Skins Shop

Introduction

This documentation outlines the implementation of a skin customization system using Streamlit(an import for simple frontend development). The system allows users to interactively select and customize skins for a base character, with the ability to create and manage custom skins.

Key Components

1. **Base Skins**: Pre-defined legendary, seasonal, and rare skins for the base character.
2. **Custom Skins**: User-created skins that can be saved and loaded.
3. **UI Components**:
   - Base Character Image Display
   - Skin Selection Dropdown
   - Selected Skin Display
   - Custom Skin Creation Interface

1. 1. **Base Skins**: Pre-defined legendary, seasonal, and rare skins for the base character.
2. 2. **Custom Skins**: User-created skins that can be saved and loaded.
3. 3. **UI Components**:
   - ◦ Base Character Image Display
   - ◦ Skin Selection Dropdown
   - ◦ Selected Skin Display
   - ◦ Custom Skin Creation Interface

Implementation Details

Loading Assets and Data

```
import streamlit as st
from skin_module import LegendarySkin, SeasonalSkin, CustomSkin
from storage_module import load_custom_skins, save_custom_skin

# Load base character image
base_character = "assets/character_base_2.png"

# Define base skins
base_skins = [
    LegendarySkin("Dragon Blaze", "Legendary", 20, "assets/skin_
    SeasonalSkin("Snowflake", "Epic", 15, "assets/skin_snowflake
    CustomSkin("Rainbow", "Rare", 10, "assets/skin_rainbow_2.png
]

# Load and combine base skins with custom skins
custom_skins = load_custom_skins()
all_skins = base_skins + custom_skins
```

Streamlit UI Setup

```
st.title("Dynamic Skin Customization System")

# Display base character
st.subheader("Base Character")
st.image(base_character, caption="Base Character", use_column_wi

# Initialize variables
selected_skin = None
```

Dynamic Skin Selection

```
# Create dynamic skin names list
skin_names = ["None"] + [skin.name for skin in all_skins]
```

```
# Create selectbox for skin selection
selected_skin_name = st.selectbox("Choose a skin to apply:", ski

# Handle skin selection
if selected_skin_name != "None":
    selected_skin = next(skin for skin in all_skins if skin.name
    st.subheader(f"Selected Skin: {selected_skin.name}")
    st.image(selected_skin.image, caption=selected_skin.display_
    st.text(selected_skin.display_info())
```

Creating Custom Skins

```
# Set up custom skin creation interface
st.subheader("Create a Custom Skin")
base_skin_name = st.selectbox("Choose a base skin:", [skin.name
custom_name = st.text_input("Enter a name for your custom skin:"
custom_options = st.text_area("Enter customization options (comm

# Add button to add new options dynamically
add_option_button = st.button("Add New Option")

# Add button to remove last option
remove_option_button = st.button("Remove Last Option")

# Handle adding new options
if add_option_button:
    if custom_options.strip():
        new_option = custom_options.strip()
        skin_names.append(new_option)
        custom_options = ""
        st.experimental_rerun()

# Handle removing last option
if remove_option_button:
    if len(skin_names) > 1:
        skin_names.pop()
```

```
        st.experimental_rerun()

# Save custom skin
if st.button("Save Custom Skin"):
    if custom_name and custom_options:
        base_skin = next(skin for skin in base_skins if skin.nar
        customization_list = [opt.strip() for opt in custom_opti
        save_custom_skin(
            custom_name,
            base_skin.rarity,
            base_skin.base_price,
            base_skin.image_path,  # Use the stored image path
            customization_list,
        )
        st.success(f"Custom skin '{custom_name}' saved! Reload t
    else:
        st.error("Please provide a name and customization optio
```

Best Practices and Considerations

1. **Error Handling**: Implement robust error handling for various scenarios, including invalid inputs and database operations.

1. 1. **Error Handling**: Implement robust error handling for various scenarios, including invalid inputs and database operations.

1. **Performance Optimization**: For large numbers of skins, consider implementing pagination or lazy loading techniques.

1. 1. **Performance Optimization**: For large numbers of skins, consider implementing pagination or lazy loading techniques.

1. **Accessibility**: Ensure all UI elements are accessible and follow Web Content Accessibility Guidelines (WCAG).

1. 1. **Accessibility**: Ensure all UI elements are accessible and follow Web Content Accessibility Guidelines (WCAG).

1. **Testing**: Implement comprehensive unit tests and integration tests to ensure reliability.

1. 1. **Testing**: Implement comprehensive unit tests and integration tests to ensure reliability.

1. **Documentation**: Maintain clear documentation for both users and developers who may interact with the system.

1. 1. **Documentation**: Maintain clear documentation for both users and developers who may interact with the system.

1. **Security**: Implement proper authentication and authorization mechanisms to protect sensitive data and actions.

1. 1. **Security**: Implement proper authentication and authorization mechanisms to protect sensitive data and actions.

Future Enhancements

1. **Skin Preview**: Implement a preview feature for custom skins before saving.

1. 1. **Skin Preview**: Implement a preview feature for custom skins before saving.

1. **Sorting Options**: Add sorting capabilities for skin lists based on various criteria (e.g., rarity, price, popularity).

1. 1. **Sorting Options**: Add sorting capabilities for skin lists based on various criteria (e.g., rarity, price, popularity).

1. **Batch Operations**: Allow users to select multiple skins for simultaneous application or management.

1. 1. **Batch Operations**: Allow users to select multiple skins for simultaneous application or management.

1. **Social Features**: Integrate social sharing options for custom skins.

1. 1. **Social Features**: Integrate social sharing options for custom skins.

1. **Analytics**: Implement usage analytics to understand popular skin trends and preferences.

1. 1. **Analytics**: Implement usage analytics to understand popular skin trends and preferences.

Conclusion

This dynamic skin customization system provides a flexible and interactive interface for users to explore, create, and manage skins for a base character. By leveraging Streamlit's powerful features and following best practices, the system offers a seamless experience for both casual users and power users alike.