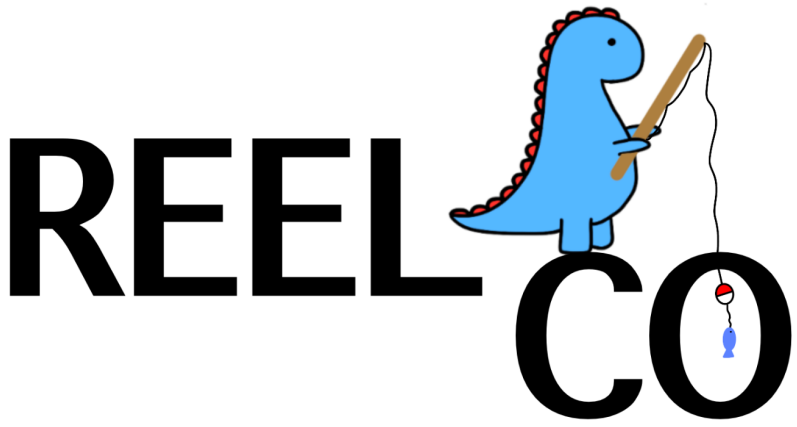
REEL CO

REEL COLOSET
Software Architecture Document

Version <1.3>



Revision History

Date	Version	Description	Author
<10/18/2022>	<1.0>	First Draft	Claire Thompson, Olivia Romig, Ron Heminway, Libby Miller, Elise Lovell, Emmy Richardson
<10/20/2022>	<1.1>	Continuing First Draft	Claire Thompson, Olivia Romig, Ron Heminway, Libby Miller, Elise Lovell, Emmy Richardson
<10/21/2022>	<1.2>	Finishing First Draft	Olivia Romig, Ron Heminway, Elise Lovell, Emmy Richardson, Claire Thompson
<10/21/2022>	<1.3>	Minor formatting fixes	Libby Miller

Table of Contents

Figures

Software Architecture Document

1. Introduction

1.1 Purpose

This document details an architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It will describe the different architectural decisions that have been made regarding the system.

1.2 Scope

This Software Architecture Document provides an architectural overview of the REEL COLOSET (RC). RC allows consumers to create an account, input their city, and input the clothing items in their closet. Then RC will generate a daily outfit based on the weather of the city the consumer's account has inputted. This product is helpful for minimizing time selecting outfits as well as providing insightful advice for creating more functional outfits.

1.3 Definitions, Acronyms, and Abbreviations

See Glossary, which is the upedu_gloss_10-02.pdf document.

1.4 References

1. REEL COLOSET - Glossary
2. REEL COLOSET - Use Case Specification
3. REEL COLOSET - Supplementary Specification

1.5 Overview

The Software Architecture Document covers the different views of the architecture design as well as the basic design principles and ideas.

2. Architectural Representation

This document presents the architecture as a series of views; use case view, process view, deployment view, and implementation view. These views are presented with Unified Modeling Language (UML).

3. Architectural Goals and Constraints

The REEL COLOSET System to be developed will be a web application tool. It consists in three major components: A web browser (or client), a web application server, and a database server.

All components must be able to execute through Modern Web Browsers such as Chrome 106, Mozilla Firefox 106, Safari 14, and Microsoft Edge 105.

The server and the Database components should be located on the same host.

4. Use-Case View

The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

Refer to Use-Case Specifications document for more information - upedu_ucspec.pdf

4.1 Use-Case Realizations

Refer Use Case Realization document – upedu_ucrea.pdf

5. Logical View

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities.

5.1 Overview

This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.

5.2 Architecturally Significant Design Packages

5.2.1 Design Model: Packages Diagram

The design model represents the structure and organizations of the RC system. Packages and corresponding classes are presented with a brief description

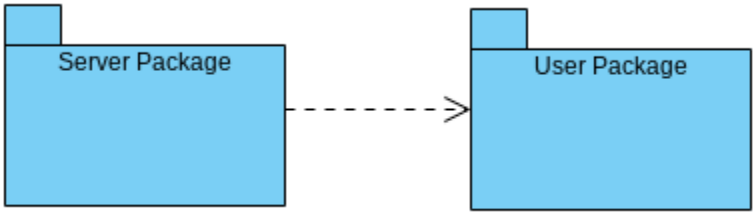


Figure 1: Design Model Package

5.2.2 Package Description

Server	
Description :	The main system package. All client queries are managed by this package.
Corresponding Classes :	RC Server
Relations :	Main RCS package. Dependant of: User
Sub-packages :	Users

User	
Description :	All information and methods regarding users are contained within this package.
Corresponding Classes:	Article Closet

	DailyOutfit Laundry User Weather
Relations:	Is a sub package of the main package Server.
Sub-packages :	None

5.2.3 Design Model: Class Diagram

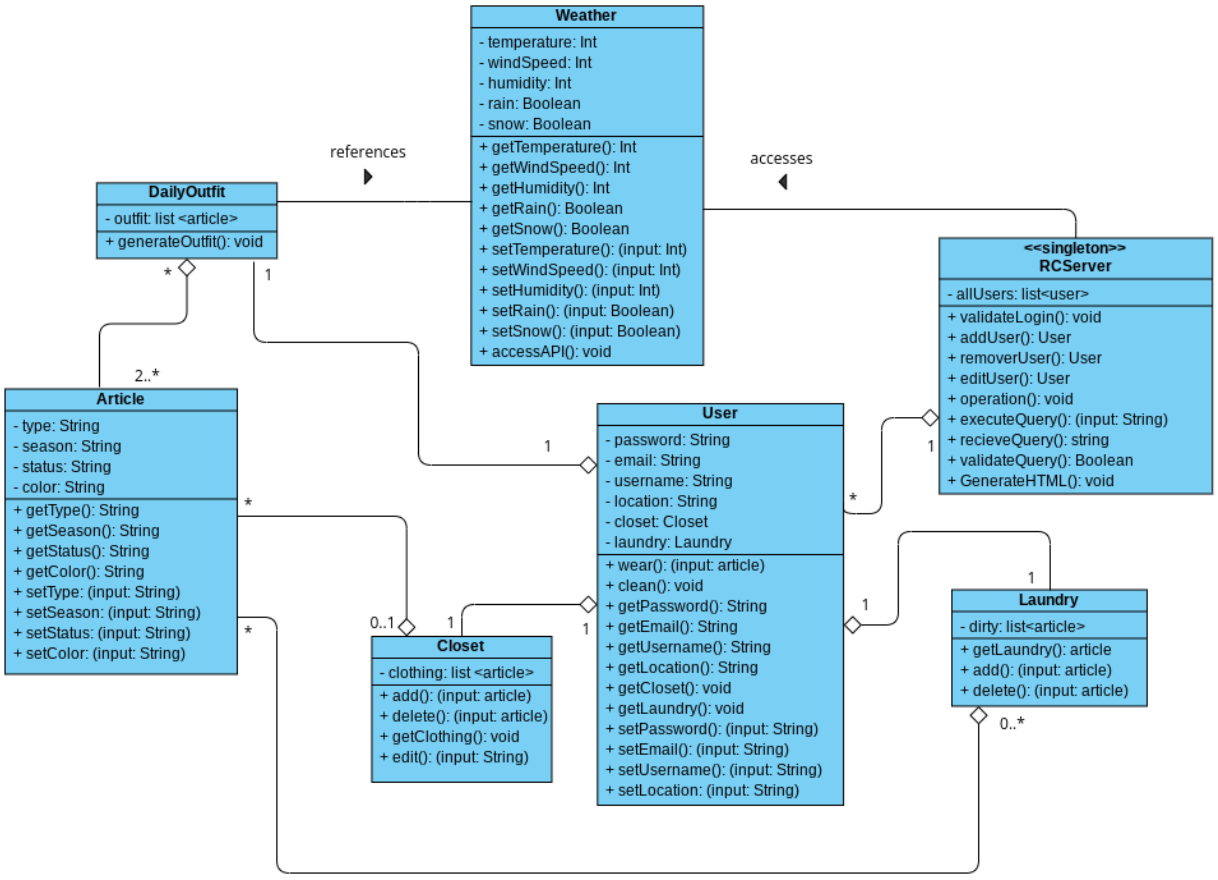


Figure 2: Class Model Diagram

5.2.4 Design Classes Description

Property	**Description**
Name	User
Description	Represents the User entity.
Responsibilities	Maintain and update all private User information.

Property	Description
	Update location of items in Closet and Laundry.
Relations	Aggregation to RCServer. The RCServer has 0 or more users. Aggregation with DailyOutfit, Closet, and Laundry. A user has 1 daily outfit, 1 closet, and 1 laundry basket.
Methods	wear(): Places a specific article of clothing into the laundry clean(): Gets all articles of clothing out of the laundry getPassword(): Retrieves password from server getEmail(): Retrieves password from server getUsername(): Retrieves username from server getLocation(): Retrieves location from server getCloset(): Retrieves list of articles in closet from server getLaundry(): Retrieves list of articles in closet from server setPassword(): Sets new password setEmail(): Sets new email setUsername(): Sets new username setLocation(): Sets new location
Attributes	password: The private variable for the user's password email: The private variable for the user's email username: The private variable for the user's username location: The private variable for the user's password closet: The public list of articles the user has in their closet laundry: The public list of articles the user has in their laundry
Special Requirements	None

Property	Description
Name	Closet
Description	Represents a Closet entity that will contain all Articles of clothing belonging to the User.
Responsibilities	Maintain the list of Articles in the Closet.
Relations	Aggregation to User. A Closet belongs to one user.
Methods	add(): Adds a new article of clothing into the Closet. delete(): Deletes an article of clothing from the Closet. getClothing(): Gets an article of clothing from the Closet. edit(): Edits an article of clothing in the Closet.
Attributes	clothing: The list of articles of clothing in the Closet.
Special	None

Requirements	

Property	**Description**
Name	Laundry
Description	Represents a Laundry entity and contains all Articles entities that have been worn.
Responsibilities	Maintain the list of Articles in the Laundry.
Relations	Aggregation to User. Each Laundry belongs to one user.
Methods	getLaundry(): Gets an article of clothing from the laundry. add(): Adds a new article of clothing into laundry. delete(): Deletes an article of clothing from laundry.
Attributes	laundry: The list of articles of clothing in laundry.
Special Requirements	None

Property	**Description**
Name	Weather
Description	Represents and keeps track of the current weather status.
Responsibilities	Interact with the weather API. Maintain accurate and current weather information.
Relations	Association with DailyOutfit and RCServer. Weather is referenced by DailyOutfit and Weather is accessed by RCServer.
Methods	getTemperature(): Returns the temperature getWindSpeed(): Returns the wind speed. getHumidity(): Returns the humidity getRain(): Returns whether it is raining or not. getSnow(): Returns whether it is snowing or not. setTemperature(): Modifies the temperature. setWindSpeed(): Modifies the wind speed. setHumidity(): Modifies the humidity. setRain(): Modifies the status of rain. setSnow(): Modifies the status of snow. accessAPI(): Sends a fetch request to the weather API and sets all attributes according to the data received.
Attributes	temperature: The temperature at the location of the user.

	windSpeed: The wind speed at the location of the user. humidity: The humidity at the location of the user. rain: The status of rain at the location of the user. snow: The status of snow at the location of the user.
Special Requirements	None

Property	**Description**
Name	RCServer
Description	Handels all interactions with the database.
Responsibilities	Maintain an updated list of users. Process and receive all queries from client to server and database.
Relations	Aggregation with User. A User interacts with one RCServer. Association with Weather. RCServer accesses Weather.
Methods	validateLogin(): Validates the login information of the user addUser(): Adds a user into the system removeUser(): Removes a user from the system editUser(): Modifies the information of a user operation(): executeQuery(): Executes a query receiveQuery(): Receives a query validateQuery(): GenerateHTML():
Attributes	allUsers: List of all users
Special Requirements	None

Property	**Description**
Name	Article
Description	Represents an article of clothing.
Responsibilities	Model an article entity and update the status of the Article.
Relations	Aggregation to DailyOutfit. DailyOutfit has 2 or more Articles. Aggregation to Closet. Closet has 0 or more Articles. Aggregation to Laundry. Laundry has 0 or more Articles.
Methods	getType(): Returns the type of clothing

	getSeason(): Returns the season of the article getStatus(): Returns the laundry status of the article of clothing getColor(): Returns the color of the article of clothing setType(): Modifies the type of clothing setSeason(): Modifies the season setStatus(): Modifies the laundry status of the article of clothing setColor(): Modifies the color of the article of clothing
Attributes	type: The type of clothing season: The season the item is worn in status: The current laundry status of the article of clothing, whether it is in the Laundry or Closet color: The color of the article of clothing
Special Requirements	None

Property	**Description**
Name	Daily Outfit
Description	Represents the daily outfit that will be recommended to the user.
Responsibilities	Create the daily generated outfit. Model the daily outfit entity. Interact with the weather API.
Relations	References the Weather. Aggregation to User. The User has 1 DailyOutfit.
Methods	generateOutfit(): Creates a randomly generated outfit
Attributes	outfit: A list of Articles making up the generated outfit
Special Requirements	None

6. Interface Description

The interface will be a website with buttons that the user can click on to navigate to different functions of the website. There is a button for logging in/logging out, create account, edit account, laundry list, closet list, and daily outfit. Once clicked the expected functionality will occur for the user. There will also be spaces for when clicked on the laundry list, closet list, and daily outfit will be presented on the webpage.

7. Size and Performance

The architecture for our system supports requirements for sizing and timing by using a client-server architecture. The user will access the system through an internet browser, and therefore minimal memory requirements will be needed from the client.

8. Quality

The software architecture supports the quality requirements, as stipulated in the Software Requirements Specification and Supplementary Specification.