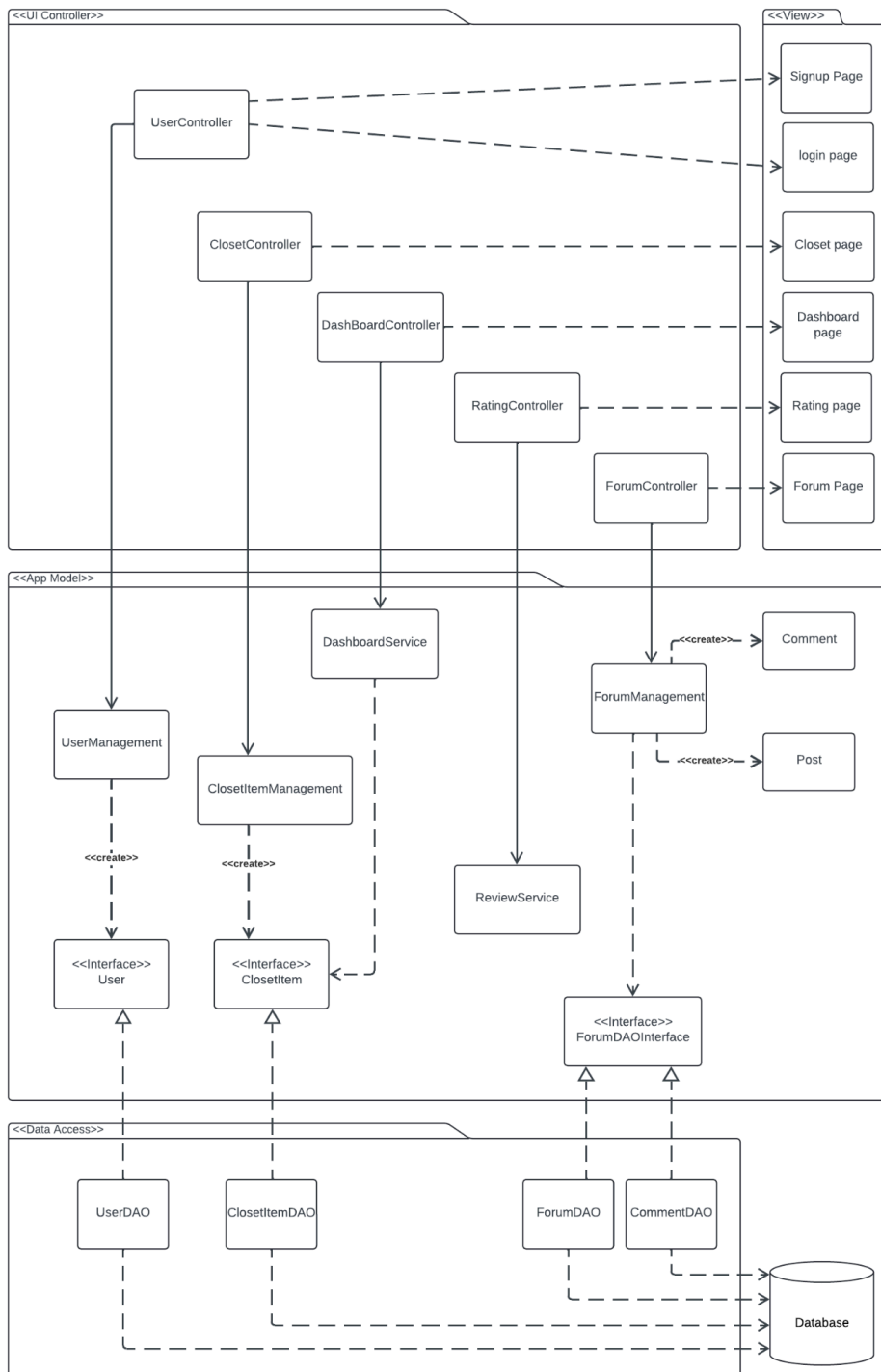
1. **Manage the User's Closet:** Users can manage their wardrobe by adding, deleting, and editing clothing items. It encompasses three sub-functionalities:
  - **Add Clothing:** Users can add new clothing items to their closet database.
  - **Delete Clothing:** Users can remove clothing items they no longer want in their closet.
  - **Edit Clothing:** Users can update information about existing clothing items.
2. **View Sustainability Analytics Dashboard:** Users can access a dashboard displaying insights into their closet sustainability.
3. **Manage the Community Forum:** Users can engage with others by sharing posts in the community forum. It encompasses three sub-functionalities:
  - **Add Post:** Users can add new forum posts to the community forum.
  - **Delete Post:** Users can remove clothing items they no longer want to share.
  - **Edit Post:** Users can update their own existing forum posts.

This feature also lets users:

  - **Post a Comment:** Users can extend their participation by commenting on posts in the forum.
4. **View Brand Sustainability Ratings:** Users can look up information on sustainability practices and ratings of various fashion brands, helping them make informed choices.
5. **Login to Account:** Users can securely access their personalized accounts to utilize the platform's features.
6. **Register a New Account:** Users can create a new account to access the platform's features.

## 2 Architectural design

### 2.1 System Architecture



## High-Level Subsystems and Their Responsibilities

1. **UI Controller Subsystem:**
  - **Role:** Handles user interactions and bridges between the user and the core system functionalities.
2. **App Model Subsystem:**
  - **Role:** Implements the core logic and services for managing system data and functionality.
3. **Data Access Subsystem:**
  - **Role:** Interfaces directly with the database to store, retrieve, and update data.
4. **Database:**
  - **Role:** Serves as the central repository to store the application's data in the system.

## Subsystem Collaboration

1. **User Interaction:**
  - The UI Controller Subsystem captures user inputs (e.g., login, closet updates) and communicates with the corresponding components in the App Model Subsystem to process the requests.
  - The processed data is fetched or updated via the Data Access Subsystem, which interacts directly with the Database.
2. **Closet Management:**
  - The ClosetController interacts with the ClosetItemManagement component to update closet data.
  - Updates are reflected in the ClosetItemDAO, ensuring data persistence.
3. **Sustainability Analytics:**
  - The DashboardController communicates with DashboardService to generate analytics.
4. **Community Forum:**
  - The ForumController interacts with ForumManagement for managing posts and comments.
  - Data related to forum activities is stored and retrieved through ForumDAO and CommentDAO.
5. **Brand Ratings:**
  - The RatingController collaborates with ReviewService to retrieve brand sustainability ratings.

## 2.2 Design Rational

We decided to choose a Layered architectural pattern because it provided us the ability to establish a clear separation of concerns, which aligns perfectly with the needs of our Ecouture project. This architecture allows modularity by dividing the system into layers: the UI and Controller Layers provide a user-friendly interface for sustainable fashion enthusiasts; the Application Model Layer processes the logic needed for most of the main features of the program; and the Data Access Layer helps retrieve information needed for the program from the database. Issues such as maintainability, scalability, and reusability were important to consider for our decision-making process, and this architecture's structured approach minimizes complexity and promotes easy integration of the application's features.

Other architectures were considered, such as the client-server model, which would simplify user and system interactions, but because our project was currently run on a local database, it seemed too complex for an application like Ecouture. We didn't use the peer-to-peer architecture because it lacks centralized control, which is important for Ecouture with its need for consistent data management across features like the forum and closet analytics. The repository model, while effective for centralized data management, would limit the flexibility required for integrating diverse components like the dashboard and forum. Ultimately, the Layered architectural pattern was chosen as it balances these trade-offs, supporting both the technical requirements and the user experience goals of Ecouture.



### 3 Key Functionality design

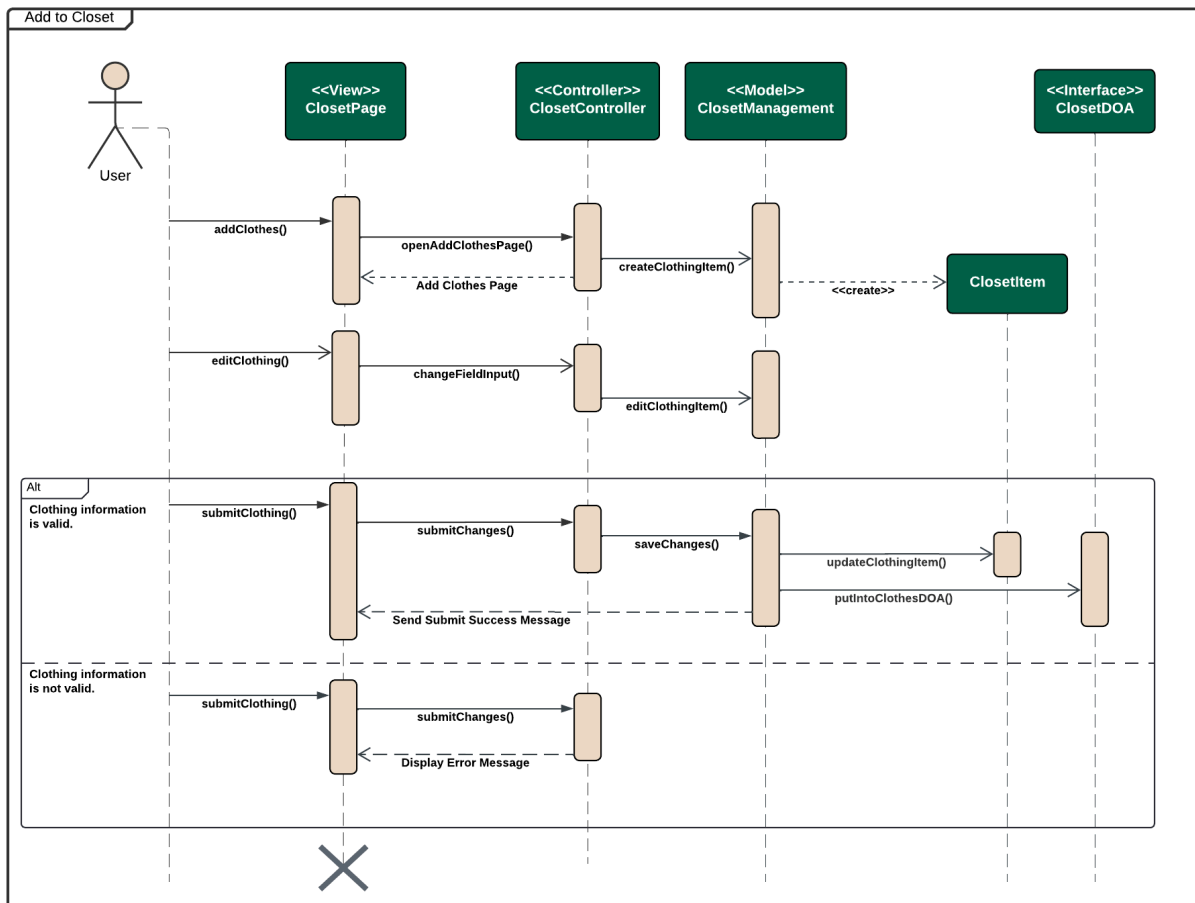
#### 3.1 Function 1: Closet

##### 3.1.1 Closet Use Cases

Use Case	Managing the user's closet
Goal in Context	To allow users to manage items from their closet.
Scope	The Ecouture's Closet system
Level	Primary Task
Primary Actor	Ecouture user
Preconditions	The user is logged in to their account
Minimal Guarantee	Clothing data is not saved.
Success Guarantee	Clothing data has been saved and updated in the closet.
Trigger	User presses the closet icon on the navigation bar
Success Scenario	Action Step
1	user may click "+" button on top right corner to add new clothing items
2	user will be prompted to add details about specific items
3	user may remove existing clothing by pressing the trash icon
4	user may modify existing items by pressing the pencil icon
5	User may save changes using the save button.
6	The system will update the closet accordingly
Extension Step	Branching Action
5a	Upon the user not wanting to save changes
	a1: next to the save button, the user can use the cancel button to

	disregard changes.
	a2: Clothing data is not saved.

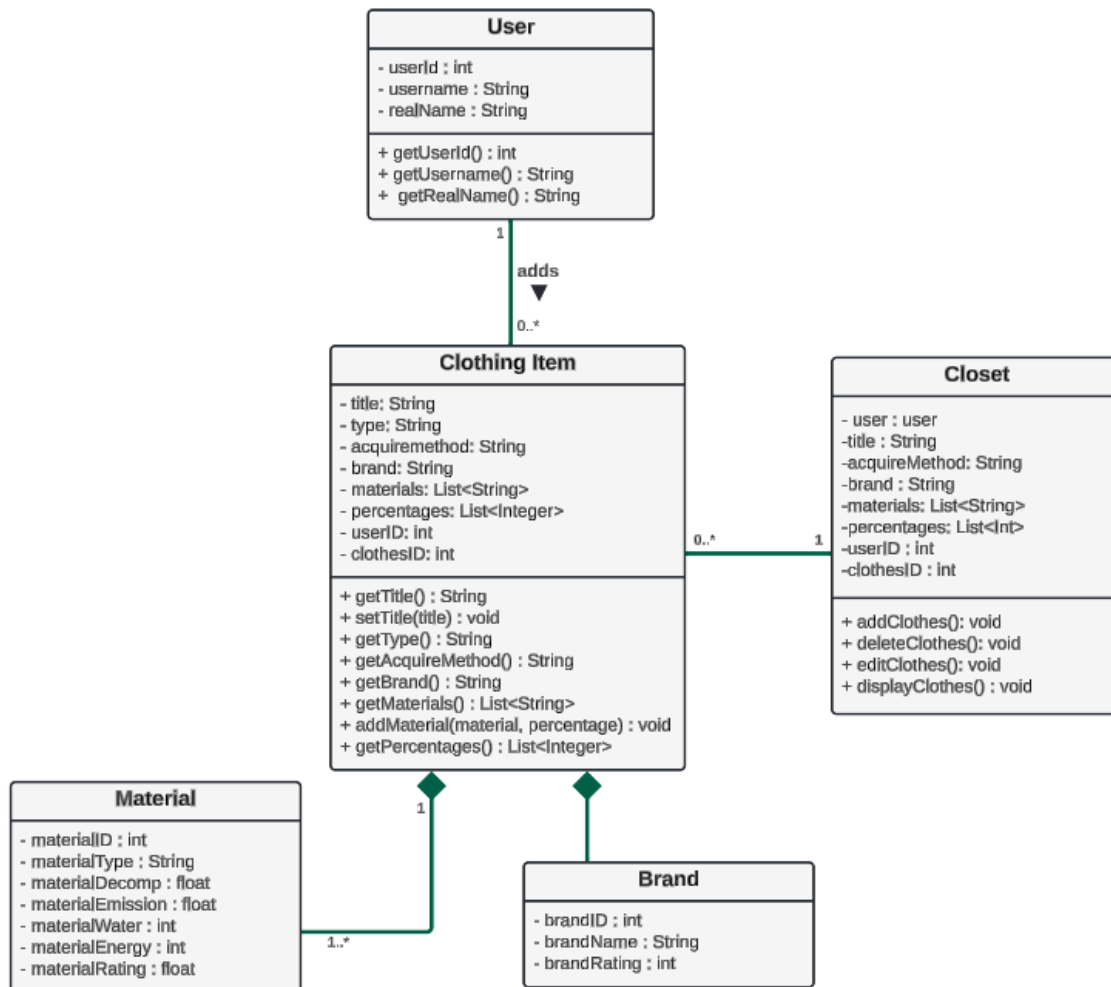
### 3.1.2 Processing sequence for Closet



This sequence diagram illustrates the "Add to Closet" functionality in the Ecouture application. It involves the interaction between the ClosetView (View), ClosetController (Controller), ClosetManagement (Model), and the ClosetDOA (Data Access Object) components. When a user initiates the process to add a clothing item, the ClosetPage sends the input data to the ClosetController, which validates the request and forwards it to the ClosetManagement module. The ClosetManagement class creates or updates a ClosetItem object and interacts with the

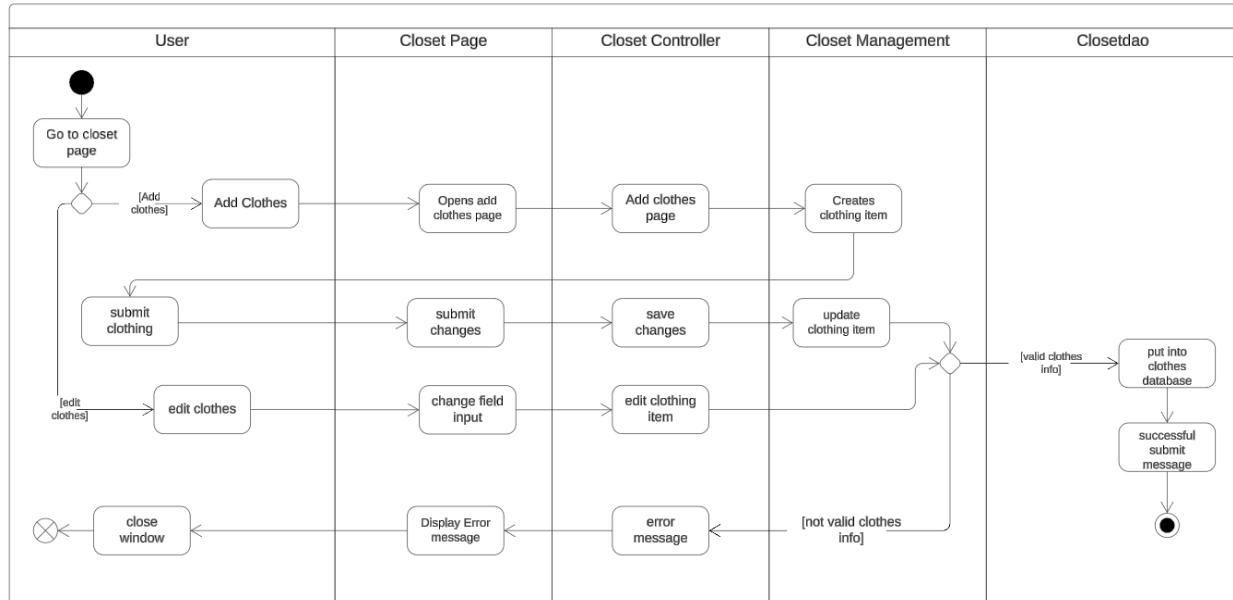
ClosetDOA to persist the data into the database. If the inputted information is not valid, the program will display an error message.

### 3.1.3 Structural Design for Closet



In this uml diagram of our Closet the User class and Closet are associated with the Clothing item class. That is because both classes interact with the clothing item class. The user is able to add clothing items using the Clothing item class and the Closet uses what's been added by the user from the Clothing Item class. Material and Brand show a filled diamond representing composition. That is because material and Brand could not exist, in the case of our program, without Clothing Item.

### 3.1.4 Key Activities



This activity diagram gives an overview of how a user interacts with the closet page in our program, whether they are adding new clothes or editing previous submitted ones. Similar to the sequence diagram, it will involve the closet page(view), closet controller(controller), and closet management(model), as well as querying the database. It does not go into the details about how the system interacts with objects and class as the sequence diagram, however, it shows more of the flow of the system.

### 3.1.5 Software Interface to other components

The Closet module interfaces with various components within the Ecouture system to ensure efficient management of user's clothing item data. The ClosetView provides the user interface for interacting with the closet so that it can execute the closet logic through ClosetControllers and ClosetManagement. The ClosetDOA is an interface that connects to a database, enabling the system to retrieve clothing data.

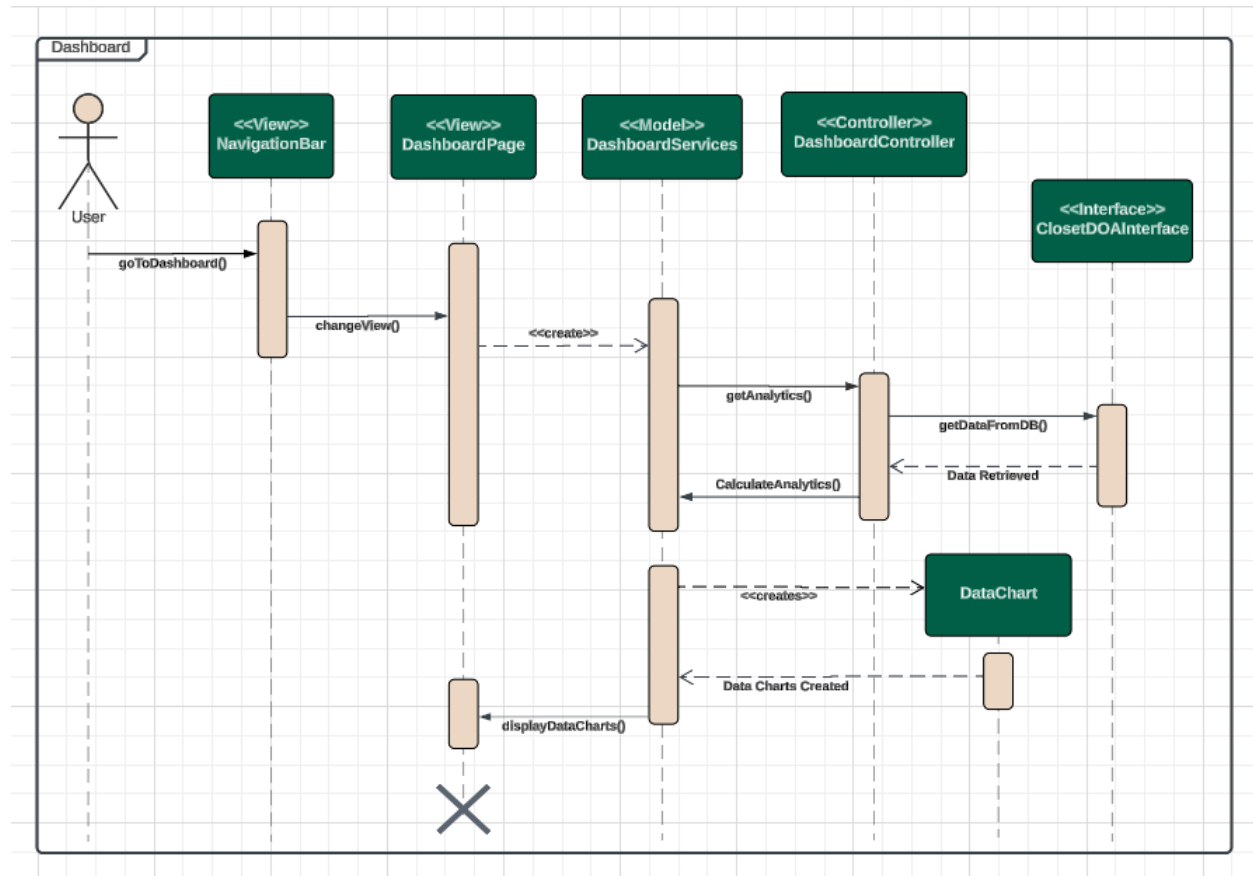
## 3.2 [Function 2: Dashboard]

### 3.2.1 [Dashboard] Use Cases

Use Case	View Dashboard
Goal in Context	A user views their sustainability analytics
Scope	The Ecouture's Analytics System
Level	Primary Task
Primary Actor	Ecouture user
Preconditions	The user is logged in to their account
Minimal Guarantee	An error message is displayed if the analytics cannot be retrieved.
Success Guarantee	The user views multiple sustainability scores displayed on the dashboard.
Trigger	User presses the analytics icon on the navigation bar
Success Scenario	Action Step
1	The system will calculate the sustainability of the materials based on the clothing material in the user's closet.
2	The system will calculate the sustainability of the brands based on the brands that the user buys from in the user's closet.
3	The system will calculate the sustainability of how the user acquires items in their closet.
4	The system will calculate the overall sustainability of all factors for one sustainability score
5	The system will display the result of the material sustainability analytics
6	The system will display the result of the brand sustainability analytics
7	The system will display the result of the acquisition sustainability analytics
8	The system will display the overall sustainability score
9	The user views the analytics and scores.
Extension Step	Branching Action

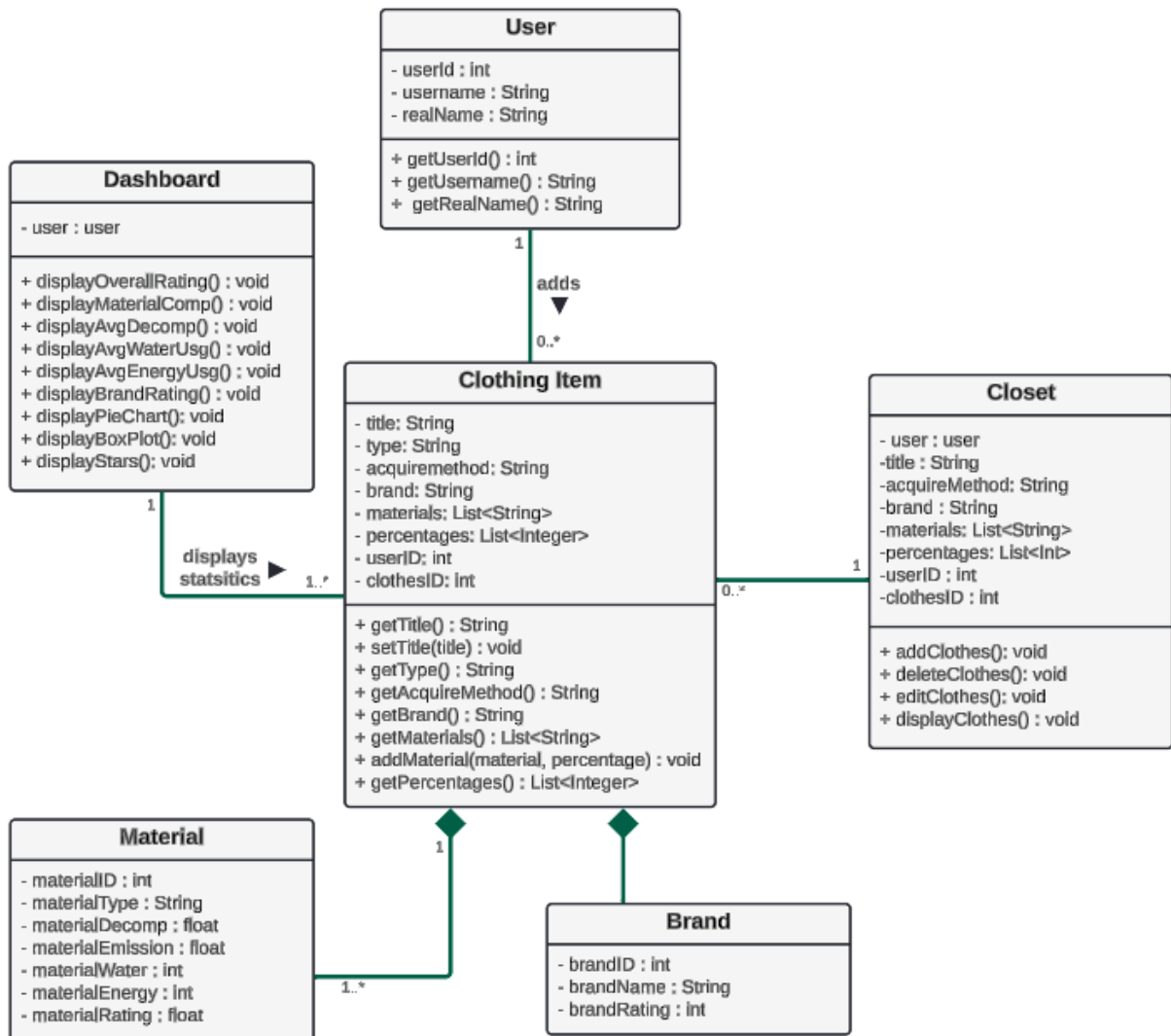
1a, 2a, 3a, 4a	Upon attempt to calculate scores without any closet data
	a1: The system creates an error message that tells the user that they must input data in the closet first.

### 3.2.2 Processing sequence for [Dashboard]



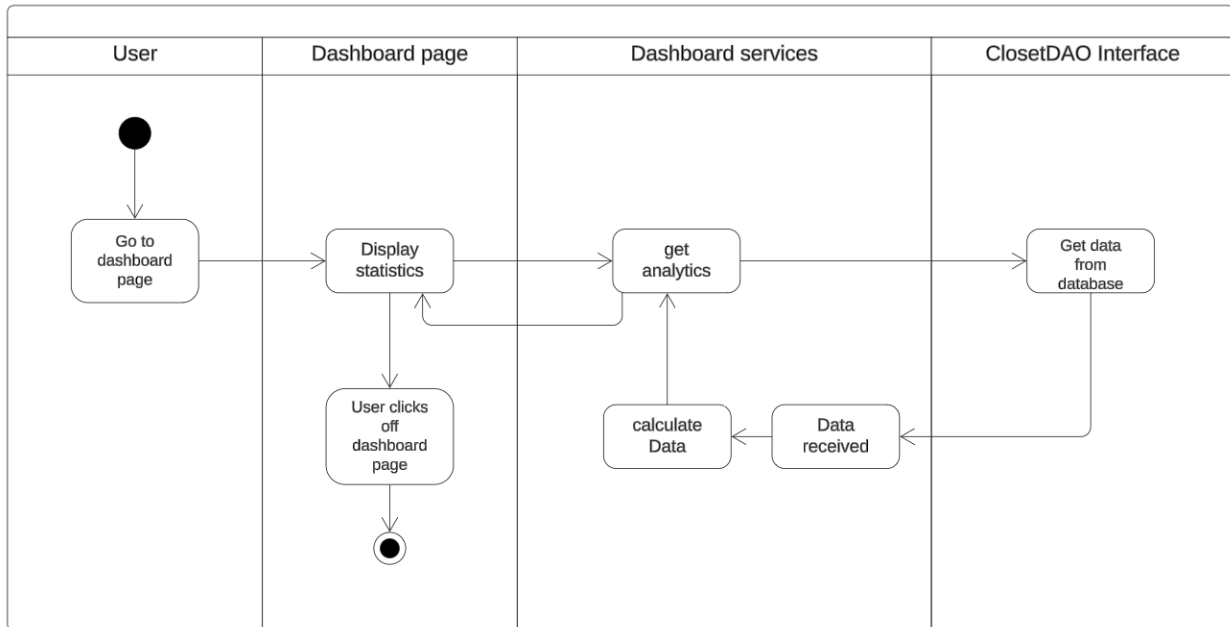
This sequence diagram illustrates the sequence of actions for the Dashboard in the Ecouture application. The process begins with the user's interaction with the NavBar (View), which redirects to the DashboardPage (View). The DashboardPage is supposed to communicate with the DashboardServices (Model) to retrieve data through the ClosetDOAInterface for processing and analysis. The data is then supposed to return and visualized as a DataChart on the DashboardPage, providing the user with a graphical representation of their wardrobe's sustainability metrics.

## 3.2.3 Structural Design for [Dashboard]



In this UML diagram of our Dashboard we see that User, Dashboard, and Closet are associated with Clothing Item Class. User class uses Clothing Item to add their items, Closet uses Clothing item to display what has been added by the user, and Dashboard uses the data from Clothing Item to then create a dashboard for the user. As previously explained as well, Material and Brand display a dependency with the Clothing Item class.

### 3.2.4 Key Activities



This activity diagram gives the overview of how the user can view their analytics from the dashboard based on their closet's composition. This diagram will involve the view, the model, controller, and closetDAO. As the user clicks on the dashboard page, they are able to view their analytics due to the services retrieving data from the database; where it will then be received by the services and calculated and displayed to the user on the dashboard page.

### 3.2.5 Software Interface to other components

The Dashboard feature interfaces with various components within the Ecouture system to display information about a user's closet data. The DashboardView provides the user interface for neatly displaying information. The Dashboard also utilizes the ClosetDOA which is an interface that connects to a database, allowing the system to retrieve clothing data in order to make and display calculations and visualize data.

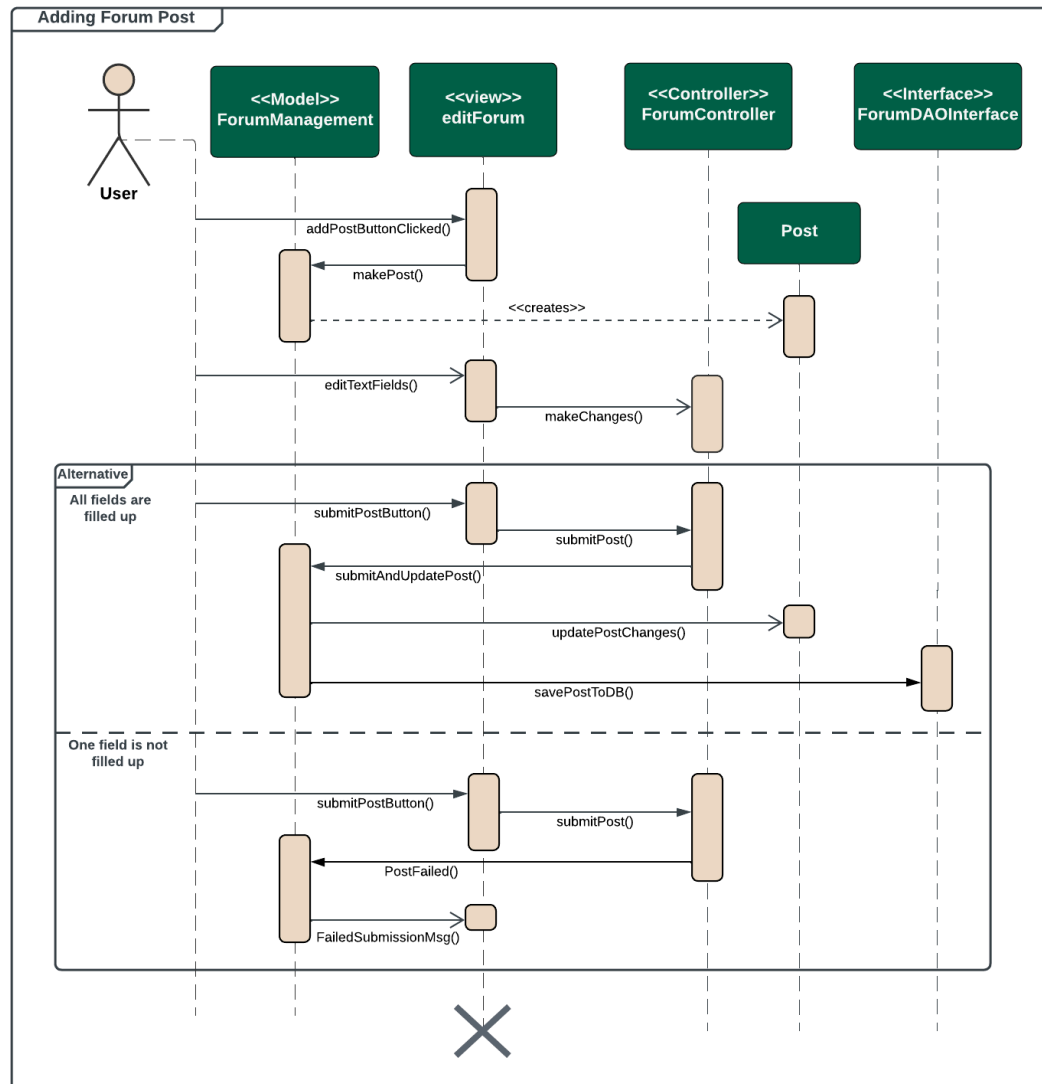


### 3.3 [Function 3: Forum]

#### 3.3.1 [Forum] Use Cases

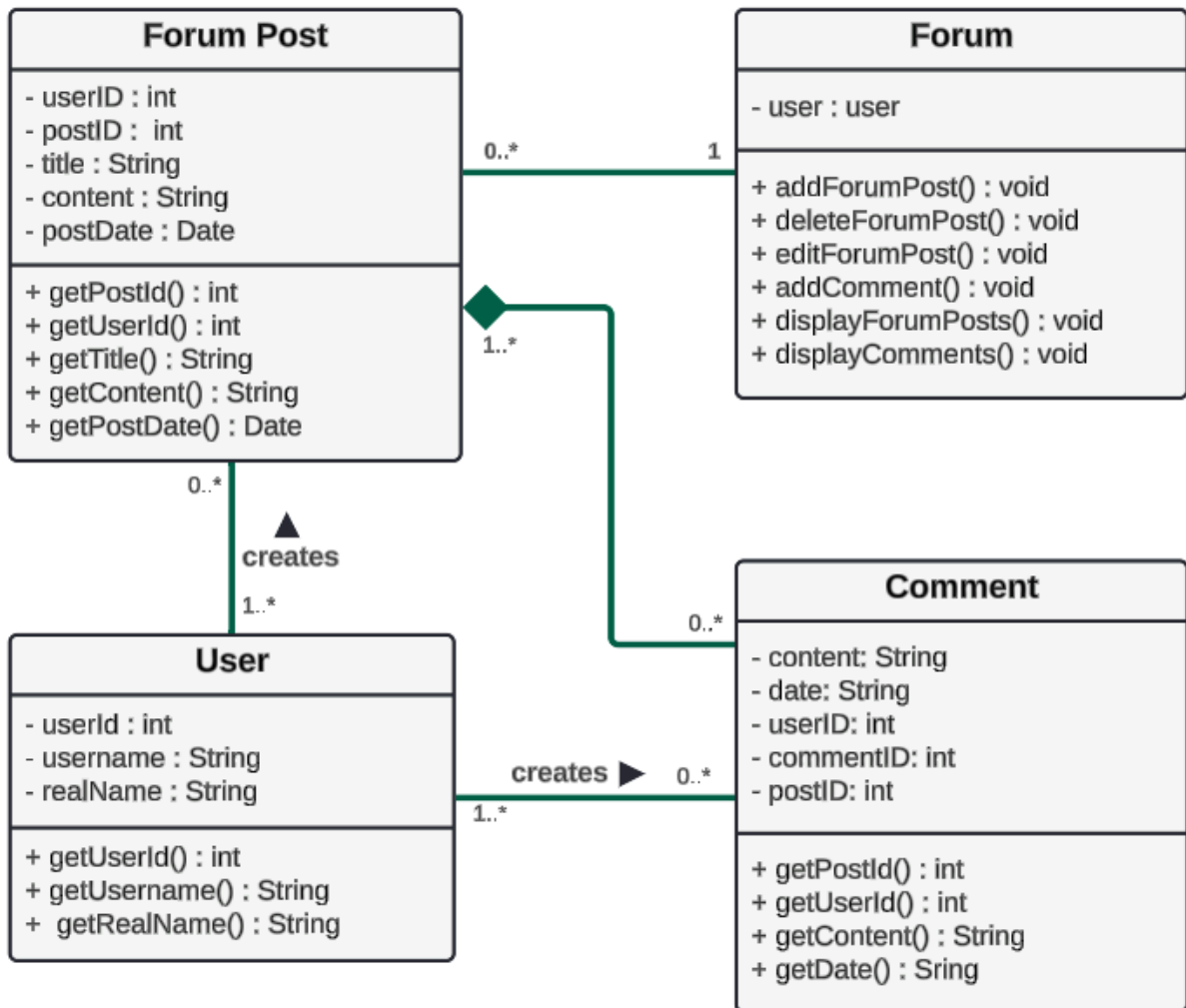
Use Case	Post on Community Forum
Goal in Context	Creating a post on the community forum
Scope	Ecouture's Forum System
Level	Primary Task
Primary Actor	Ecouture user
Preconditions	The user is logged in to their account
Minimal Guarantee	If the post cannot be created, the system will notify the user.
Success Guarantee	The user creates a post, which is visible to the community.
Trigger	User presses the forum icon on the navigation bar.
Success Scenario	Action Step
1	Within 1s the system will create a blank forum post template for the user to fill out.
2	User will fill out the forum post title via the title textbox input.
3	User will write forum content into the forum textbox input.
4	User will hit the submit button.
5	The system will save forum post data to display it.
Extension Step	Branching Action
4a	Upon trying to post onto the forum with empty textbox fields.
	a1. The system will alert the user that it's required to have some content in the textbox before posting.

## 3.3.2 Processing sequence for [Forum]



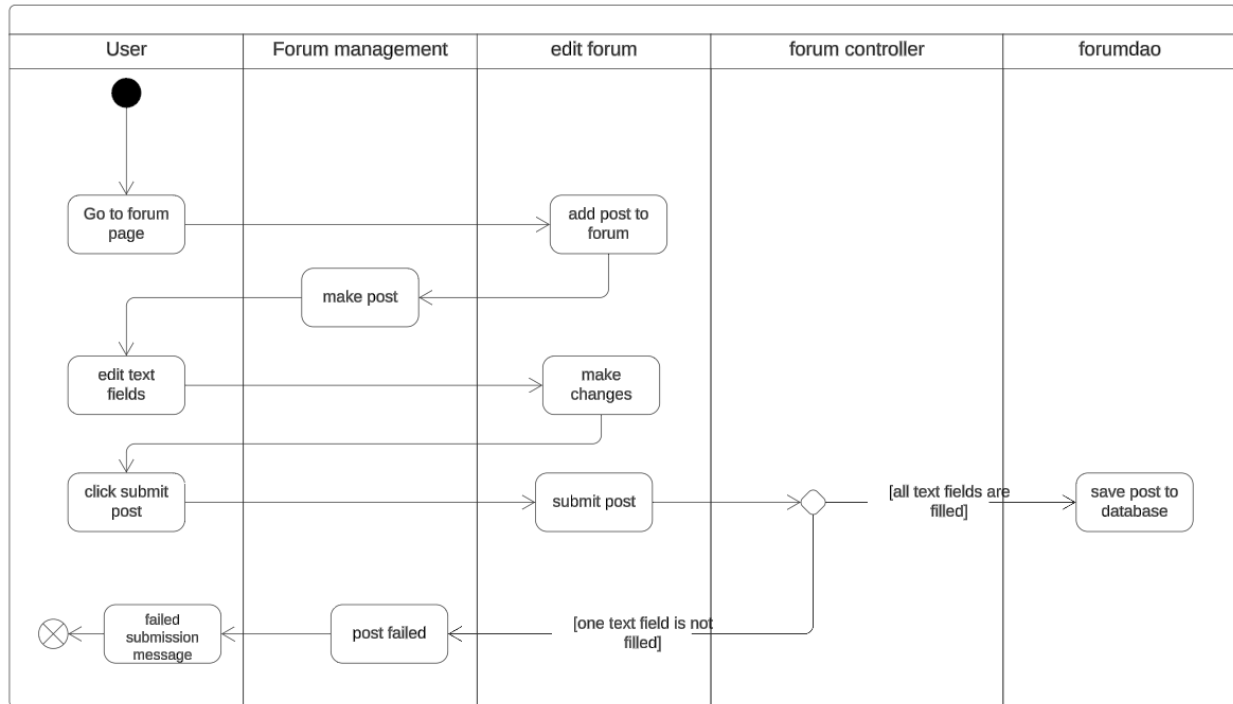
This sequence diagram illustrates the "Add Forum Post" functionality in the Forum Feature of the Ecouture application. It involves the interaction between the ForumView (View), ForumController (Controller), ForumManagement (Model), and the ForumDAOInterface (Data Access Object) components. The ForumManagement class creates or updates a Post object and interacts with the ForumDOA to persist the data into the database. When a user initiates the process to submit a forum post, the ForumPage sends the input data to the ForumController, which forwards it to the ForumManagement module to validate the request. If the inputted information is not valid, the program will display an error message.

## 3.3.3 Structural Design for [Forum]



In our UML diagram for the Forum page we can see that User is associated with the Forum Post class and Comment class. That is because User uses Forum Post to create posts to the forum and the user is able to use Comment to create comments on the Forum page. We also see that Forum Post is associated with Forum and that is because Forum uses Forum Post's posts to display. Lastly, Comment class has a dependent relationship with Forum Posts and that is due to without a Forum Post there would be no way to make comments.

### 3.3.4 Key Activities



This activity diagram shows how users can interact with the forum page of the program to add posts to the forum. There are four components that are needed for the flow of the forum page, as the user first accesses the forum page, they can choose to add a post to the forum via the edit forum, the forum management will make the post, then the user has to edit the text fields which will be reflected in the edit forum. After that, as long as all text fields have been filled, then the user's post will be pushed into the database and be displayed in the forum.

### 3.3.5 Software Interface to other components

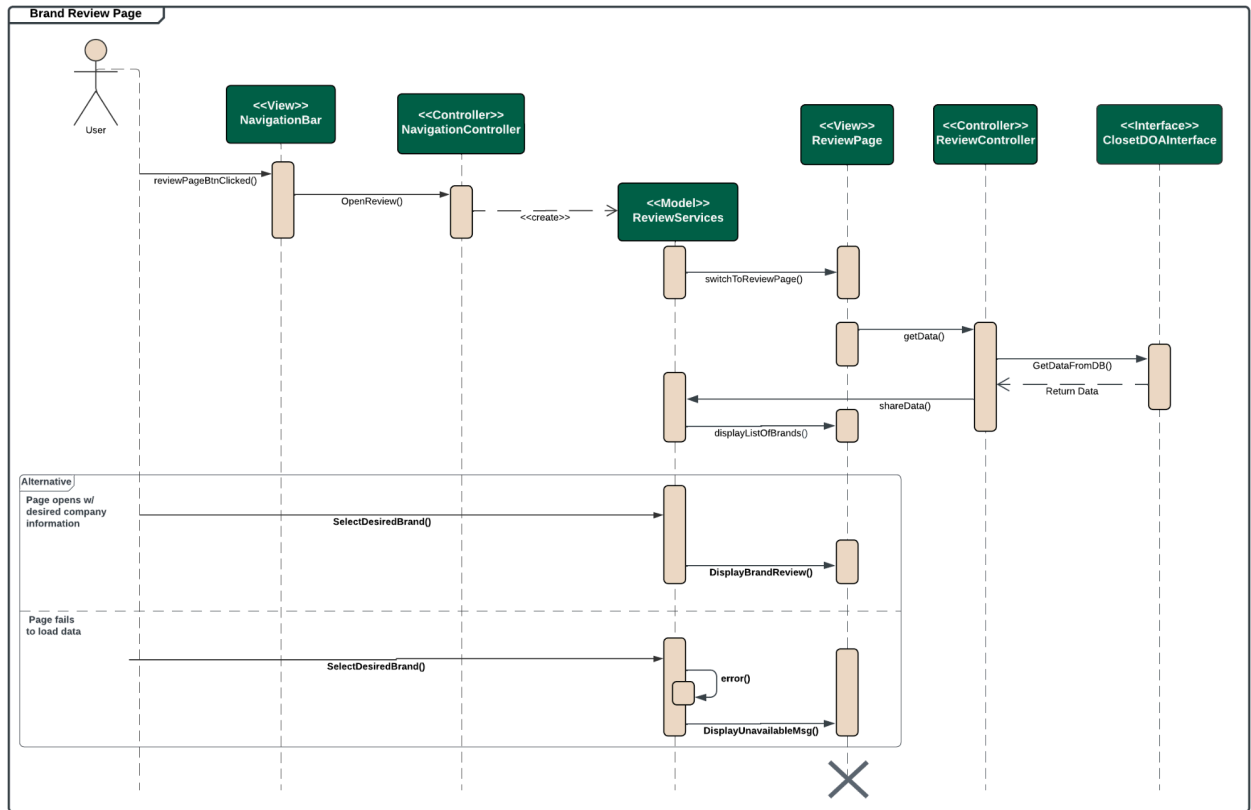
The Forum feature interfaces with many components within the Ecouture system to provide efficient management of forum discussions. The ForumViews provide the user interface for interacting with the forum so that it can execute its logic through ForumController and ForumManagement. The forumPostDOA and commentDAO is an interface that connects to a database, enabling the system to retrieve forum posts and comment data.

### 3.4 [Function 4: Brand Review]

#### 3.4.1 [Brand Review] Use Cases

Use Case	View Brand Ratings
Goal in Context	A user views brand ratings regarding sustainability
Scope	Ecouture's Brand Browse System
Level	Primary Task
Primary Actor	Ecouture user
Preconditions	The user is logged in to their account
Minimal Guarantee	The system will display default information or an error message if the analytics cannot be retrieved
Success Guarantee	The system generates a rating of brands based off their sustainability
Trigger	User presses the brand ratings icon on the navigation bar
Success Scenario	Action Step
1	The user will be able to browse through major brands (A - Z ) using a scrolling page
2	The use may click on a desired brand for more information
3	The system will create a popup that will display the information in regards to the selected brand
3	The brand will be given a rating from 1 - 5 on how sustainable it is and why they were given that rating
4	alternative brands will be offered if the rating is less than 4.
Extension Step	Branching Action
2a	popup box does not appear upon selecting desired brand
	a1: short error message will appear and give user option to redirect back to main page of Brand Browse System

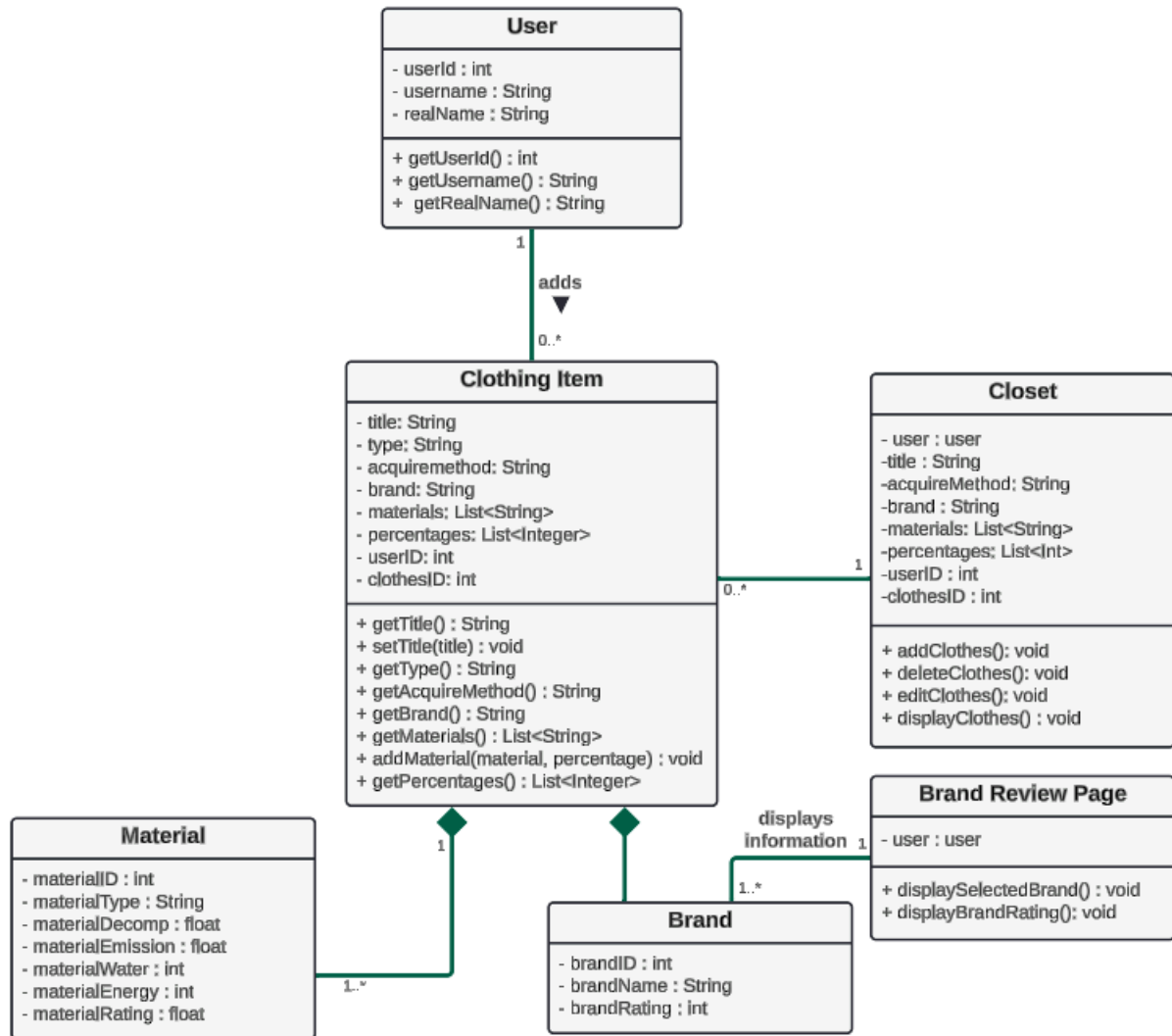
## 3.4.2 Processing sequence for [Brand Review]



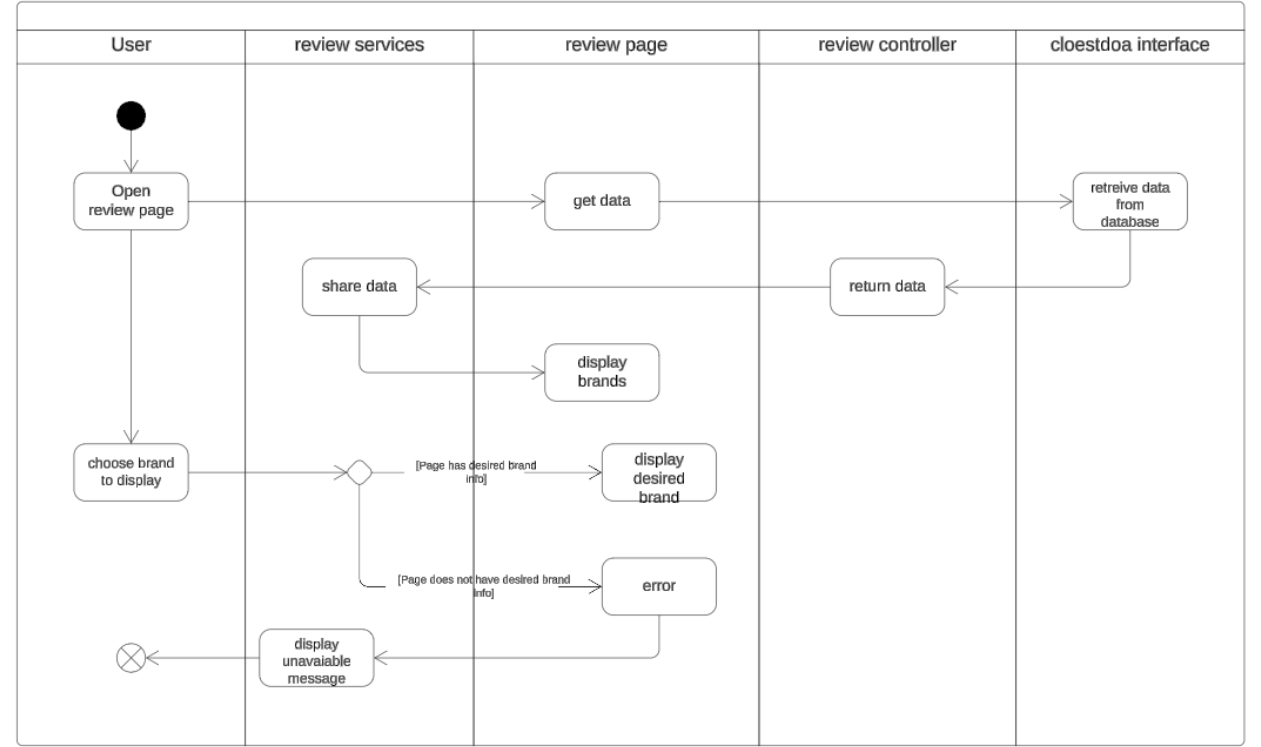
This illustrates the "Brand Review Page" feature in the Forum Feature of the Ecouture application.

The process begins with the user's interaction with the NavBar (View), which redirects to the ReviewPage (View). The ReviewPage is then supposed to communicate with ReviewServices (Model) to retrieve data through the ClosetDOAInterface for processing and analysis. The data is then supposed to return and display a star rating on the ReviewPage, providing the user with a visual representation of the rating of popular brands.

## 3.4.3 Structural Design for [Brand Review]



For the UML diagram for Brand review we are able to see that, like previously mentioned for Closets UML, that Clothing Item is associated with both User and Closet due to User and Closet using Clothing Item and that Material and Brand are dependent on Clothing Item due to the logic of if there was no Clothing Item then Material and Brand would not be needed. What is newly displayed here is that Brand Review Page shows association with Brand and that is because Brand Review Page class uses Brand and its brandRating method for its page.



This activity diagram shows how users can view the environmental impact that some of the most popular brands contribute. The users can achieve this by navigating to the review page which will then retrieve the data from the database, this is based on what brand the user decided to review, then it will return the data and share it. Then, it will be displayed in the review page for the users to get an overview of the brand impact that their clothes have. If the data is in the database, then it will display the desired brand, if not then there will be an error message and return to the review page.




The brand review module interfaces with various components within the Ecouture system to efficiently manage clothing brand data. The ReviewView provides a user interface for users to interact with the brands. The ClosetDOA is also used by this feature which connects to a database to retrieve clothing brand data in order to display it.



## 4 User interface design

### 4.1 Interface design rules

The color scheme is consistent and mainly consists of these colors:

	<b>#EBDBC3</b> rgb(235,219,195)	main background
	<b>#006349</b> rgb(0,99,73)	main title, section title, main button background, nav bar background, some icon buttons
	<b>#FFB2C2</b> rgb(255,178,194)	some icon buttons, navigation bar icons
	<b>#EFC931</b> rgb(239,201,49)	some icon buttons
	<b>#000000</b> rgb(0,0,0)	main body text, section body text, section button text
	<b>#F7F8F7</b> rgb(247,248,247)	panel / section background, main button text

Fonts are consistent through each page.

- Titles must be in the Oswald Semi-Bold font.
- Body text must be in the Lato font.

Buttons are consistent through each page (when appropriate).

### 4.2 Description of the user interface

#### 4.2.1 User Management: Login and Signup

The login and signup pages allow a user to either login to their account, or if they do not have an existing account yet, they can easily sign up and create a new account.

#### 4.2.1.1 screen Image

The image displays two side-by-side screenshots of the Ecouture web application. Both screenshots are titled 'Ecouture' in the browser window header.

**Figure 1: Login** (Left Screenshot):

- Header: Ecouture
- Logo: A green hanger with a leaf inside, and the word 'ecouture' in a green script font below it.
- Section: **LOGIN**
- Form Fields:
  - USERNAME**: A text input field.
  - PASSWORD**: A text input field.
- Buttons: Two green buttons at the bottom, labeled **SIGN UP** and **LOGIN**.

**Figure 2: Signup** (Right Screenshot):

- Header: Ecouture
- Section: **SIGN UP**
- Form Fields:
  - USERNAME**: A text input field.
  - PASSWORD**: A text input field.
  - RETYPE PASSWORD**: A text input field.
  - PREFERRED NAME**: A text input field.
- Buttons: Two green buttons at the bottom, labeled **LOGIN** and **SIGN UP**.

Figure 1: Login

Figure 2: Signup

#### 4.2.1.2 Objects and Actions

The main features on these pages are the form-like UI structure that provides the user with input textboxes. These input text boxes allow the user to fill out their information in a quick and easy way. If the user inputs invalid information, an error message will pop up to kindly remind the user on how to successfully fill out the login / signup form.

### 4.2.1 Start Page

The start page provides the user with a brief introduction on who we are (what our project is) and what our mission is (goals).

#### 4.2.1.1 screen Image



Figure 1: Start Page

#### 4.2.1.2 Objects and Actions

The main feature on this page is the navigation bar, located at the bottom of the screen. The navigation bar utilizes icon buttons that allow the user to navigate between pages, providing them with the full experience of our project. The text on the start page gives the user insight into the purpose of the program and the reasoning behind its creation.

## 4.2.2 Forum Page

The forum page allows the user to communicate with other Ecouture members to ask sustainable fashion related questions or to share their knowledge about sustainable fashion.

### 4.2.2.1 screen image



Figure 1: Main Forum Page



Figure 2: Forum Post Editor

### 4.2.2.2 Objects and Actions

The main object on this page is the "Add" icon button located in the upper-right corner, which prompts users to create posts and engage with the Ecouture community. When the "Add" button is clicked, the user is presented with two input text fields: one to enter a title for their post and another to write the content of their post. At the bottom of the page, there are

"Submit" and "Cancel" buttons, allowing the user to either post/submit their content or cancel and discard it.

On the main panel of the forum page, users can view their own posts as well as posts from other users. For engagement, there is a yellow "Comment" icon button located toward the lower center of each post, enabling users to leave comments. Additionally, a green "Edit" button is available for users to edit their existing posts, and a pink "Delete" button allows them to remove posts they no longer wish to keep.

### 4.2.3 Closet Page

The Closet page allows users to add their wardrobe to a virtual closet. In this virtual closet the user will provide a title, clothing type, brand, and materials which will all be displayed to them.

#### 4.2.3.1 screen image

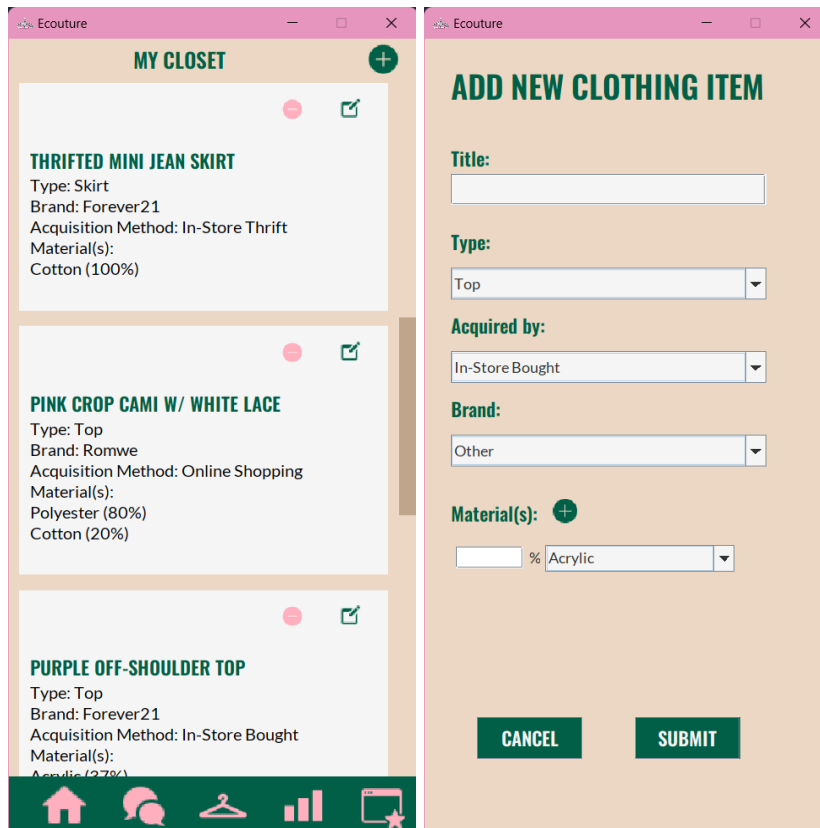


Figure 1: Closet Main Page

Figure 2: Clothing Item Editor

#### 4.2.3.2 Objects and Actions

The main feature on this page is the "Add" button, located in the upper-right corner, which prompts users to create clothing items for their closet. This allows users to receive their sustainability rating. When creating a new clothing item, the user is provided with the following fields and menus:

- A title field to name the clothing item as desired.
- A type menu bar to select the type of clothing item.
- A brand menu panel to choose the brand that made the clothing item.
- An acquisition method menu bar to specify how the item was obtained.
- A material menu bar along with a percentage text field for the user to input the materials the item is made of.

After filling in the necessary details, the user can save the clothing item using the "Save" button at the bottom of the page or cancel and discard it using the "Cancel" button, also located at the bottom of the page.

On the main panel, where existing closet items are displayed, the user can edit an existing item using the green "Edit" button located in the upper-right corner of each item. Alternatively, they can remove an item using the pink "Minus" button, also located in the upper-right corner of each item.

#### 4.2.4 Dashboard Page

The dashboard provides the user with a personalized dashboard that is based on their existing virtual closet. The dashboard gives an overall 5 star system rating and pie charts of their materials.

#### 4.2.4.1 screen image

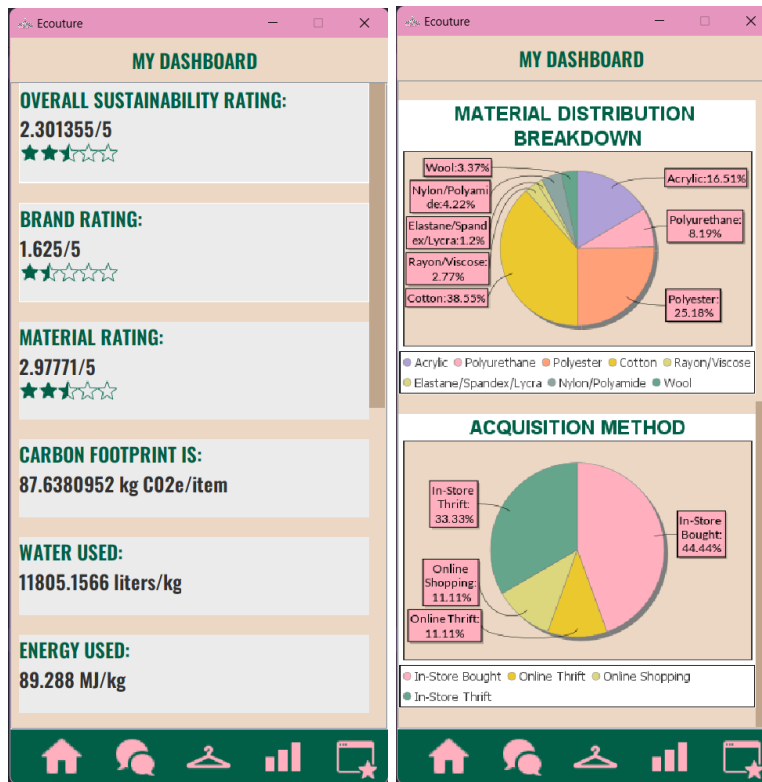


Figure 1: Top of Dashboard    Figure 2: Bottom of Dashboard

#### 4.2.4.2 Objects and Actions

The main feature of the dashboard is to display the statistics that are generated by the clothes from the closet, this includes environmental impacts, breakdowns of materials, and acquisition methods. These are then displayed into separated sections that the user can scroll through to see certain statistics such as their material rating, brand rating, and overall rating. Some of the statistics will be represented through data visualization such as pie charts, this will allow for better visualization of the composition of their closet.

#### 4.2.5 Brand Review Page

The brand review page provides the user with further insight on popular brands and their sustainability rating and a description on the brands practices.

#### 4.2.5.1 screen image

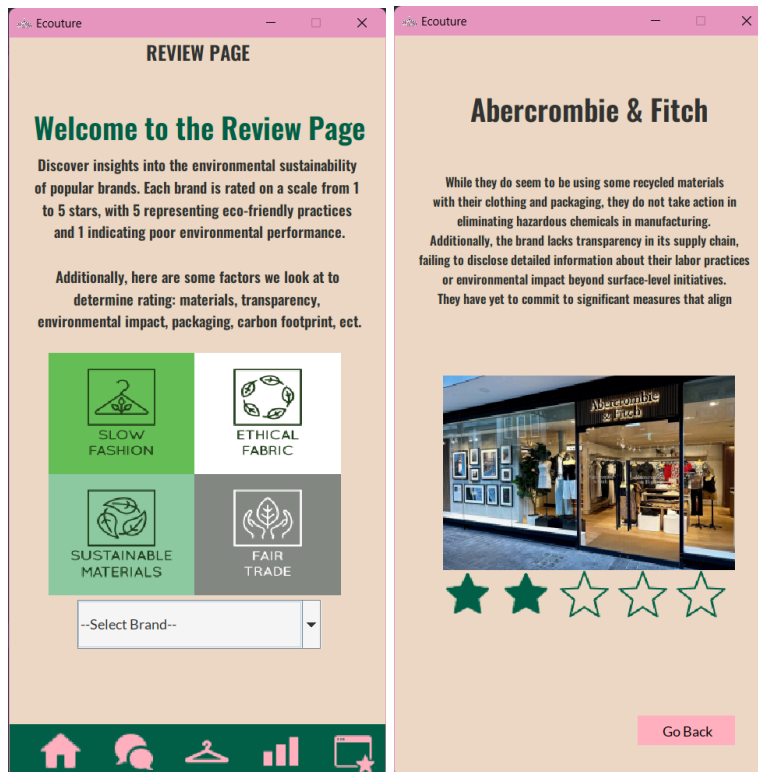


Figure 1: Brand Review Start    Figure 2: Brand Review of Specific Brand (Abercrombie & Fitch)

#### 4.2.5.2 Objects and Actions

The main feature on this page is the menu bar, which allows the user to sort through different brands and select a desired brand to learn more about. Once a brand is selected, the user is presented with text fields containing a description of the brand's sustainability practices. Each brand page includes a "Go Back" button, located in the bottom-right corner, which redirects the user back to the main panel.



## 5 Restrictions, Limitations, and Constraints

- Working on different operating systems
  - The UI looked different across devices operating with different operating systems.
- Learning Java from scratch.
  - None of us have coded in java before this class.
- Learning how to use the JDBC API to connect our application to a database.
- Delivering working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Estimation Risks
  - The size of the software is underestimated.
  - The time required to develop the software is underestimated.

## 6 Testing Issues (SLO #2.v)

Test strategy and preliminary test case specification are presented in this section. Additional test cases are located in the Appendix section in spreadsheet format.

### 6.1 Types of tests

We have performed a Performance Test and Accuracy Test for our project. We implemented an accuracy test to determine if the queries return the expected results. For example, we tested a query to see if it will accurately give the correct user's name given their user ID. We also implemented a Performance Test to see the response time for information retrieval within the forum feature. For example, we tested to see if each forum post would take less than 3 seconds to populate.

## 6.2 List of Test Cases

Test Type	Accuracy Test
Testing range	Retrieving user's information such as their ID and real name
Testing Input	User ID and real name
Testing procedure	Enter user ID and real name
Expected Test Result	The user's ID and real name will be the one returned
Testers	Jason Le
Test result	Passed

The screenshot shows a Java class named `userTest` with a method `getUserID` and a JUnit test `testGetUserInfo`. The code uses JDBC to connect to a database and retrieve user information. The test results show that the test passed successfully.

```

public class userTest {
    public user getUserID(int userID) throws SQLException {
        //Set the parameter of user_id = ? to userID
        PreparedStatement stmt = preparedStatement.setInt(1, userID);
        //Execute the statement
        ResultSet rs = preparedStatement.executeQuery();
        if (rs.next()) {
            //Stores the id from user_id column into id
            int id = rs.getInt("user_id");
            //stores the real name from the user_realname column into real_name
            String real_name = rs.getString("user_realname");
            //returns the new user
            return new user(id, real_name);
        }
        return null;
    }

    @Test //Getting the userID
    public void testGetUserInfo() throws SQLException {
        user = getUserID(1);
        assertNotNull(user);
        assertEquals("expected 1, user.getUserID()", 1, user.getUserID());
        assertEquals("Real name should be Jason", "Jason", user.getRealName());
        System.out.println("User id is " + user.getUserID() + " and the real name is " + user.getRealName());
    }
}

```

Test Results:

- userTest (model) 837 ms
- testGetUserInfo 837 ms
- Tests passed: 1 of 1 test - 837 ms

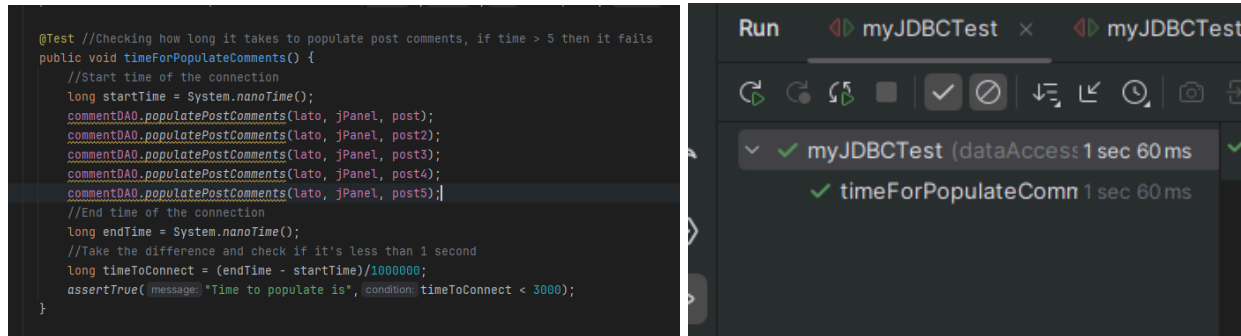
Output:

```

C:\Users\le312\.jdk\openjdk-22.0.2\bin\java.exe .
connected to my local database successfully!
User id is 1 and the real name is Jason
Process finished with exit code 0

```

Test Type	Performance Test
Testing range	Each forum post should take no longer than 3 seconds to populate.
Testing Input	A database of 5 forum posts and a request to retrieve all posts.
Testing procedure	<ol style="list-style-type: none"> <li>1. Open JDBC Connection</li> <li>2. Send a request to retrieve and display all forum posts.</li> <li>3. Measure the time taken for the posts to fully populate on the user interface.</li> </ol>
Expected Test Result	The system should successfully load all 5 forum posts within 3 seconds.
Testers	Jason Le
Test result	Passed



### 6.3 Test Coverage

As a new user, when I click the add clothing items button, I would like to see a form-like page appear so that I can understand what information is required of me in order to successfully add a clothing item to my closet.

- This has been achieved by the system as our UI provides an easy, organized way to see information that is required for the user to fill out.

As a user, I would like to be able to use a navigation bar so that I can easily navigate and access through each core functionalities of the application.

- This has been achieved by the system as we have a navigation bar implemented for the user's convenience.

As a user, I would like to be able to scroll through a page that contains a lot of information so that all the information is not cramped and hard to read.

- This has been achieved by the system as our UI includes scroll panels to scroll through each feature that contains a lot of information.

As a user who enjoys the design of applications, I would like to see an application with a set color, font, and layout theme and design so that it enhances my user experience.

- This has been achieved by the system as our UI includes scroll panels to scroll through each feature that contains a lot of information.

As a user, I would like a closet feature where I can add, delete, and edit my clothing information so that I can manipulate and see how sustainable my closet is via the dashboard.

- This has been achieved by the system as our project implements a closet feature with all 3 functions.

As a user who wants to interact with the sustainability community, I would like a forum system where I can add, delete, and edit my forum posts so that I can participate in community building.

- This has been achieved by the system as our project implements a forum feature with all 3 functions to allow the user to interact with other users in the community.

As a user who is conscious about sustainability, I would like to see the numbers and percentages of my carbon footprint so that I am aware of my environmental impact.

- This has been achieved by the system as our project implements a dashboard feature with information that updates each time a user manipulates their closet.

As a user, I want the system to process 5 forum posts within 3 seconds, so I don't have to wait long for the forum page to load up.

- This has been achieved by the system as our test shows that it takes 1.06 seconds to load up 5 forum posts.

The system must perform 100% data validation both at the application level and at the MySQL level to prevent invalid or corrupted data from being written to the database.

- This has been achieved through our hashing process for passwords, signup and login validation, and escaping sql possible syntax injections

## 7 Appendices

### 7.1 Packaging and installation issues

In order to prepare the system to run, you must first install two APIs and their respective jar files as well as MySQL Workbench 8.0. The first API is Java Database Connectivity (JDBC) (<https://dev.mysql.com/downloads/connector/j/>) and the second API is JFreeCharts (<https://www.jfree.org/jfreechart/>).

First, download mySQL WorkBench8.0 (<https://dev.mysql.com/downloads/workbench/>). Once installed, open it and create a new database and call it "jdbc\_ecouture" with the password "cs370". If you have used a different name or password, you may need to change the source code with that information directly. Once you have created the database, open a blank script workspace and copy and paste the [sql code](#) that can be found on the project's src folder under jdbc\_ecouture.sql . Once, you have pasted the sql code into your workspace, run the file and now your database is set up!

Once downloaded, you must open IntelliJ and open up the Ecouture project. If the project does not yet have a "lib" folder, create one first. Add all the jar files to your new "lib" folder. Then, go to the upper left corner of IntelliJ to the menu bar. (If you don't see the menu bar, you may have to click on the three lines first.) Then, click "File" → "Project Structure".

A modal will pop up. On the left side, there are menu options. Click on "Modules". Then navigate to where it says "Dependencies". Then, find the "+" button and click on that. Then click "Jars or Directories". A second modal will pop up, and so you must navigate back to the lib folder you created and select all the jar files you have added within that folder. Then press "OK" to escape the second modal. On the bottom of the first modal, click on "Apply" and then "OK". Then you are ready to use the application.

### 7.2 User Manual

1. Run the software application. The application will redirect the user to the login page.
2. If the user does not have a registered account, click on the "SIGNUP" button. If the user does have a registered account skip to step 6.
3. The user must fill in the input fields with their desired username, password, retyped password, and preferred name.

4. Once the user has correctly filled in all the inputs, the user may click on the “SIGN UP” button.
  - a. If the user has an invalid input, they must fix it before clicking the “SIGN UP” button.
  - b. If the user has filled in all valid inputs, their account will be successfully registered.
5. Once the user has successfully registered their account, they must then click the “LOGIN” button to return back to the login page.
6. On the Login Page, the user must fill in their correct login credentials and click on the “LOGIN” button.
  - a. If the user has an incorrect input, they must fix it before clicking the “LOGIN” button.
  - b. If the user has filled in all inputs correctly, they will be successfully logged in and redirected to the start page.
7. Once successfully logged in, the user will be directed to the start page.
8. Now, the user can either
  - use the navigation bar at the bottom to traverse through multiple functions of the application: Forum, Closet, Dashboard, and Brand Reviews Page.
  - Logout with the green logout icon button on the top right corner.

#### Forum:

1. Once the forum button is clicked from the navigation, the user can see the forum posts created by other users.
2. To comment on a post, click on the yellow comment button under a post. Type in a comment before hitting submit. Otherwise, a comment will not be sent.
3. To post a forum post, click on the green add button located at the top right of the page. This will redirect the user to a form-like page with two input boxes.
  - Forum Post Title
  - Forum Post Content
4. The user must fill up both text boxes before submitting. Else, an error message will appear.
  - If successfully submitted, the user will be redirected back to the main forum page where their new forum post will be displayed at the top.
  - If the user changes their mind about making a post, they can click on the cancel button.
5. The user also has the choice to edit their forum post via the green edit icon button or delete their forum post via the pink delete icon button located above their own forum posts.

#### Closet:

1. Once the closet button is clicked on the navigation bar, the user will be redirected to their closet page.
2. To add a clothing item to their closet, click on the green add button located at the top right of the page. This will redirect the user to a form-like page with a few text input text boxes and dropdown boxes.
  - Clothing Item Title (Textbox)
  - Type (Dropdown box)
  - Acquisition Method (Dropdown box)
  - Brand (Dropdown box)
  - Materials (Combination of textbox and dropdown box)
3. If a clothing item has a more complex material composition, there is a green add button next to “Materials” that will allow the user to add more materials to the composition list.
4. The user must fill up the Clothing Item Title text box before submitting. Else, an error message will appear.
  - If successfully submitted, the user will be redirected back to the main closet page where their new clothing item will be displayed at the top.
  - If the user changes their mind, they can click on the cancel button.
5. The user also has the choice to edit their clothing item via the green edit icon button or delete their clothing item via the pink delete icon button located above each clothing item panel.

#### Dashboard

1. Once the dashboard button is clicked on the navigation bar, the user will be redirected to the dashboard page.
2. If there are no clothing items in the user’s closet, all statistics will show as “0” and the charts will be empty.
3. If there is at least one clothing item in the user’s closet, they will be able to see the following analytics in either the form of a star rating, raw number, or pie chart:
  - Overall Sustainability Rating (Star Rating)
  - Brand Rating (Star Rating)
  - Material Rating (Star Rating)
  - Carbon Footprint (Raw Number)
  - Water Used (Raw Number)

- Energy Used (Raw Number)
- Material Distribution (Pie Chart)
- Acquisition Methods (Pie chart)

#### Review Page

1. Once the review page button is clicked on the navigation bar, the user will be redirected to the brand review start page.
2. There will display some information about what the review page is and a drop down menu.
3. Use the Drop down menu to browse through the many brands in the database and select one
4. Once selected, the user can read more information about a selected brand.
5. When they are done, they can use the “Go Back” button to go back to the main brand reviews page with the dropdown menu bar.

### 7.3 Open Issues

- Allowing the user to upload an image of their clothing item to their closet alongside the details.
- Allow the user to upload images in their forum posts.
- Allowing the application to give users personalized recommendations based on the materials in their closet or brands they shop from often.

### 7.4 Lessons Learned

- Estimation Risks
  - We bit more than we could chew when we initially started planning out our project in our project proposal. We learned that we should have been more reasonable in our proposal and estimation of the scope of our project.
- The importance of using scrum meetings
  - We learned that the more scrum meetings we had, the better our team workflow and communication was.



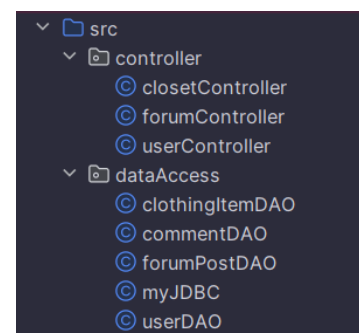
### 7.4.1 Project Management & Task Allocations (SLO #2.i)

- Our team allocated tasks and responsibilities depending on each teammates' strengths and weaknesses. If one team mate had more experience with security and backend work they took responsibility for that task. In the future, we could improve by being more confident in the language we used to build our program.
- In terms of project planning activities, our team would schedule 2-3 scrum meetings per week (Usually on monday, friday, and sunday or whenever we had time). We would complete each scrum meeting through a discord voice call or in-person in order to communicate efficiently and make changes via agile methodologies.
- When we realized that we had overestimated our time and project scope, we had to cut some unimportant features out in order to meet our deadlines in time and to focus on enhancing the important, critical features.
- This was usually the way our team updated progress through our scrum meetings:
  - At the beginning of each sprint meeting, we spoke of what we had just completed. We would check off user stories as we completed them.
  - Then, we would discuss what we needed to complete. After finding out the things we needed to complete, we allocated tasks to each other and gave ourselves deadlines on when we believed we would be able to complete each task.
- Each person in our team did well in keeping themselves responsible for their tasks and each person did well in communicating any delays they may have due to different factors.

### 7.4.2 Implementation (SLO #2.iv)

Our team conducted code reviews by pushing updated code to GitHub to stay on top of version control. If the code design was not compatible with the overall system, we would allocate one person to adjust it accordingly to ensure it works with other dependent classes.

We also refactored all our code to follow the MVC pattern and package design after all code was completed. Preferably, we should have implemented the MVC pattern from the start of developing our code. However, we



had gotten an early head start on coding and we had learned MVC much later. At that point it was difficult to implement the code refactoring in the middle of our development. So, we opted to refactor at the end once we know that our whole system works.

The screenshots provided below show how we implemented the MVC pattern which is mostly consistent with our package diagram. In comparison to our system design, our code implementation is about 80% consistent.

```
public class signupView extends JPanel { 10 usages
    public signupView(Font oswald, Font lato) { 1 usage

        //execute an attempt to login if login is clicked.
        loginButton.addActionListener(e -> userController.signupLoginButtonActionPerformed(oswald, lato, signupView: this));

        //execute signup method if signup is clicked.
        signupButton.addActionListener(e -> userController.signupSignupButtonActionPerformed(oswald, lato, signupView: this));
    }
}
```

```
public class userController { 6 usages
    public static void signupButtonActionPerformed(Font oswald, Font lato, loginView loginView) { 1 usage
        topFrame.repaint(); // Repaint the frame
    }

    public static void signupSignupButtonActionPerformed(Font oswald, Font lato, signupView signupView) { 1 usage
        //when clicked, sign up the user to database.
        userDao.signup(oswald, lato, signupView);
    }
}
```

```
public class userDao { 3 usages
    public static void signup(Font oswald, Font lato, signupView signupView) { 1 usage
        Connection connect = myJDBC.openConnection();
        //Ensure there is a connection to the database
        if (connect == null || connect.isClosed()) {
            System.out.println("Database connection is not established.");
            return; // Exit the method if connection is not valid
        }

        boolean isError = userManager.validateSignup(signupView); //check for any errors in the signup fields.

        //If fields are successful...
        if (!isError) { //if there are no errors
            userDao.insertUser(oswald, lato, signupView);
        }
    }
}
```

```
public class userManagement { 4 usages
    public static String hashPassword(String password) throws NoSuchAlgorithmException { 1 usage
        return hexstring.toString();
    }

    public static boolean validateSignup(signupView signupView) throws SQLException { 1 usage
        // To Ensure real name has no special characters
        signupView.fullname = signupView.fnField.getText(); //get name
        Pattern fullnameP = Pattern.compile(regex: "[^a-zA-Z -]"); //pattern finds special characters
        Matcher fullnameM = fullnameP.matcher(signupView.fullname); //match the string to pattern
        boolean isValidRealname = fullnameM.find(); //boolean to see if there is a match
    }
}
```

### 7.4.3 Design Patterns

The design patterns implemented in our program were the Model-View-Controller (MVC) pattern and the Data Access Object (DAO) pattern. These patterns were chosen to create a well-organized, maintainable, and modular system. By using the MVC pattern, we achieved a clear separation of concerns. Each component has a well-defined role, making the system easier to understand, maintain, and extend in the future if we wanted to. With a DAO pattern, we were able to allow our system to access the database in order to retrieve important information to display dashboard calculations and to save individual user data.

### 7.4.4 Team Communications

The team's majority of communication and scrum meetings happened via Discord through voice calls with an option of screen sharing. The team was also able to meet in person at Visat Engineering Pavillion for some scrum meetings.

Our scrum meetings usually consisted of what we completed, what we still need to complete, and what we need from each other to progress as a team. Additionally, we kept track of our user stories as a way to see progress.

Beyond scrum meetings, we also used Discord to message each other if we were unable to meet all together. Though it was not as effective as voice calls, we were able to send quick updates of our program's progress, share documents, and discuss meeting times.

In the future, we could probably use more organization and time management tools such as utilizing our Trello board more often or creating gantt charts to understand time constraints.

#### 7.4.4 Technologies Practiced (SLO #7)

The new technologies we would not have liked to use would be swing UI. We would have probably liked to opt for an easier UI designer specifically for mobile programming with the ability to drag and drop UI elements instead of hard coding them. There was an option to drag and drop, but that was really hard to understand and use, and it did not look good.

#### 7.4.5 Desirable Changes

Jason: If I had another month to work on this project, I would like to improve work on more data visualization and the design of them for the dashboard. I would also like to have found more environmental impacts that clothes would have and been able to display them.

Kate: If I had more time to work on this project, I would like to improve on code refactoring to make the code more efficient and less redundant. Additionally, I would like to do small tweaks on the UI design (especially in the dashboard and review page) to make the application more consistent throughout.

Gabby: If given more time to work on this project I would have liked to have been able to implement the image upload feature so that users could upload their clothing item image alongside their descriptions. Also I would have liked to have been able to polish the brand review page some more with alternative brand options for brands that did not have a good sustainability rating.

#### 7.4.6 Challenges Faced

Jason: I believe that system design was the hardest task as none of us had experience with Java so it was a challenge on learning and researching Java as we made our project. I think as we kept on discovering new tools for Java it gradually became easier as many tools were able to help us implement our system.

Kate: I think that system design was the hardest task because it was difficult for me to envision our project and then translate it into diagrams. Also, system implementation was just as difficult because it didn't really reflect what was required from us for the system design diagrams. So it's

either we had to change the system design diagrams or change the system's code implementation.

Gabby: Among the requirements, system design was the most time consuming followed by system implementation. With little to no experience with java turning ideas into an actual visual program took a lot of trial and error. We also had to learn each other's coding style when it came to working together on certain classes while working on the system implementation which had some setbacks but at the end it made us each better in working in group environments.