

University of Information Technology



2024-2025 Academic Year

CST-5105: Artificial Intelligence

Fashionista

Submitted by

Group-4

Date:17.09.2024

## Members

Shun Lai Oo (b9 - 1803)

Pyae Pyae Thu (b9 - 1813)

Hnin Shwe Yi Wint (b9 - 1853)

Myat Shwe Yi Moe (b9 - 1861)

Kyi Sin Shun Lett (b9 - 1903)

## Roles and responsibilities

Shun Lai Oo Analysis + Knowledge base

Pyae Pyae Thu Knowledge base + User Interface

Hnin Shwe Yi Wint User interface+ Knowledge base

Myat Shwe Yi Moe Personalize + Knowledge base

Kyi Sin Shun Lett Analysis +Knowledge base



# Contents

- Introduction
- Objectives
- Requirements
- Algorithm and Technique
- System flow diagram
- System design
- Current limitation
- System implementation
- conclusion
- Further extention
- References

## Introduction

- In the modern era of technology, artificial intelligence (AI) has revolutionized various fields, including fashion.
- The "Fashionista" project harnesses the power of AI to transform how individuals select and coordinate their outfits.
- This project integrates Prolog, a logic programming language, with PHP, a server-side scripting language, to deliver a sophisticated fashion advisory system.

## Objective

- To provide personalized outfit recommendations
- To provide convenience
- To make fashion accessible, fun, and stress-free
- To cater to a diverse range of body types, skin tones, and style preferences, promoting inclusivity in fashion.

## Requirements

**Hardware Requirement** - 64-bit operating system, x64-based processor

**Software Requirement** - Visual Studio (window 11), SWI-Prolog, XAMPP

# Algorithm and techniques

## Heuristic Evaluation:

- By sorting items based on match count, applying a heuristic approach to prioritize items with the highest number of matches. In AI, heuristics are used to make decisions or rank options based on certain criteria.

% Helper rule to sort matches by count (highest to lowest)

sort\_matches\_by\_count(Matches, SortedMatches) :-

sort(2, @>=, Matches, SortedMatches). % Sort by the second element (MatchCount) in descending order

% Helper rule to remove duplicates based on item

remove\_duplicates(List, UniqueList) :-

sort(1, @<, List, UniqueList). % Sort by the first element (Item) to remove duplicates

# Algorithm and techniques

a  
snippet  
of  
our  
knowledge  
base  
in  
prolog

```
clothing('Pleated Skirt', 'Elegant', 'Adult', 'Pear', 'Heart', '#ae703a', 'Female', 'Bottom', 'image/pleated_skirt.png').
clothing('Silk Blouse', 'Elegant', 'Adult', 'Rectangle', 'Heart', '#e9b885', 'Female', 'Top', 'image/silk_blouse.jpg').
clothing('Maxi Skirt', 'Elegant', 'Adult', 'Hourglass', 'Round', '#ad6c44', 'Female', 'Bottom', 'image/maxi_skirt.png').
clothing('Casual Slim Fit Pants', 'Casual', 'Adult', 'Pear', 'Square', '#ae703a', 'Male', 'Bottom', 'image/casual_slim_fit_pants.png').
clothing('Casual Jogging Set', 'Casual', 'Adult', 'Inverted Triangle', 'Heart', '#ad6c44', 'Male', 'Set', 'image/casual_jogging_set.png').
clothing('Casual T-Shirt', 'Casual', 'Adult', 'Rectangle', 'Oval', '#f5c1a2', 'Male', 'Top', 'image/casual_tshirt.png').
clothing('Bohemian Blouse', 'Casual', 'Adult', 'Pear', 'Oval', '#30150e', 'Female', 'Top', 'image/bohemian_blouse.jpg').
clothing('Patterned Scarf', 'Casual', 'Adult', 'Pear', 'Oval', '#30150e', 'Female', 'Top', 'image/patterned_scarf.png').
clothing('Bohemian Blouse', 'Casual', 'Adult', 'Pear', 'Oval', '#30150e', 'Female', 'Top', 'image/bohemian_blouse.jpg').
clothing('Patterned Scarf', 'Casual', 'Adult', 'Pear', 'Oval', '#30150e', 'Female', 'Top', 'image/patterned_scarf.png').
clothing('Cute Casual Shorts', 'Cute', 'Teenager', 'Oval', 'Heart', '#dfb175', 'Male', 'Bottom', 'image/cute_casual_shorts.png').
clothing('Cute Denim Jeans', 'Cute', 'Teenager', 'Heart', 'Diamond', '#ae703a', 'Male', 'Bottom', 'image/cute_denim_jeans.png').
clothing('Cute Cargo Pants', 'Cute', 'Teenager', 'Square', 'Round', '#ad6c44', 'Male', 'Bottom', 'image/cute_cargo_pants.png').
```

# Algorithm and techniques

a  
snippet  
of  
our  
knowledge  
base  
in  
prolog

```
suitable_color('#dfb175', 'Warm shades such as coral, orange, and earth tones').  
  
suitable_color('#ae703a', 'Rich colors like olive green, brown, and mustard yellow').  
  
suitable_color('#ad6c44', 'Deep hues such as burgundy, navy blue, and dark green').  
  
unsuitable_color('#dfb175', '#30150e').  
  
unsuitable_color('#ae703a', '#e9b885').  
  
unsuitable_color('#ae703a', '#ad6c44').  
  
  
haircut_recommendation(male, oval, 'Pompadour').  
  
haircut_recommendation(male, oval, 'Quiff').  
  
haircut_recommendation(male, square, 'Buzz Cut').  
  
haircut_recommendation(male, square, 'Undercut').  
  
haircut_recommendation(female, round, 'Bob Cut').
```

# Algorithm and techniques

a  
snippet  
of  
our  
knowledge  
base  
in  
prolog

```
find_items_by_gender_and_body_type(Gender, BodyType, Items) :- findall(Item,  
    clothing(Item, _, _, BodyType, _, _, Gender, _, _), Items).  
  
clothing_items_for_body_type(BodyType, Items) :-  
    findall(Item, clothing(Item, _, _, BodyType, _, _, _, _), Items).  
  
recommend_clothing(Style, AgeGroup, BodyType, FaceShape, SkinTone, Gender, Category, Item, ImagePath) :-  
    clothing(Item, Style, AgeGroup, BodyType, FaceShape, SkinTone, Gender, Category, ImagePath),  
    format('Item(~w, ~w).~n', [Item, ImagePath]).  
  
count_matches([], [], 0).  
count_matches([X|T1], [X|T2], N) :-  
    count_matches(T1, T2, N1),  
    N is N1 + 1.  
count_matches([_|T1], [_|T2], N) :-  
    count_matches(T1, T2, N).  
  
suitable_colors_for_skin_tone(SkinTone, SuitableColors) :-  
    findall(ColorDesc, suitable_color(SkinTone, ColorDesc), SuitableColors).
```

# Algorithm and techniques

a  
snippet  
of  
our  
knowledge  
base  
in  
prolog

```
recommend_clothing_min1(Style, AgeGroup, BodyType, FaceShape, SkinTone, Gender, Category) :-  
    % Filter clothing items based on selected Gender and Category  
    findall([Item, ImagePath, MatchCount],  
        (clothing(Item, StyleDB, AgeGroupDB, BodyTypeDB, FaceShapeDB, SkinToneDB, GenderDB, CategoryDB,  
            ImagePath),  
            StyleDB = Style,          % Match the Style exactly  
            AgeGroupDB = AgeGroup,    % Match the AgeGroup exactly  
            CategoryDB = Category,    % Match the Category exactly  
            (Gender = 'Other' ; GenderDB = Gender), % Match Gender if not 'Other'  
            count_matches([BodyType, FaceShape, SkinTone, Gender, Category],  
                [BodyTypeDB, FaceShapeDB, SkinToneDB, GenderDB, CategoryDB],  
                MatchCount),  
            MatchCount >= 1), % At least 1 feature matches (excluding Style and AgeGroup)  
        Matches),  
    remove_duplicates(Matches, UniqueMatches), % Remove duplicate items  
    sort_matches_by_count(UniqueMatches, SortedMatches), % Sort by match count  
    display_sorted_matches(SortedMatches). % Display the sorted matches
```

## System Design

### 1. Frontend Components (UI/UX)

Home Page:

Navigation Bar: Contains links for:

- Home
- Personalized (Fashion recommendations)
- Analysis (Results of fashion analysis based on user input)
- About (Information about Fashionista Website)

Interactive Elements:

- Dropdown menus for selecting options
- Image sliders for showing clothing items or suggestions.
- Buttons for navigation (e.g., "Personalized," "Analysis").

Home

Personalized

Analysis

About

Category:

Top

Top

Bottom

Set

Choose style:

Streetwear

Casual

Formal

Elegant

Cute

Streetwear

## System Flow Diagram

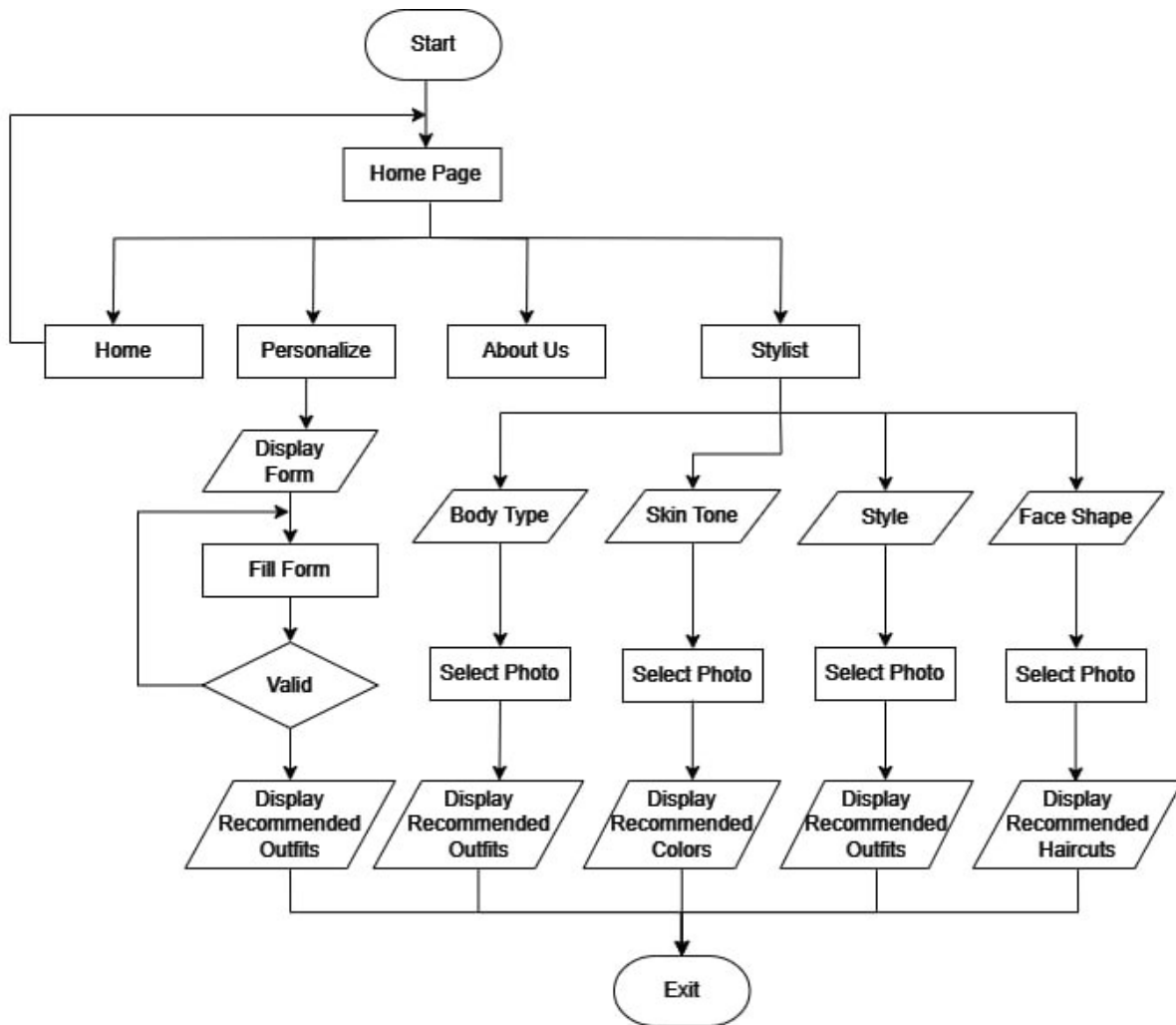


Figure 1: System Flow Diagram

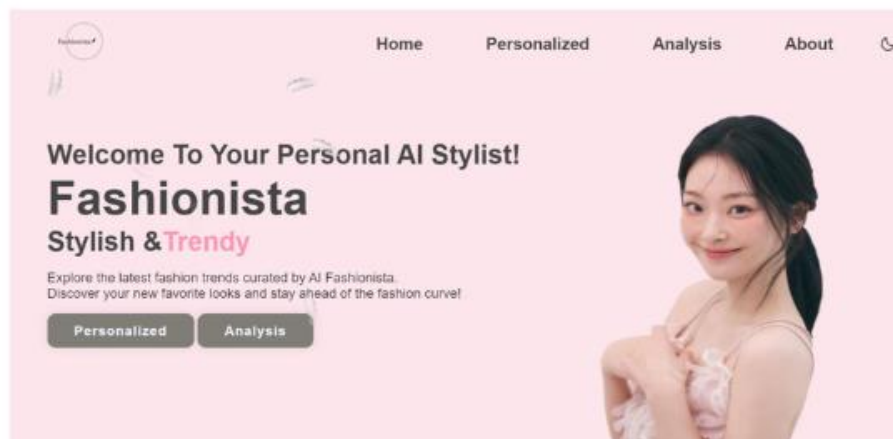


## Current Limitation

- Limited clothing knowledgebase size reduces recommendation variety.
- Simplified body type classifications may not reflect all user shapes.
- Skin tone options are restricted, limiting tailored recommendations.
- Gender categorization is binary, excluding non-binary preferences.
- Style recommendations are basic and may not fully capture user tastes.
- No real-time updates on fashion trends or new collections.
- Lack of interactive user feedback prevents personalized improvements.

## System Implementation

### Home page



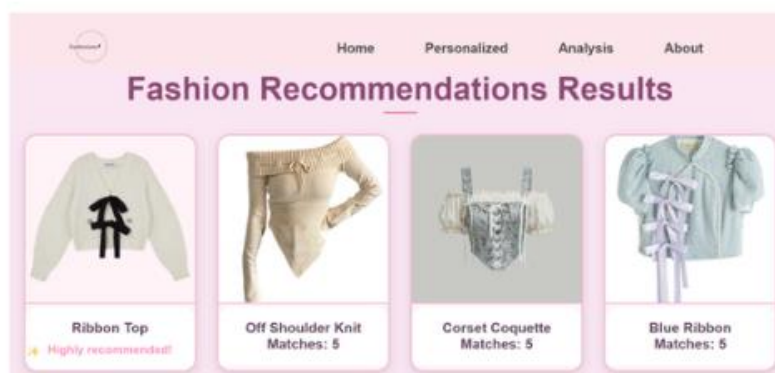
# System Implementation

## personalized form

A screenshot of a web application interface titled "Find Your Style". The interface has a pink header with navigation links: Home, Personalized, Analysis, and About. Below the header, the form is organized into several sections. At the top, there are two dropdown menus: "Style:" with "Casual" selected and "Age Group:" with "Teenager" selected. Below these are two rows of image-based selection options. The "Body Type:" row contains six icons of female torsos in various styles. The "Face Shape:" row contains six icons of face shapes. Below these rows are two more dropdown menus: "Gender:" with "Male" selected and "Category:" with "Top" selected. At the bottom of the form, there is a "Skin Tone:" section with five colored circles (light orange, orange, brown, dark brown, black) and a large pink button labeled "Get Recommendations".

# System Implementation


## personalized form result



# System Implementation

## analysis form

### Clothing Items



Select Gender and Style to Find Clothing Items


Choose style:

Casual

### Haircuts

Select Your Face Shape to Find Haircuts

Choose face shape:



Round Oval Square Heart Diamond

Choose gender:


Male

# System Implementation


## analysis form


### Skin Tone

Select Your Skin Tone to Find Your Personal Color



Find Colors

Skin Tone: 

For the skin tone , the following colors are considered suitable:


Soft pastels and light colors like beige and light blue.

These colors are carefully selected to enhance and complement your natural complexion. Incorporating these colors into your wardrobe will help you achieve a polished and harmonious look that aligns with your skin tone, making your appearance more vibrant and flattering.

### Body Type

Select Your Body Type to Find the Perfect Clothing Items

Choose body type:



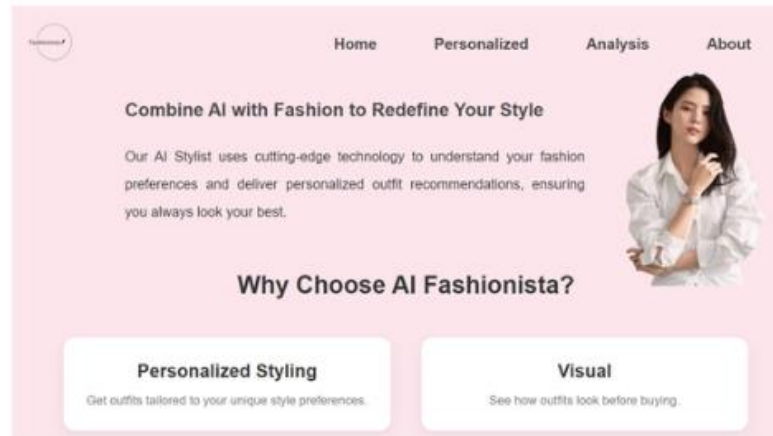
Rectangle Pear Inverted Triangle Hourglass Apple

Choose gender:

Male

# System Implementation

## About us



## Further Extension

- **Enhanced User Profiles:** Incorporating more detailed user profiles with preferences, body measurements, and style history to generate even more precise recommendations.
- **Machine Learning Integration:** Leveraging machine learning to analyze fashion trends and user feedback, improving the system's ability to suggest new and evolving styles.
- **Image Recognition:** Implementing image recognition to allow users to upload photos of their wardrobe items and receive mix-and-match suggestions.

## Conclusion

- **Prolog and PHP Integration:** The project leverages Prolog for developing a knowledge base of fashion rules and utilizes PHP to build a user-friendly web interface, facilitating seamless user interactions.
- **Personalized Recommendations:** By inputting individual preferences and attributes, users receive tailored fashion advice, enhancing their personal style choices.
- **Innovative Use of AI:** "Fashionista" showcases the application of artificial intelligence in fashion, setting a precedent for future technological advancements in the industry.

## Reference

<https://www.w3schools.com/>

<https://www.pinterest.com/>

<https://cliopatria.swi-prolog.org/tutorial/>