

# WardrobeSystemDocs

## Wardrobe System - Complete Developer Documentation

### What Is This Project?

The **Wardrobe System** is a fully functional, plug-and-play clothing and outfit management system for Unreal Engine. It allows players to interact with a wardrobe, change individual clothing pieces (head, upper body, lower body, shoes, **or more! YES! you can have more slots or even less slots, and *the system adapts to your choice!*** saving both performance and your precious time!), switch complete outfits, and see their character react with animations when trying on clothes.

### Key Features:

- ✓ **Modular Clothing System** - The count of Skeletal Mesh Component is up to you
  - ✓ **Pre-Made Outfits** - Create and equip complete outfit sets instantly
  - ✓ **Interactive Wardrobe Trigger** - 3D wardrobe prop with collision-based interaction
  - ✓ **Cinematic Camera** - Smooth camera transition when entering wardrobe mode
  - ✓ **(BONUS) Character Reaction Animations** - Characters play animations when trying on clothes
  - ✓ **UI Widget System** - Automatically adapts to how much "Wardrobe Groups" you have!
  - ✓ **Data-Driven Architecture** - All clothing and outfits are defined using Data Assets
  - ✓ **UE4 & UE5 Mannequin Support** - Includes example clothing for both skeleton types
  - ✓ **MetaHuman Compatible** - Example character blueprint supports MetaHuman meshes
- 

### What Can You Do With It?

- Add wardrobe functionality to your game characters
  - Create clothing stores, dressing rooms, or character customization systems
  - Build fashion games, RPGs with equipment systems, or social simulation games
  - Extend it with save/load systems, multiplayer synchronization, or shopping mechanics
-

## Project Structure Overview

```
SimpleSeries/Wardrobe/
├── Blueprints/
│   ├── AC_SimpleWardrobe.uasset           // Main wardrobe component
│   ├── BP_WardrobeTrigger.uasset         // Interactive wardrobe actor
│   ├── Data/                             // Data Asset base classes
│   │   ├── PDA_ClothPieceData            // Individual clothing item
│   │   ├── PDA_ClothingGroup             // Collection of clothing items
│   │   └── PDA_OutfitData                // Complete outfit definition
│   ├── Game/
│   │   ├── BP_TPC_WC.uasset              // UE4 Mannequin character example
│   │   └── BP_TPC_MH_WC.uasset           // MetaHuman character example
│   └── Widgets/
│       ├── WB_WardrobeSelect             // Main wardrobe UI
│       ├── WB_ClothingGroupSlot          // Clothing category button
│       └── WB_OutfitSlot                 // Outfit button
├── Clothes/
│   ├── Data/                             // All clothing data assets
│   │   ├── Items/                       // Individual clothing pieces
│   │   ├── Outfits/                     // Pre-made outfit combinations
│   │   └── WardrobeGroups/              // Clothing collections by
├── category
│   └── UE4/                             // Skeletal meshes for clothing
├── Animations/
│   └── Data/                             // Character reaction animations
├── Maps/
│   └── LVL_WardrobeDemo.umap             // Demo level
```



# Quick Start Guide - Using the System

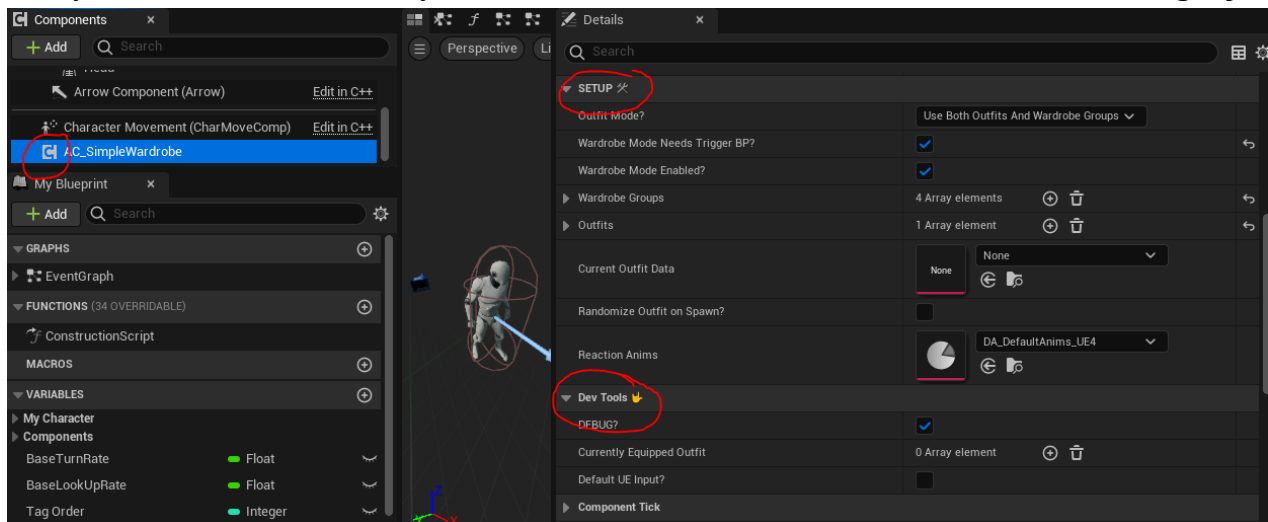
Few things to keep in mind. If you don't wanna read the whole doc,

**just read and internalize this.**

If you're already advanced in Blueprints,

you'd get the whole system immediately just from this.

- I'd recommend you to play around the level, and check out all the possibilities with the Placeholder Assets I've prepared for this demo first
- **"Cloth Piece" Data Asset is the Building block of All Clothes.** It consists of 3 variables, the most important of them; "DesignatedTag" bcs this has to match with the "ComponentTag" of your Skeletal Mesh Component in your CharacterBP
- "WardrobeGroups" Data Asset consists of "ClothPiece" Data assets with the same "DesignatedTag"
- And "Outfits" are just a set of "Cloth Piece" Data, each with unique "DesignatedTag"
- Widgets that are shown are **modular** and are **based on the same building blocks**
- Every variable that concerns you as devs are in the **"SETUP" or "DevTools" category**



- There are comments and a lot of variables have tooltips to help you understand everything better

**That's it! Simple, right?**

If you're a beginner, don't worry, I have this whole doc prepared, just for you :D

Don't be intimidated, you're very welcome to read the whole thing carefully :)

I mean, If you're like me... you can always dive in head-first and experiment, maybe come back here when you break things, lol

And if you're lost, just hmu on Discord :D

Good Luck with your Project!

# Step-by-Step Customization Guide

## PART 1: Adding the Wardrobe to Your Character

### Option A: Use the Provided Example Character

#### 1. Find the character blueprints:

- For UE4 Mannequin: `/Game/SimpleSeries/Wardrobe/Blueprints/Game/BP_TPC_WC`
- For MetaHuman: `/Game/SimpleSeries/Wardrobe/Blueprints/Game/BP_TPC_MH_WC`

#### 2. Set it as your player character:

- Open your Game Mode blueprint
- Set **Default Pawn Class** to one of the character blueprints above

### Option B: Add Wardrobe to Your Existing Character

#### Prerequisites:

- Your character must have a main **Skeletal Mesh Component**
- Your character must use a **Character Blueprint** (derived from `Character` class)

#### Steps:

#### 1. Open your character blueprint

#### 2. Add the Wardrobe Component:

- In the **Components** panel, click **Add Component**
- Search for `AC_SimpleWardrobe`
- Add it to your character

#### 3. Add Skeletal Mesh Components for clothing slots:

- Add **4 Skeletal Mesh Components** as children of your main character mesh
- Name them whatever you want, for example :
  - Head
  - UpperBody
  - LowerBody
  - Shoes
- Set each component's **Component Tags** to match their name (e.g., Head component gets "head" tag)
- Set their **Leader Pose Component** to your main character mesh in the Construction Script
- Set **Skeletal Mesh Asset** to `None` (clothing will be assigned at runtime)

#### 4. Configure the Wardrobe Component:

Select the `AC_SimpleWardrobe` component and checkout the "SETUP" section:

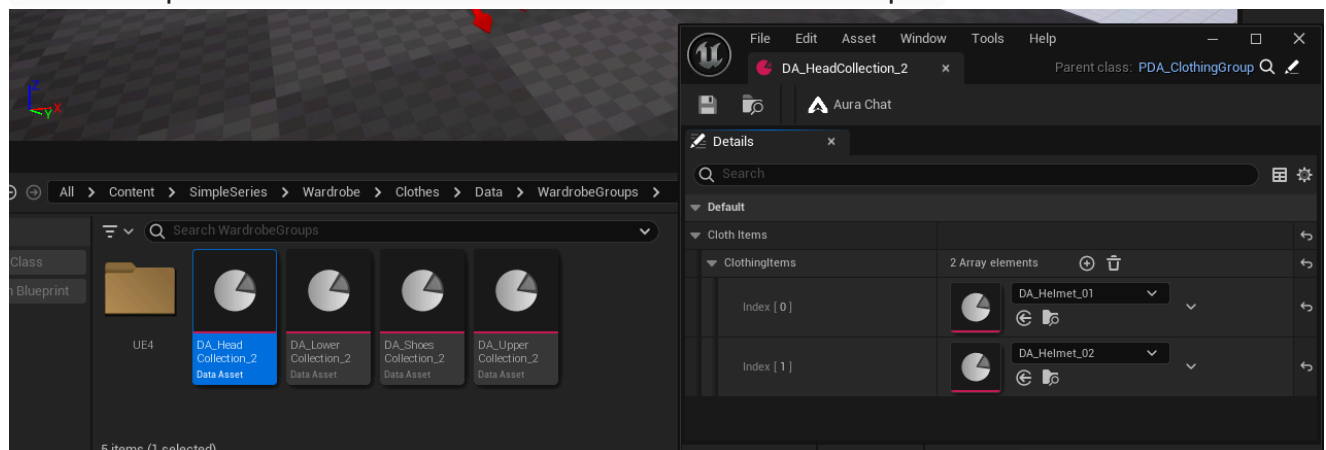
<b>WardrobeModeNeedsTriggerBP?</b>	<b>True = Needs BP_WardrobeTrigger on the level, False = You can access it anytime</b>
<b>WardrobeModeEnabled?</b>	True = Enables the wardrobe system False = Disables it
<b>OutfitMode?</b>	Outfit & Wardrobe Groups = uses both Wardrobe Groups = Individual Clothing Selections Outfits Only = Pre-Made Clothing Selections
<b>Wardrobe Groups</b>	Put your ClothPiece Collection here
<b>Outfits</b>	Put your Outfits here
<b>ReactionAnims</b>	Open the Data Asset to customize it
<b>RandomizeOutfitOnSpawn?</b>	Based on the Array of "Outfits" you've added

These Variables have their Tooltips too, so you can check them out :)

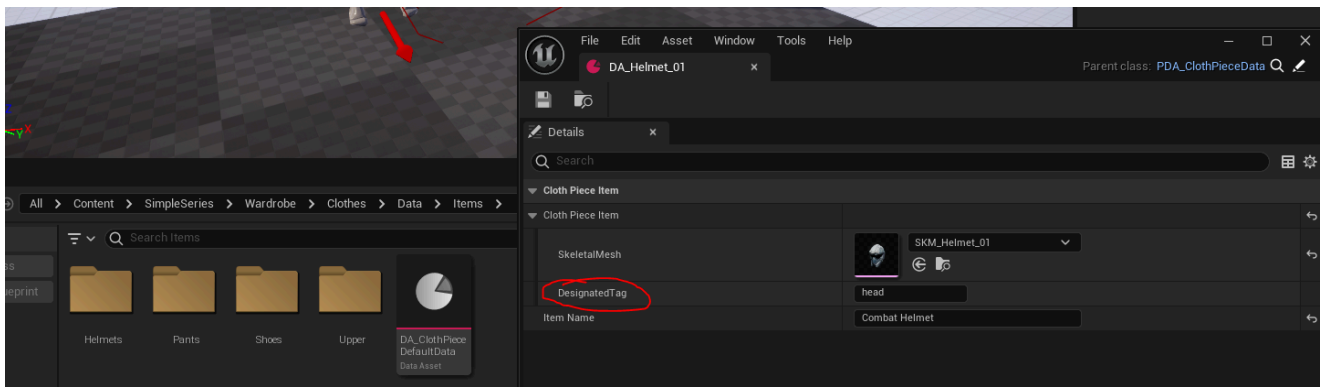
#### 5. Assign Clothing Collections (Wardrobe Groups):

There are 4 Groups, bcs I've setup 4 Skeletal Meshes, but it doesn't have to be 4. it can be Less and More. However many you've put your Skeletal Meshes on your character, the Wardrobe Groups would recognize it as long as you have assigned its tag :)

To create your own Wardrobe Group : Just duplicate one of the DA in  
`/Game/SimpleSeries/Wardrobe/Clothes/Data/WardrobeGroups/`



Alright, All these Data Assets hold a specific type of Data Assets called "ClothPiece"  
And its from `/Game/SimpleSeries/Wardrobe/Clothes/Data/Items/`



Pay attention to "DesignatedTag", Because an Ideal "WardrobeGroup" Data asset is the one containing "ClothPiece" Data Assets **with the exact same "DesignatedTag"**

## 6. Assign Pre-Made Outfits:

In the **Outfits** array, add outfit data assets:

- /Game/SimpleSeries/Wardrobe/Clothes/Data/Outfits/
- Add more from the `Outfits` folder as needed

To Create your own outfit, just duplicate any Outfit data Asset, and add your own "ClothPiece" items onto the array. You can add multiple items of **different "DesignatedTag"**.

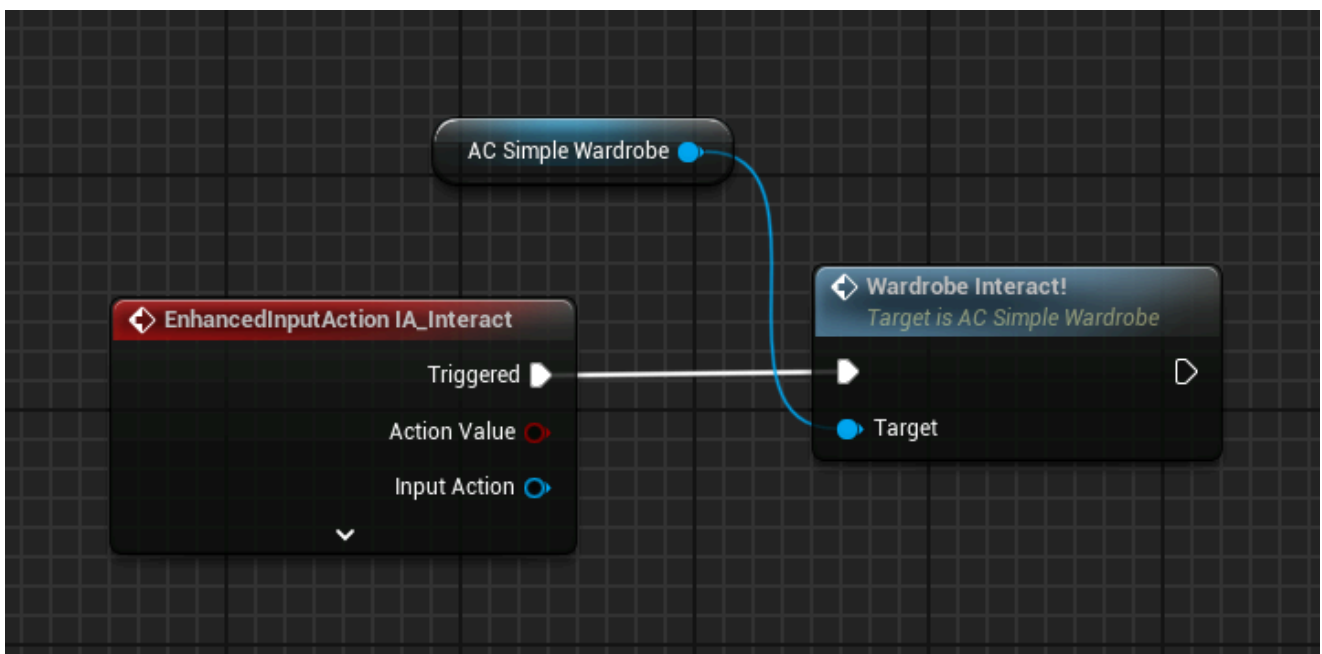
If you add multiple items with the same tag, though... the system would just replace the previously applied with the last one... so it's a redundant thing to do.

## 7. Setup Input (if not using BP\_WardrobeTrigger):

If you want to open the wardrobe directly without a trigger actor:

- Set **WardrobeModeNeedsTriggerBP?** to `False`
- In your character's **Event Graph**, call the custom event:  

Wardrobe Interact! (on AC\_SimpleWardrobe component)



## PART 2: Adding the Wardrobe Trigger Actor to Your Level

The **BP\_WardrobeTrigger** is a physical wardrobe prop that players can interact with.

### 1. Drag the wardrobe into your level:

- Open your level
- Navigate to: `/Game/SimpleSeries/Wardrobe/Blueprints/BP_WardrobeTrigger`
- Drag it into your level

### 2. Position the wardrobe:

- The wardrobe includes a **placement guide arrow** (visible in editor only)
- Position your player character where the arrow points for proper camera framing

### 3. Configure the wardrobe actor:

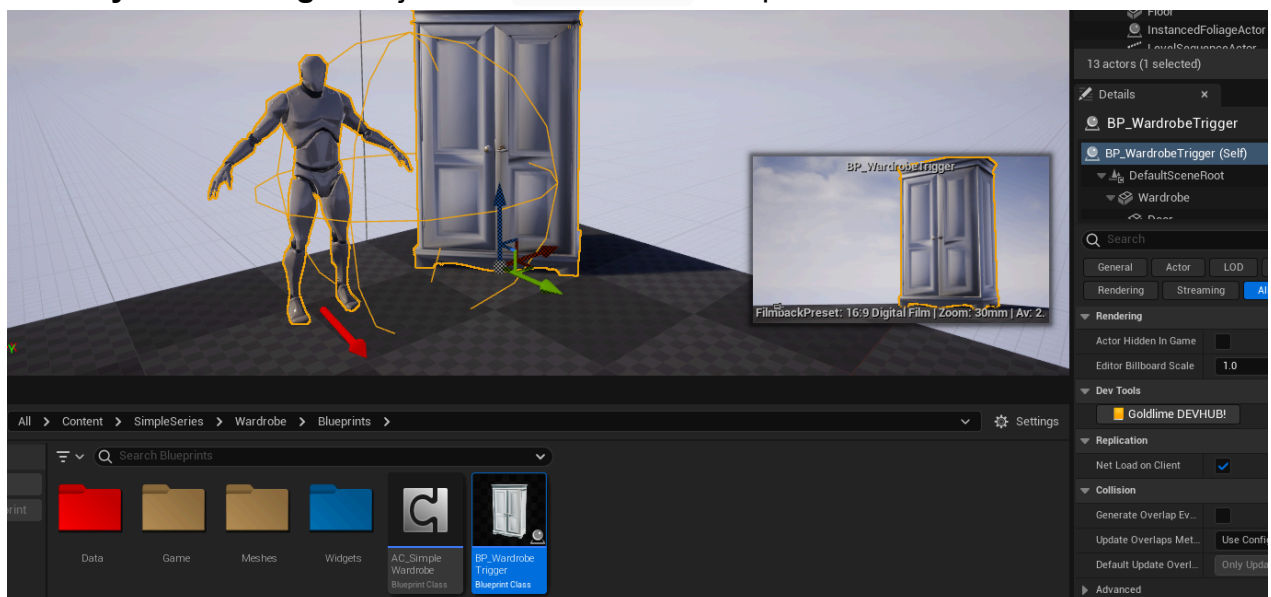
<b>UseWardrobeMesh?</b>	<b>Shows the wardrobe 3D model</b>
<b>LerpRotationOnly?</b>	Only rotates player, doesn't move them

### 4. How it works:

- Players walk into the **sphere collision** (green sphere in editor)
- A **3D widget button** appears prompting "Press E to interact"
- Pressing **E** opens the wardrobe UI and transitions the camera

### 5. Customization options:

- **Change the mesh:** Assign a different Static Mesh to the `Wardrobe` component
- **Change interaction key:** Modify the `IA_Interact` Input Action
- **Adjust collision range:** Scale the `TRIGGER` (SphereComponent)
- **Modify camera angle:** Adjust the `CineCamera` component's transform



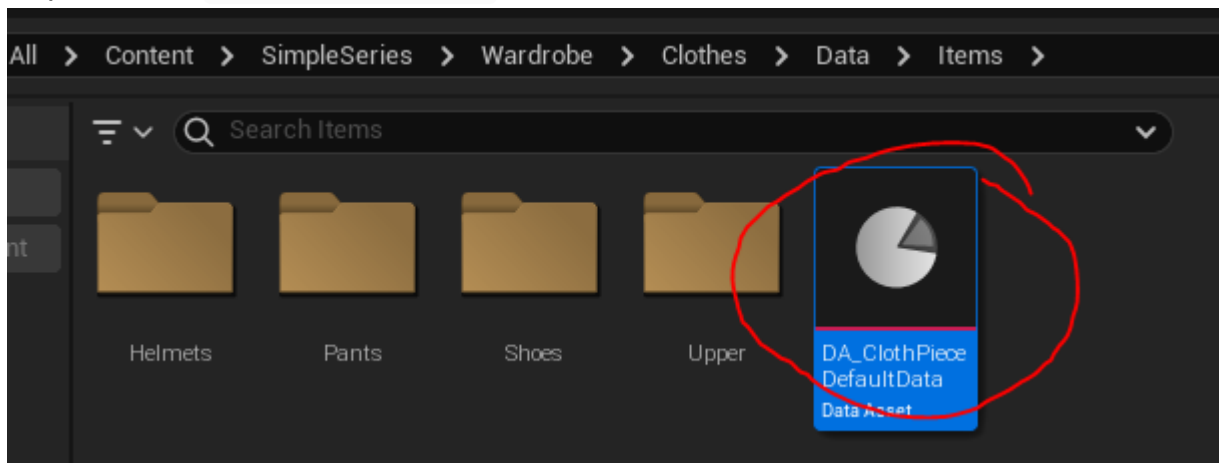
## PART 3: Creating Your Own Clothing Items

### Step 1: Prepare Your Skeletal Mesh

1. Import your clothing mesh into Unreal Engine
2. Ensure it uses the same skeleton as your character (e.g., UE5\_Mannequin\_Skeleton)
3. Save it in: /Game/SimpleSeries/Wardrobe/Clothes/ (or your own folder)

### Step 2: Create a Clothing Data Asset

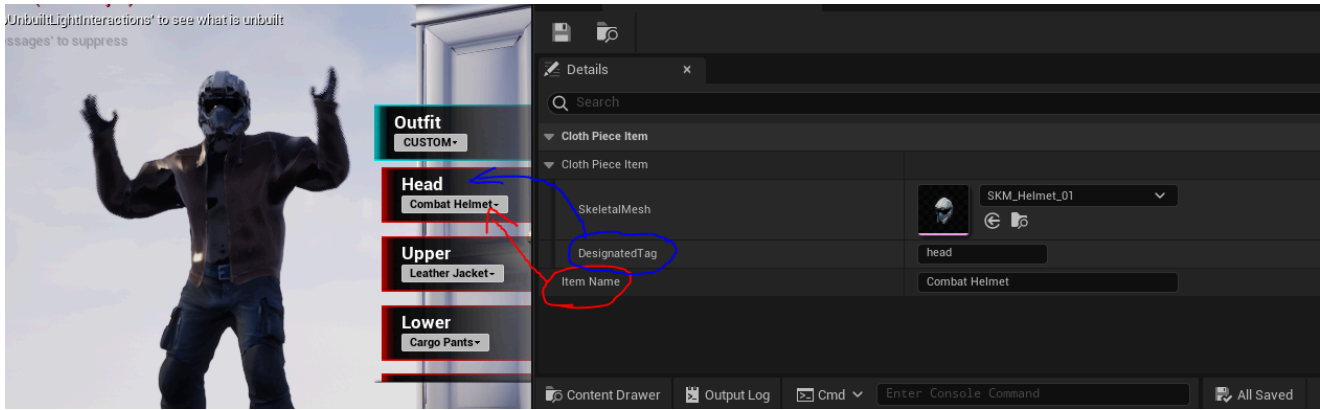
1. Navigate to: /Game/SimpleSeries/Wardrobe/Clothes/Data/Items/
2. Duplicate the DA\_ClothPieceData



3. Name it: DA\_YourClothingName (e.g., DA\_Jacket\_Leather )
4. Open the new Data Asset and configure:

<b>SkeletalMesh</b>	<i>(Your imported mesh)</i> it ideally should use the same skeleton as your CharacterBP	The SKM designated to the SKM in your CharacterBP
<b>DesignatedTag</b>	head? lower? earrings? gloves?	Which body slot this clothing occupies
<b>ItemName</b>	This is What the player sees	Display name for the item





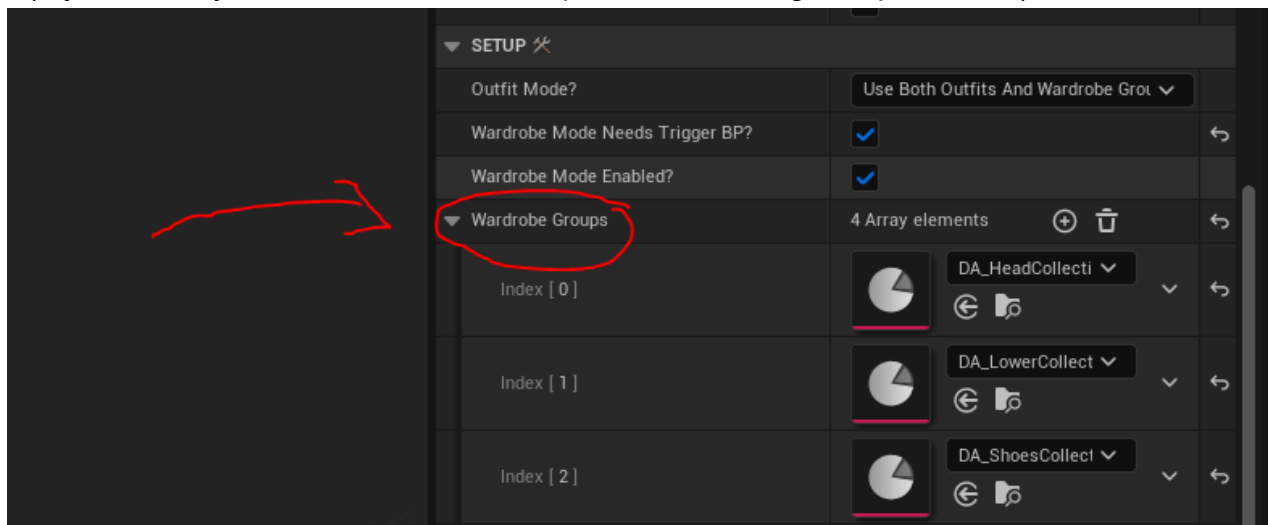
So the Tag, would be what is displayed in the Widget, (there's a code that sets the 1st letter to be uppercase so) we as developers can just focus on making sure the DesignatedTag is matching with the Designated Skeletal Mesh on your CharacterBP

And the "ItemName" Would be the actual text that would be displayed on the Selector widget. So We gotta name it like how we'd want the players to see it :)

### Step 3: Add Your Clothing to a Collection (Wardrobe Groups)

#### 1. Open a Wardrobe Group Data Asset:

- /Game/SimpleSeries/Wardrobe/Clothes/Data/WardrobeGroups/DA\_(YourGroup)
- (If you made your own WardrobeGroup Data, don't forget to put it here) :



#### 2. Add your clothing:

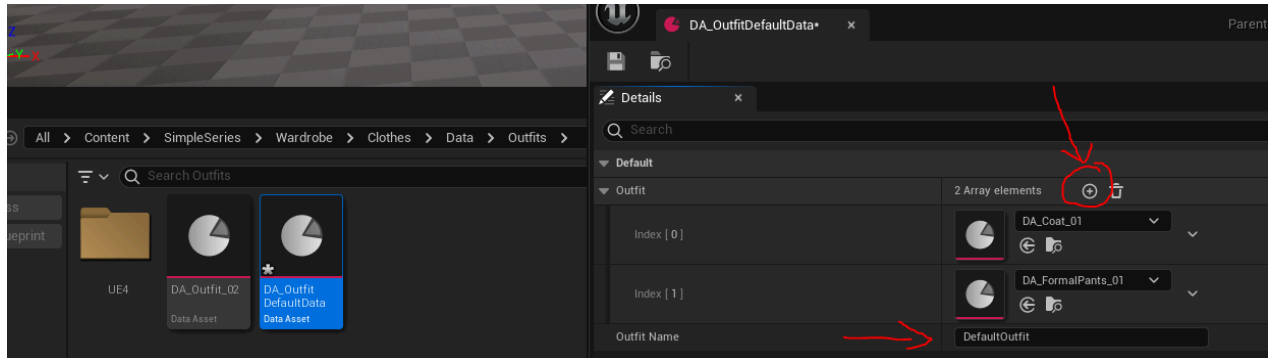
- Expand **ClothItems** → **ClothingItems** array
- Click + to add a new element
- Add your new `DA_YourClothingName` data asset

Your clothing will now appear in the wardrobe UI!

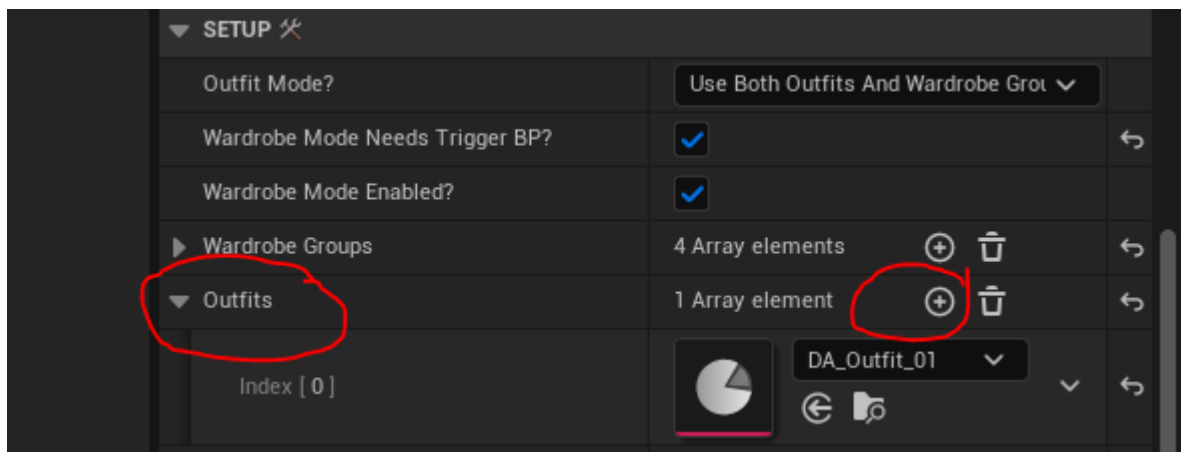
## PART 4: Creating Pre-Made Outfits

Outfits allow players to equip a complete set of clothing with one click.

1. **Navigate to:** /Game/SimpleSeries/Wardrobe/Clothes/Data/Outfits/
2. Duplicate one of the Data in there "DA\_OutfitDefaultData"



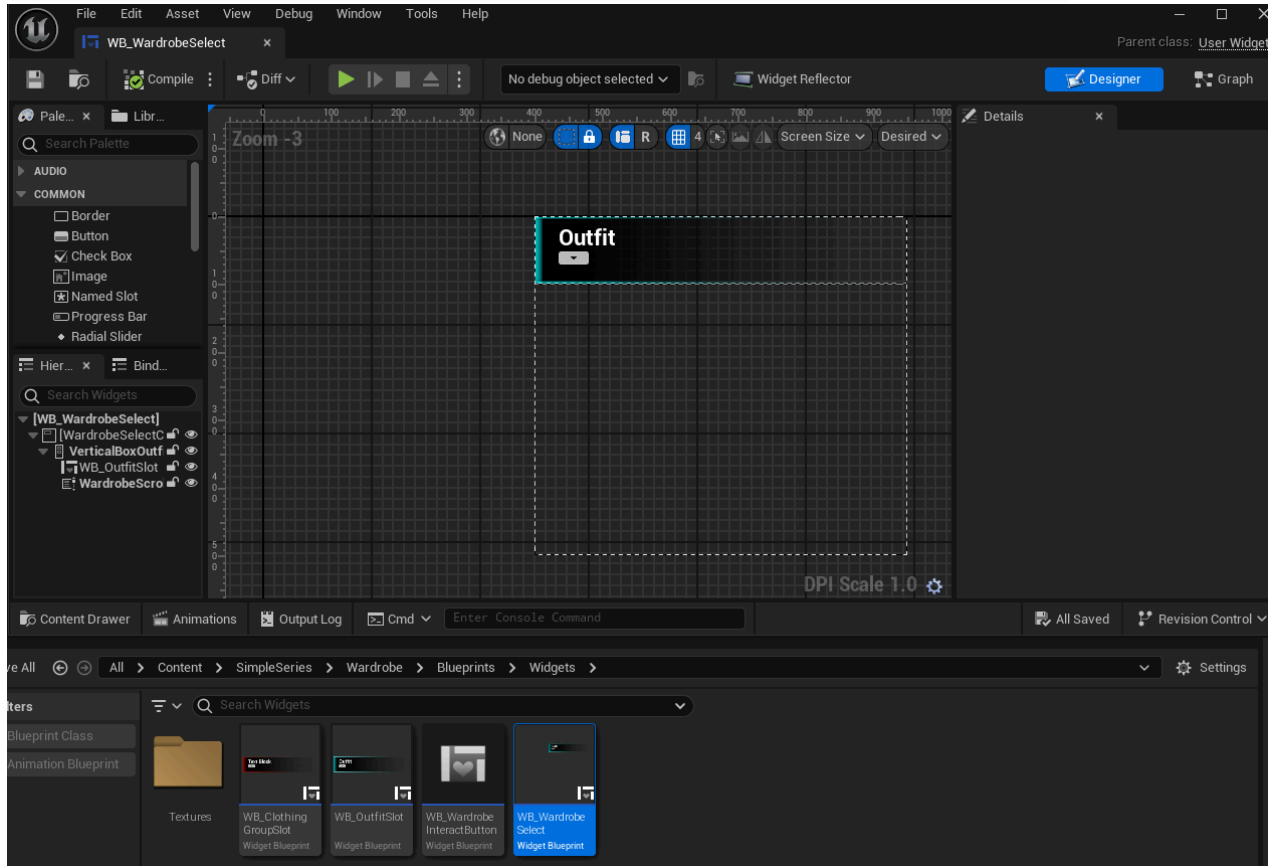
3. **Name it:** DA\_Outfit\_YourOutfitName (e.g., DA\_Outfit\_CasualLook)
4. Open it and add ClothPiece Data Assets (Click the plus icon to add more) you can also remove so you'd have less items. Just remember that when the outfit is selected, it will remove **every equipped ClothPiece** and replace it with *just whatever is inside this Outfit Data*
5. **Add the outfit to your character:**
  - Open your character blueprint
  - Find the AC\_SimpleWardrobe component
  - In the **Outfits** array, add your new outfit data asset



## PART 5: UI Overview

The wardrobe UI is located at:

- `/Game/SimpleSeries/Wardrobe/Blueprints/Widgets/WB_WardrobeSelect`



1. The displayed widget is **WB\_WardrobeSelect**
2. **Appearance:**
  - Change colors, fonts, sizes in the **Designer** tab
  - The UI uses a **ScrollBox** for browsing clothing items
  - Outfit buttons are at the top in a **VerticalBox**
3. **Key widgets:**
  - **WB\_ClothingGroupSlot**: Individual clothing item buttons (auto-generated)
  - **WB\_OutfitSlot**: Pre-made outfit buttons (auto-generated)

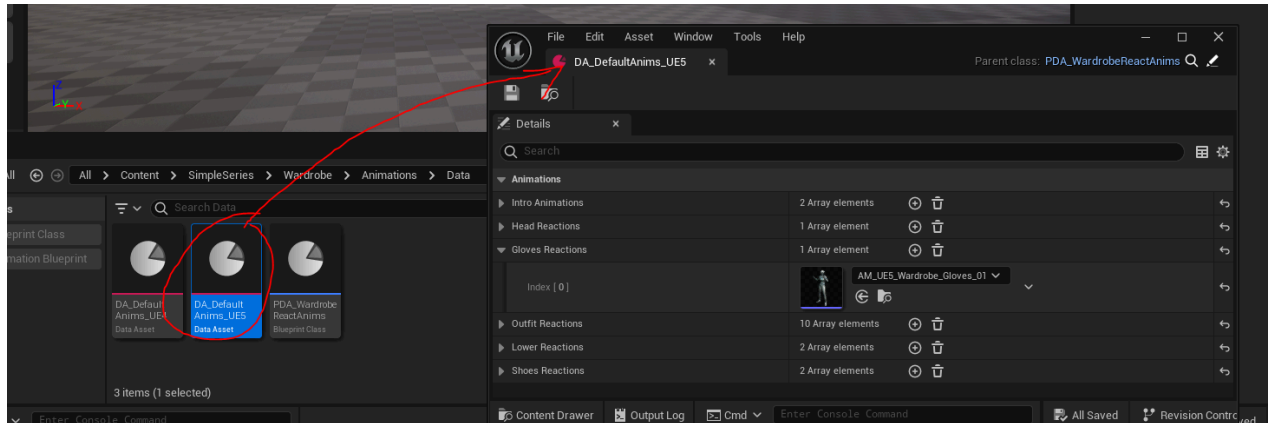
To change button textures:

- Navigate to: `/Game/SimpleSeries/Wardrobe/Blueprints/Widgets/Textures/`
- Replace `T_UiTexture_01` or `T_UIScrollThumb` with your own textures

## PART 6: Adding Character Reaction Animations

Characters can play animations when trying on clothes (e.g., a spin, pose, or happy reaction).

1. **Create or import your animation montages** (must be compatible with your character's skeleton)
2. **Navigate to:** /Game/SimpleSeries/Wardrobe/Animations/Data/
3. **Create a new Data Asset:**
  - Duplicate this Data Asset if you're on UE5, the left is for UE4



- Name it whatever you'd like to name it: DA\_MyReactionAnims
  - Open it and Start Adding your own Anim Montages, for each reaction type
4. **Assign to your character:**
    - Open your character blueprint
    - Select the AC\_SimpleWardrobe component
    - Set **ReactionAnims** to your new data asset

### Animation Tags:

The system automatically selects reactions based on clothing tags:

- head → Plays head-specific reaction
  - upper → Plays upper body reaction
  - lower → Plays lower body reaction
  - shoes → Plays shoes reaction
-

## PART 7: Integration with Your Game

### Saving/Loading Outfits

The wardrobe component stores the current outfit in:

- **CurrentlyEquippedOutfit** (array of PDA\_ClothPieceData)

To implement save/load:

1. **Create a Save Game object** that stores this array
2. **On save:**
  - Get reference to `AC_SimpleWardrobe` component
  - Copy the **CurrentlyEquippedOutfit** array to your save game
3. **On load:**
  - For each clothing piece in the saved array, call:  
`Change Cloth Piece (on AC_SimpleWardrobe)`

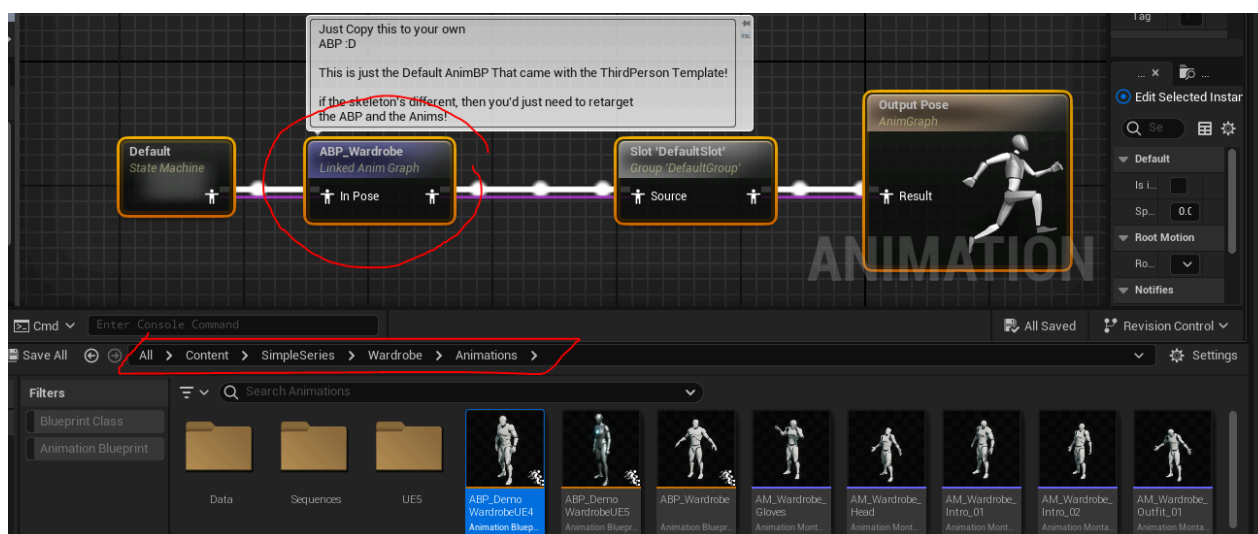
Personally I've never tried it, but theoretically, it should work, Just remember that the only variable to be concerned about by your save system is **CurrentlyEquippedOutfit**

Yea I know, it's only a wardrobe system and I'm not obligated to help with the save system. But naah, I gotta take into account for the possibility of expansion.

I want this project to be a solid building block for your game :D

## PART 8: Animation Blueprint

- Open your own ABP, and see where you last play the montage 'SlotDefaultSlot'
- put a "LinkedAnimGraph" Node just right before it Class (ABP\_Wardrobe) Just like this :D



## Networking (Multiplayer)

The wardrobe component is **not replicated by default**. To add multiplayer support:

1. **Make AC\_SimpleWardrobe replicated:**

- Open `AC_SimpleWardrobe`
- Set **Component Replicates** to `True`
- Mark **CurrentlyEquippedOutfit** as `Replicated`

2. **Call wardrobe functions on the server:**

- Wrap `Change Cloth Piece` calls in a **Server RPC**
- Use **Multicast** to update all clients

Well, it's really not that simple ofc, and no, I won't be responsible in helping you to make it replicated properly. This asset is currently intended for Single-Player only. But don't lemme stop you :D

## Randomizing Outfits

The component has a built-in randomization feature, And you can use this feature for NPCs, or Other Supporting Character. Not loading the whole wardrobe system would save on performance. :

### At runtime:

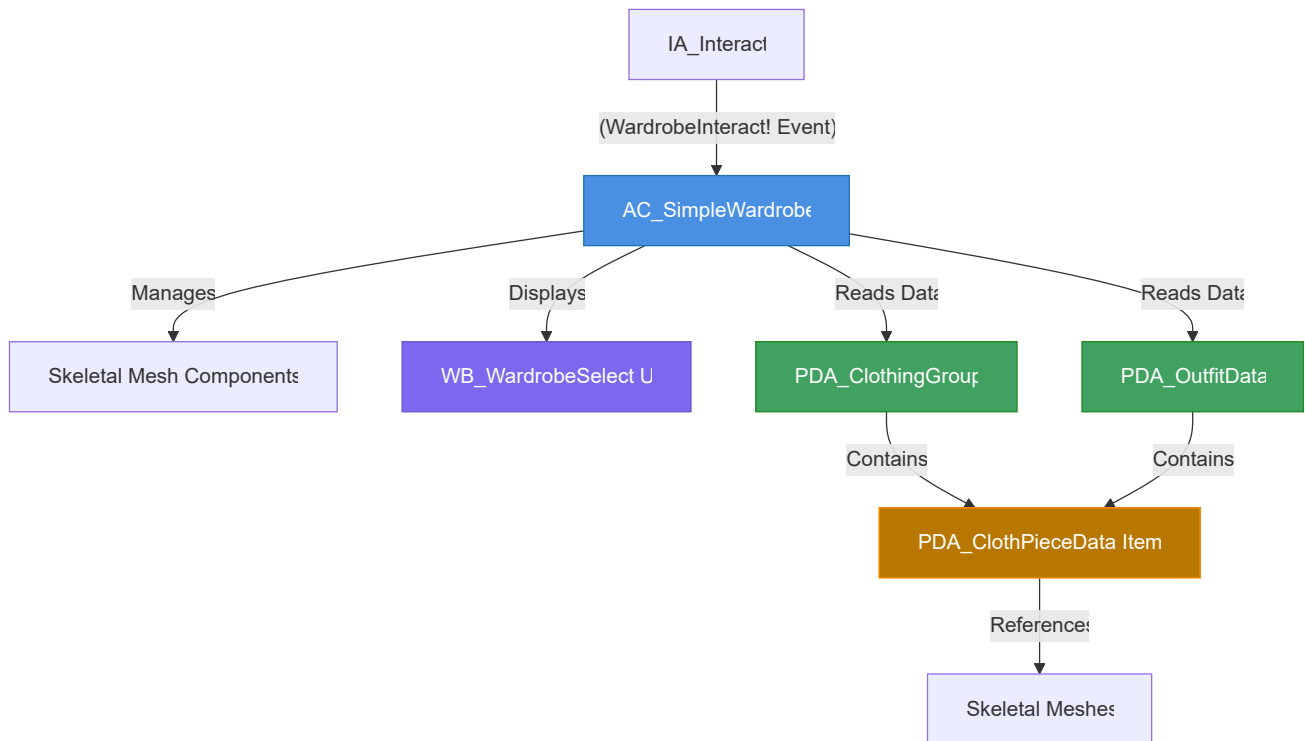
- Call the custom event: `Randomize Outfit` (on `AC_SimpleWardrobe` component)

### On character spawn:

- Set **RandomizeOutfitOnSpawn?** to `True` in the component properties
-

# 🎨 System Architecture (For Developers)

## Core Components:



## Data Flow:

1. **Trigger calls** `Wardrobe Interact!` on `AC_SimpleWardrobe`
  2. **Component spawns** `WB_WardrobeSelect` UI widget
  3. **UI reads** clothing from `Wardrobe Groups` and `Outfits` arrays
  4. **Player clicks** a clothing item button
  5. **UI calls** `Change Cloth Piece` on `AC_SimpleWardrobe`
  6. **Component updates** the appropriate skeletal mesh component
  7. **Component plays** a reaction animation based on the clothing tag
-

## Technical Reference

### AC\_SimpleWardrobe Component API

Function/Event	Description
Wardrobe Interact!	Opens/closes the wardrobe UI
Change Cloth Piece	Equips a single clothing item
Remove Cloth Piece	Removes a clothing item from a slot
Randomize Outfit	Equips a random combination of clothing
Update Outfit	Refreshes the character's appearance
Enable Wardrobe Mode	Activates wardrobe camera and input
Close Wardrobe Mode	Deactivates wardrobe mode
Character React Montage	Plays a specific reaction animation

### Data Asset Structure:

**PDA\_ClothPieceData** (Individual clothing item):

```
├─ ItemName (FName): Display name
├─ ClothPieceItem (S_ClothPiece struct):
│   └─ SkeletalMesh: The 3D model
│   └─ DesignatedTag: "head", "upper", "lower", or "shoes"
```

**PDA\_ClothingGroup** (Collection of clothing):

```
└─ ClothItems (S_ClothItems struct):
    └─ ClothingItems (Array): List of PDA_ClothPieceData assets
```

**PDA\_OutfitData** (Complete outfit):

```
├─ OutfitName (FName): Display name
└─ Outfit (Array): 4 PDA_ClothPieceData assets (one per slot)
```

---



## ? Troubleshooting

### Problem: Wardrobe UI doesn't appear

#### Solutions:

- Ensure `WardrobeModeEnabled?` is `True` on the `AC_SimpleWardrobe` component
- Check that `Wardrobe Groups` array is populated
- If using `BP_WardrobeTrigger`, ensure `WardrobeModeNeedsTriggerBP?` is `True`

### Problem: Clothing doesn't appear on character

#### Solutions:

- Verify skeletal mesh components are properly Tagged!, and should match with the "DesignatedTag" of the `ClothPiece`. Click on the `SKM` component, and add the matching tag on its `ComponentTags` variable
- Check that **Leader Pose Component** is set to the main character mesh
- Verify clothing meshes use the same skeleton as your character

### Problem: Clothing is misaligned or floating

#### Solutions:

- The clothing mesh and character mesh must share the **same skeleton**
- Re-import clothing with correct skeleton reference
- Check bone weights in your 3D modeling software

### Problem: Animations don't play

#### Solutions:

- Verify **ReactionAnims** data asset is assigned
- Ensure animation montages use the correct skeleton
- Check that the character's Animation Blueprint allows montage playback
- Confirm `Character React Montage` event is being called

### Problem: Wardrobe trigger doesn't respond

#### Solutions:

- Check collision on the `TRIGGER` (`SphereComponent`)
- Ensure player character has a collision channel that overlaps with the trigger
- Verify input action `IA_Interact` is bound correctly
- Check that Enhanced Input system is enabled in your project
- or maybe you've just set the "UseWardrobeTrigger?" bool to false?

---

## Recommended Practices

1. **Organize your clothing by category** - Sure helps us handling tons of assets
  2. **Optimize skeletal meshes** - Remove unnecessary bones to improve performance
  3. **Use LODs** - Add Level of Detail meshes for clothing to improve performance
  4. NPCs should have Wardrobe Mode disabled
- 

## Advanced Customization

### Adding New Body Slots (WardrobeGroup)

To add slots beyond head/upper/lower/shoes (e.g., gloves, accessories):

1. Add a new skeletal mesh component to your character
2. Give it a unique component tag (e.g., "gloves")
3. Create ClothPiece Items for the new slot, and make sure the "DesignatedTag" matches
4. Create a new DA\_ClothingGroup (e.g., "DA\_GlovesCollection") for the new category

### Integrating with Shopping Systems

To add purchasing mechanics:

1. Add a "Price" variable to `PDA_ClothPieceData`
2. Before calling `Change Cloth Piece`, check if the player can afford it
3. Deduct currency and mark item as "owned" in your save system
4. Filter the wardrobe UI to only show owned items

### Color/Material Customization

To allow players to change clothing colors:

1. Create Material Instances for your clothing meshes with color parameters
2. Add a color picker widget to the UI
3. When color is changed, use `Set Vector Parameter Value` on the clothing's material
4. Store selected colors in your save system

**OR!!** hear me out here, if you want speed, just duplicate the SKM asset and change its material

---

# **Support & Resources**

## **Included Documentation:**

- This comprehensive guide
- Demo level with example setup
- Commented Blueprint graphs

## **Project Assets:**

- UE4 and UE5 mannequin-compatible clothing examples
- MetaHuman setup example
- Fully functional UI widgets
- Example animations

## **Customization Freedom:**

- Full source Blueprints (no compiled code)
  - Modular, component-based architecture
  - Data-driven design for easy content addition
- 

## **You're Ready to Go!**

The Wardrobe System is designed to be both **easy for beginners** to use out-of-the-box and **flexible for advanced developers** to extend. Whether you're building a fashion game, an RPG with equipment, or a social simulation, this system provides a solid foundation.

## **Start by:**

1. Playing the demo level to understand the user experience
2. Following the "Adding Wardrobe to Your Character" guide
3. Creating your first custom clothing item
4. Experimenting with the settings to match your game's style

Thanks for Reading!, this doc took me hours (T\_T), even with the help of AI  
I actually prefer making vids over this :D  
Check out my [Youtube Channel](#) for Unreal Tutorials!