FINAL REPORT OF CETA

Fashion agent for
Clothing Editing and Try-on Application

Group Number: 16
Team Name: Code All Night

Rain all day, code all night

TEAM MEMBER

- Gong Xinyu A0295988W
- Liu Peilin A0291951X
- Su Chang A0296250A
- Wang Xinji A0265810H
- Zhang Wenqing A0296886Y

Note: The list is sorted alphabetically by the first letter of the participants' names.

Contents

FINAL REPORT

1 OVERVIEW

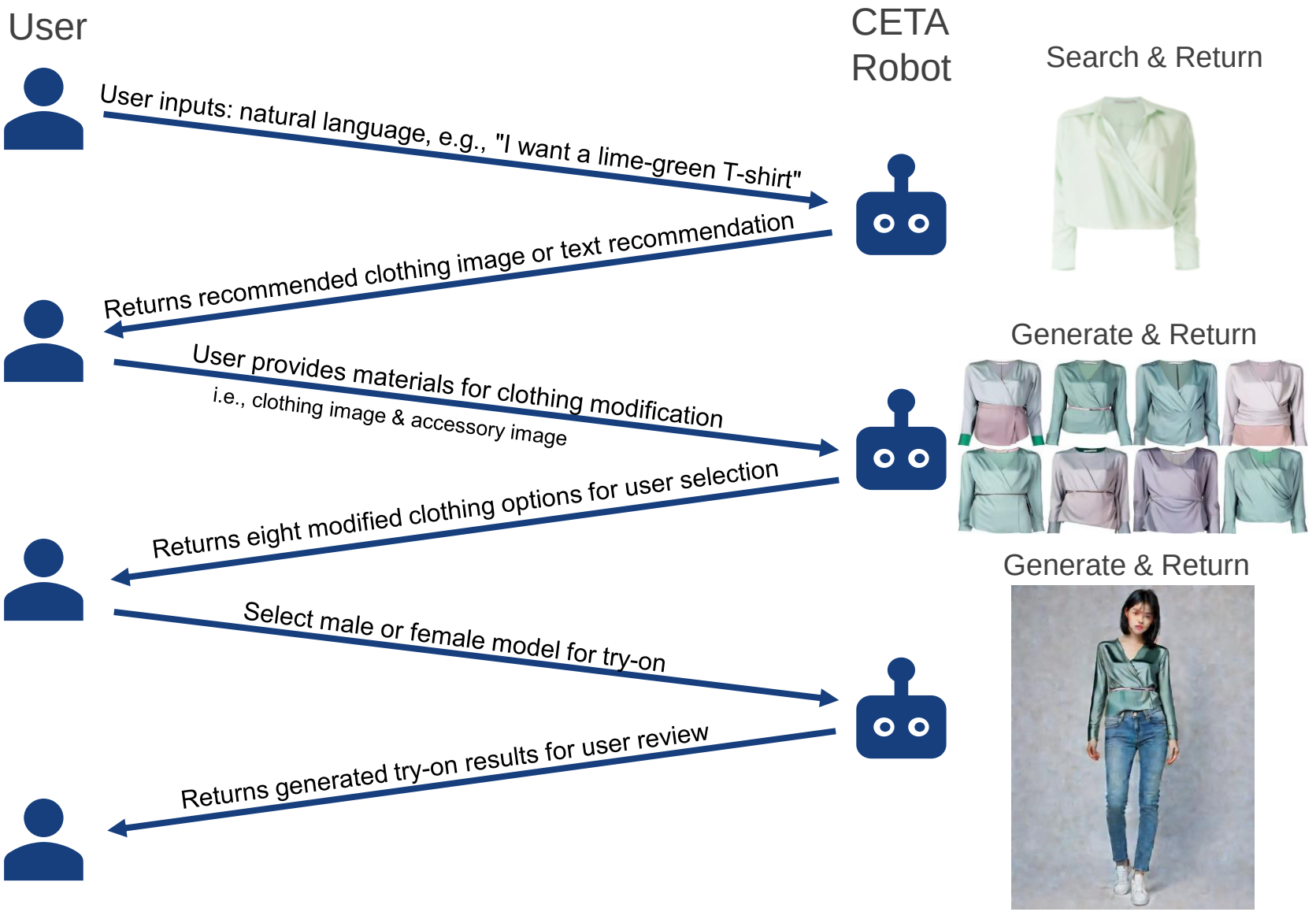
- What's CETA?
- Why CETA?
- How CETA?

What's CETA

➢ Introduction

- CETA is an AI-driven fashion assistant designed to provide personalized clothing recommendations, edit accessories on clothing, and offer virtual try-on experiences.
- Leveraging advanced AI models and sophisticated text analysis, CETA can achieve the following tasks:
  - Interpreting users' natural language inputs and suggesting tailored fashion options in text or image format.
  - Generating customized clothing designs based on user-provided images and specific clothing accessory requirements.
  - Providing model try-on image generation functionality.

What's CETA
Typical Usage Flow



Why CETA

Advantages of CETA :

- **Accurate Personalization in Recommendation:**
  CETA leverages NLP to deeply understand users' needs, providing more precise and tailored clothing recommendations in text or image format.
- **Virtual Try-On Experience**
  Allows users to preview outfits before purchasing, reducing return rates and enhancing shopping confidence.
- **Smart Interaction**
  Using a simple and intuitive interface through Telegram Bot, users can easily get suggestions and fitting results.

How CETA

Frontend :

- **Telegram Bot**: Acts as the interaction interface, allowing users to communicate using natural language or images.
- **Personalized Recommendations**: Provides customized clothing suggestions based on user input.

Backend :

- **SQLite Database:** Used to store clothing data and user interaction records.
- **Fashion Adapter Module:** Responsible for generating customized clothing designs.
- **CatVTON Module:** Used to generate virtual try-on effects.

By integrating multiple modalities, CETA provides users with a smooth and seamless interactive experience throughout the entire process.

2 MARKET RESEARCH

- Market Demand
- Target Customer Analysis
- Result of Market Research

Market Demand

Consumer Personalization Needs: Apparel shoppers now expect an enhanced shopping experience, including outfit recommendations tailored to rich, multi-dimensional user profiles that go beyond simple keyword matching and leverage natural language communication. They also seek tools to add accessories to clothing images or virtually try on outfits, helping them make more confident purchasing decisions.

Limitations in Flexibility of Virtual Try-On Platforms: Current image generation models and platforms, such as DALL-E and GPT-4o, primarily generate entirely new images rather than making modifications to existing ones, resulting in limited flexibility and seamless interaction for users.

Lagging Behind Innovation Speed: As fashion cycles accelerate, brands must respond to trends faster than ever. Traditional design and production processes require significant time and investment in hiring models for try-ons, which can be costly for designers with limited budgets. With AI, however, this cycle can be expedited, enabling designers to save both time and money.

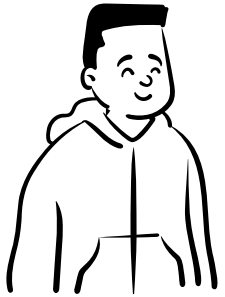
Target Customer Analysis

Online Shoppers

- They can use the **Virtual Try-on Model** to visualize how clothing fits themselves.
- They can also use **Garment Modifying Model** to add accessories to personalise their choice.
- The **LLM-based recommendation system** ensures more personalized and satisfying shopping experiences compared to traditional keyword-based searches.

Fashion Designers

- The **Garment Modifying Model** would help them instantly visualize design changes and get a clearer idea of how different components will work together.
- They can also use **Virtual Try-on Model** to generate virtual prototypes, speeding up the feedback loop between design and production.



E-Commerce Platforms

- Retailers can integrate the **Garment Modifying Model** and **Virtual Try-on Model** into their websites to offer customers a more personalised and immersive shopping experience.
- They can also integrate the **Recommendation Chat Tool** to better promote their clothing product, increasing customer satisfaction and loyalty.

Result of Market Research

➢ Questionnaire Survey

Research Objects

Online shopping enthusiasts
Students majoring in fashion design

Questionnaire content

- Shortcomings of existing platforms for clothing sector editors
- Personalized recommendation needs
- Virtual try-on experience

Survey Outcome

- More than 85% of consumers surveyed said they want to be able to use living language to get recommendations when shopping online, rather than matching based on simple keywords.
- 88% of respondents believe virtual fitting is a key feature to enhance the online shopping experience
- 92% expressed interest in AI-assisted apparel design that could help speed up design and provide more options.

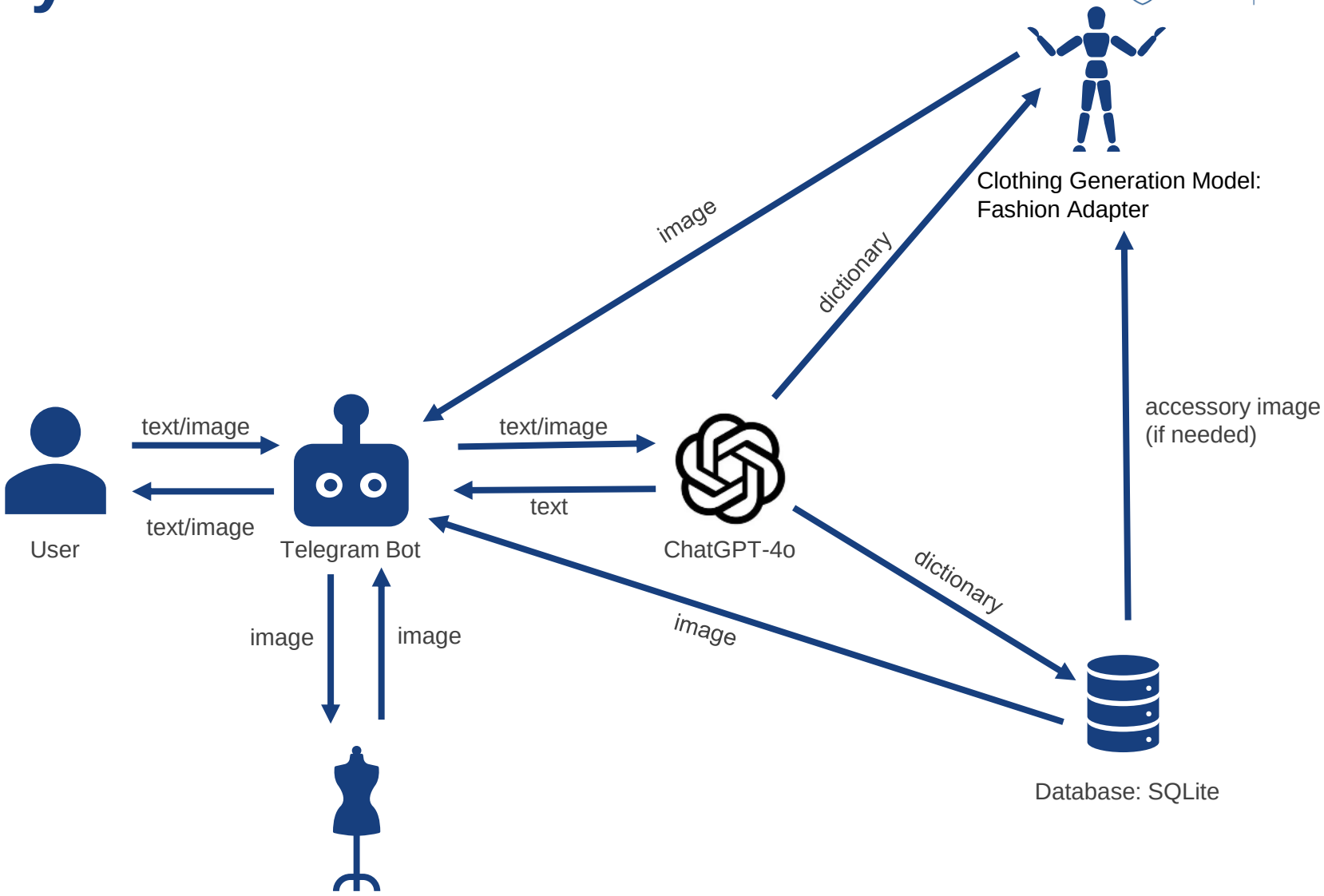
Analysis of Open Source Data

- According to *Global Fashion E-commerce market data*, the global online apparel market is expected to reach a total revenue of nearly $1.5 trillion by 2025, at a compound annual growth rate (CAGR) of 7.18%.
- A survey on virtual try-on technology reveals that the market penetration of the technology is increasing year on year and has reached about 18% utilisation in 2023, which is expected to grow significantly in the next three years.

3 SYSTEM DESIGN / MODEL

- System Architecture & Workflow Dependence
- Functional Description of Components
  - TelegramBot
  - Prompt for chatgpt API
  - database & databaseAPI
  - fashion_adapter
  - CatVTON

System Architecture



System Architecture

CETA system consists of five main components:

- TelegramBot
- Chatgpt
- Database
- Clothing Generation Model
- AI Virtual Try-on Model

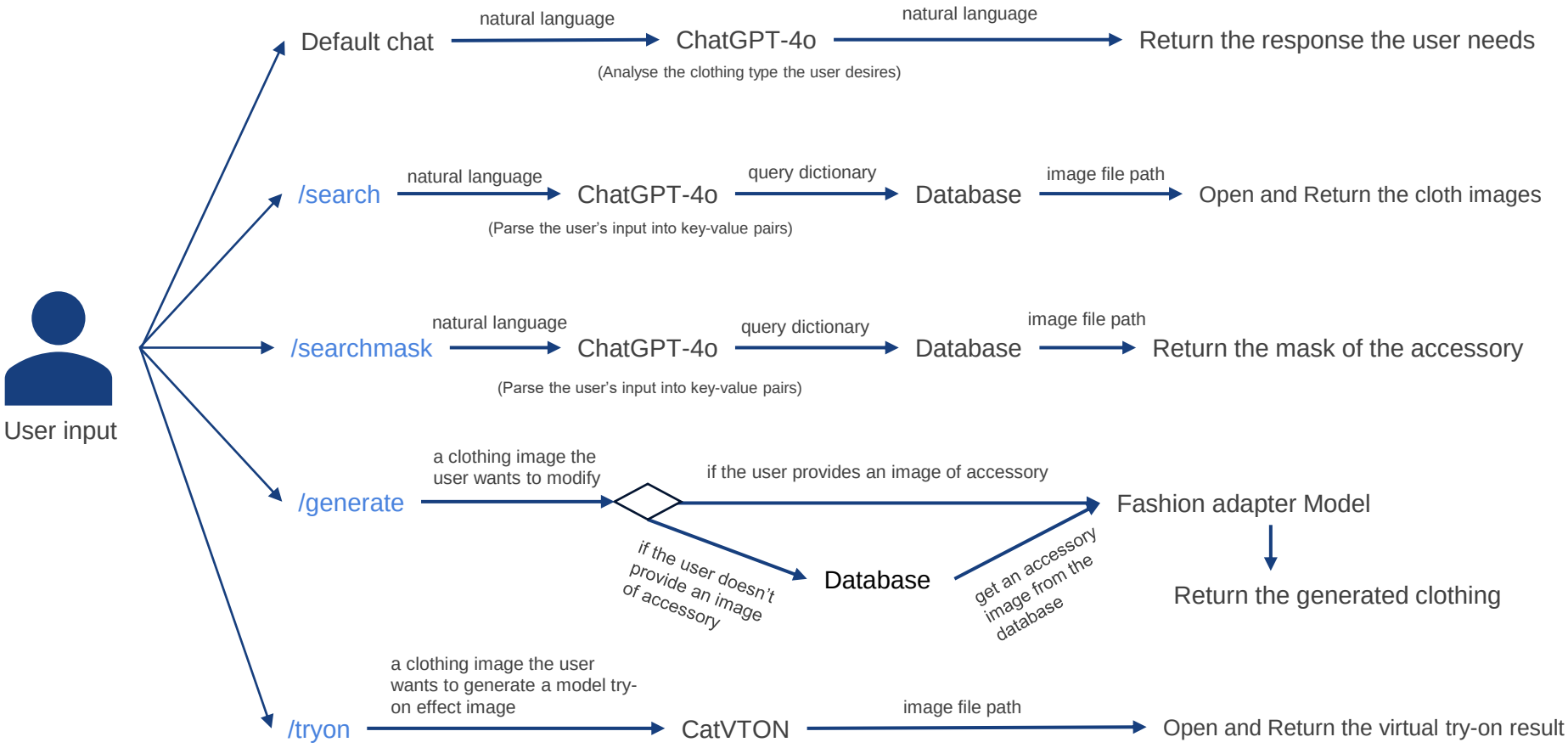
In the following sections, we will detail how each of these five components works

Workflow Dependence

The CETA bot will provide 5 functions, each function's workflow is shown below.

Function	Function Description	Input	Output
Default chat	Engage in a conversation with a fashion design assistant powered by ChatGPT	Text	Text
/search	Search for desired clothing through natural language	Text	Image
/searchmask	Search for the desired clothing accessory mask through natural language	Text	Image
/generate	Generate the desired clothing through an AI model	Image/Text	Image
/tryon	Generate virtual try-on images for clothing through an AI model	Image	Image

Workflow Dependence



Functional Description of Components

➢ TelegramBot

Direct Text Chat

Function: Allows users to interact with the bot for fashion design advice.

Description: Users can request clothing suggestions, style tips, or customization advice via text messages.

Commands

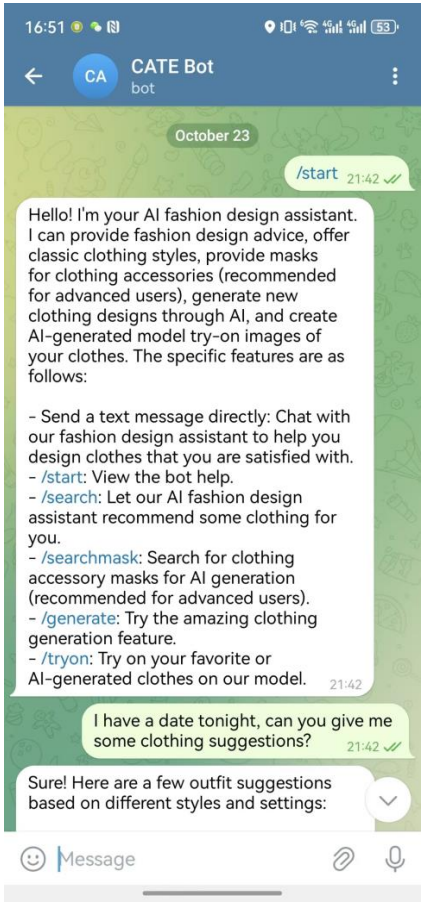
/start Initiates interaction and provides an overview of bot features.

/search Recommends clothing based on user preferences.

/searchmask Searches for accessory masks for AI design (for advanced users).

/generate Creates new clothing designs through AI.

/tryon Virtual try-on to preview clothing or designs on a model.



Functional Description of Components

➢ Prompt for Chatgpt API | why prompt engineering

Optimize user experience

Prompt guides GPT to generate personalized clothing recommendations, which allows users to get results that match their needs faster, and also reduces irrelevant information or lengthy replies, improving the efficiency of user interaction with the system.

Reduce hallucinations

By specifying input sources and data boundaries in the prompt, and explicitly signaling that the GPT will only return existing garment parts in the database, the GPT's ability to generate free-form recommendations can be limited to prevent the model from generating fabricated or unrealistic recommendations

Control the content of outputs

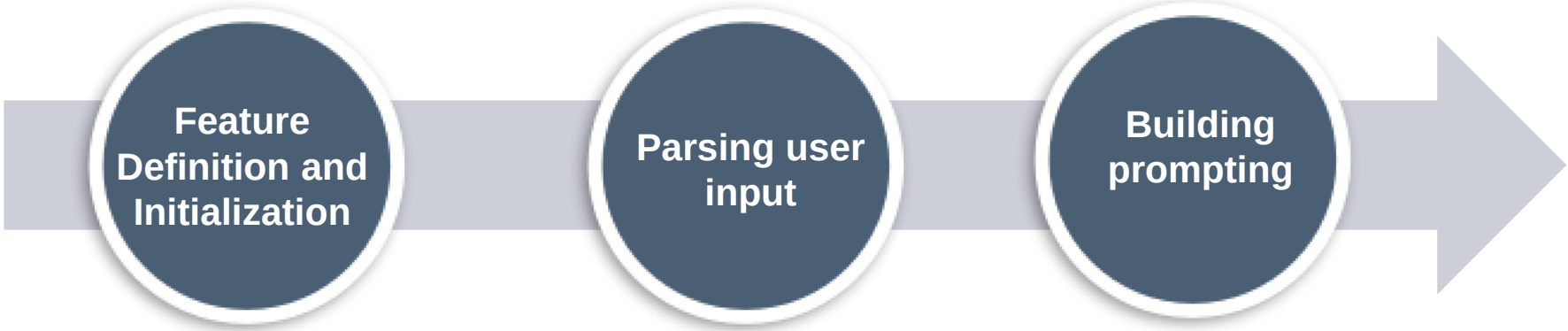
Restrict the GPT to output only garment features that are relevant to the user's needs and require that the results be returned to the database in a specific structure (dictionary). With explicit instructions, prompt can effectively minimize the generation of irrelevant or invalid content and ensure the accuracy and consistency of the output.

User guidance and contextualization

By embedding suggestions within responses, GPT can not only answers user queries but also introduces relevant options, such as customizing components or exploring accessories. Contextualization ensures consistency across interactions, maintaining a seamless flow and encouraging users to discover additional functionalities.

Functional Description of Components

➢ Prompt for Chatgpt API | interaction flow



Define a list of features that serves as the basis for feature parsing, recognizing and using only these predefined features when parsing user input.

The user inputs a natural language description, which is parsed by invoking the GPT model to convert the user input into a dictionary format containing the desired features.

Describe the optional values of each feature clearly so that the model is consistent with the predefined features when parsing.

Functional Description of Components

➢ Prompt for Chatgpt API | interaction flow

"I want a simple green long dress for the night party."

Dictionary generation rely on defined garment features and attributes

"category": "dress"
"color": "green"
"sleeve_length": "long"
"style": "simple"
"suitable_age": "youth"
"season": "summer"

Search for recommended clothing according to the dictionary

After user receive the original recommendation：

You may choose to add one of the following accessories:
- belt
- body_piece
- button
- collar
- cuff
- hood
- pocket
- waist_band
- placket

"Please add a belt."
(Users can also upload images directly)

"category": "dress"
"accessory": "belt"

Search for recommended cloth according to the dictionary. Look up the dress and its mask according to the new dictionary, and extract sector through the model for subsequent editing

The example of features and attributes defined in prompt part:

category
Hoodie
leather skirt
leather pants
polo shirt
dress
...

season
summer
winter
spring and autumn
...

Functional Description of Components

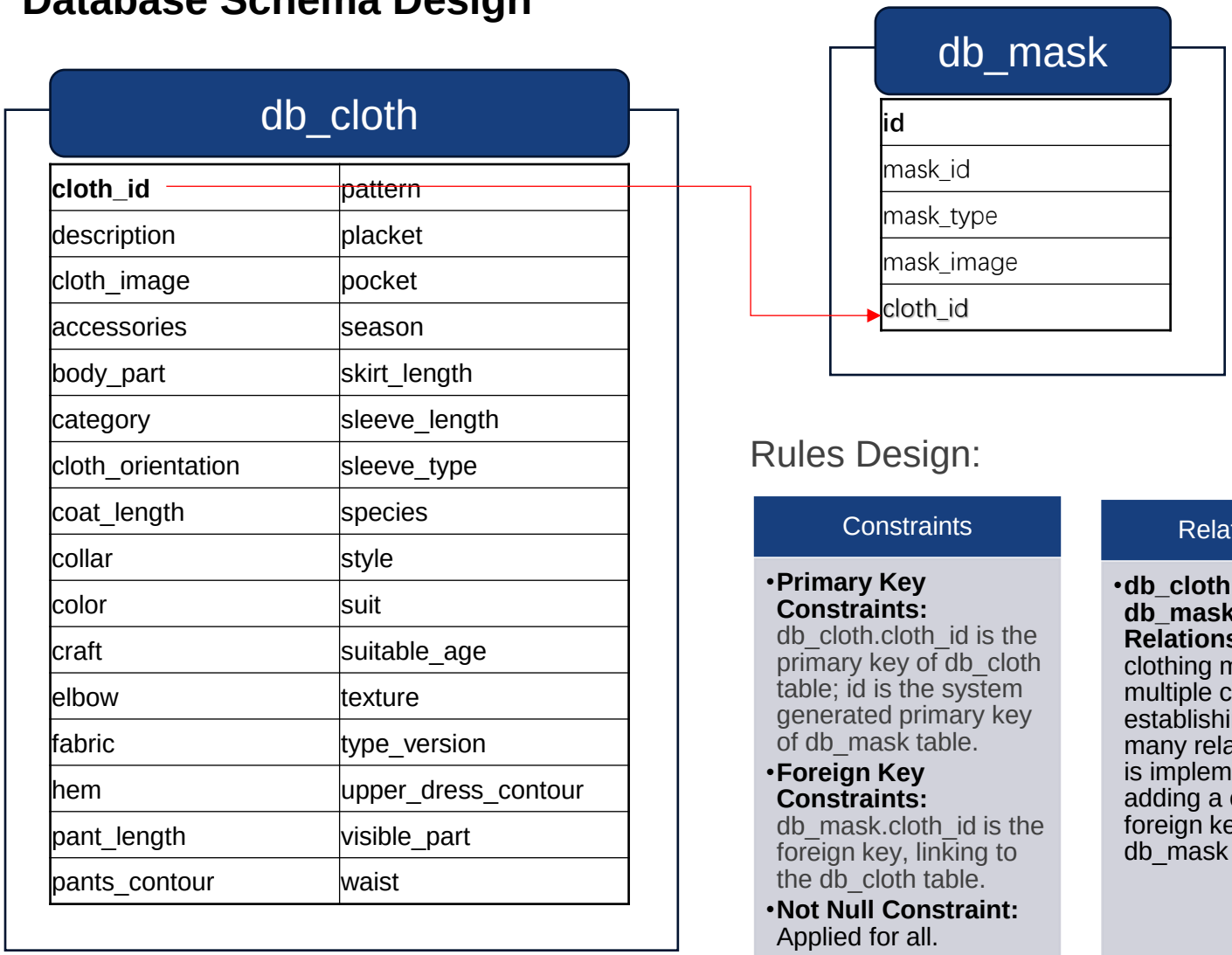
➢ database & databaseAPI

Database Type and Technology Selection

- **Reasons for Choosing Django ORM as API**: **1.** Django ORM allows mapping database tables to pure Python objects, making data operations like insert, delete, modify, and query simpler and code more readable by eliminating the need for explicit SQL code usage in our programme. **2.** Django ORM provides rich query methods for handling complex filtering, sorting, and relationships between tables, making it ideal for projects that require frequent database operations. **3.** Django ORM includes built-in tools for data migration, making it easier to update database structures as the system evolves, thus enhancing both maintainability and scalability. **4.** Django ORM allows for seamless switching between different database backends (e.g., SQLite, PostgreSQL, MySQL), increasing database compatibility across environments, and we choose SQLite3 for data storage and extraction.
- **Application in Non-Web Framework Environment:** Though originally designed for web application, Django ORM can run independently of the web framework, requiring only basic configuration of a Django settings file, including database connection info and app modules. This approach allows full use of Django ORM without the dependencies of the Django web framework. This flexibility allows Django ORM to be suitable for our tasks.
- **Limitations of Django ORM: 1.** Django ORM might be less efficient than raw SQL. However, it is still suitable for our project without extensive needs on querying speed. **2.** Django ORM requires some foundational configurations (e.g., settings.py file), which, while functional without the web framework, can be a bit cumbersome.

Functional Description of Components

Database Schema Design



Rules Design:

Constraints

- **Primary Key Constraints:** db_cloth.cloth_id is the primary key of db_cloth table; id is the system generated primary key of db_mask table.
- **Foreign Key Constraints:** db_mask.cloth_id is the foreign key, linking to the db_cloth table.
- **Not Null Constraint:** Applied for all.

Relationships

- **db_cloth Table and db_mask Table Relationship:** Each clothing may have multiple components, establishing a one-to-many relationship. This is implemented by adding a cloth_id foreign key in the db_mask table.

Functional Description of Components

Storage Data Example

db_cloth

cloth_id	104851	**pattern**	no pattern
description	Light blue crewneck fitted jumper from ALYSI featuring ribbed-knit edge, crew neck, long sleeves and straight hem.crewneck fitted jumper	**placket**	hedging
cloth_image	./cloth_data_\id_00010 4851\id_000104851.jpg	**pocket**	no pocket
accessories	bead	**season**	spring and autumn
body_part	top	**skirt_length**	none
category	knitted sweater	**sleeve_length**	extra long sleeve
cloth_orientation	back	**sleeve_type**	regular sleeve
coat_length	conventional	**species**	women
collar	regular turtleneck	**style**	simple_wild_casu al
color	Light blue	**suit**	none
craft	other craft	**suitable_age**	youth(18-34)
elbow	conventional	**texture**	solid color
fabric	knitted	**type_version**	Slim fit
hem	conventional	**upper_dress_co ntour**	X type
pant_length	none	**visible_part**	complete
pants_contour	none	**waist**	none

…

db_mask

id	3802
mask_id	2
mask_type	sleeve
mask_image	./cloth_data_\id_00 0104851\id_000104 851_seg\id_000104 851_2_sleeve.bmp
cloth_id	104851

db_mask

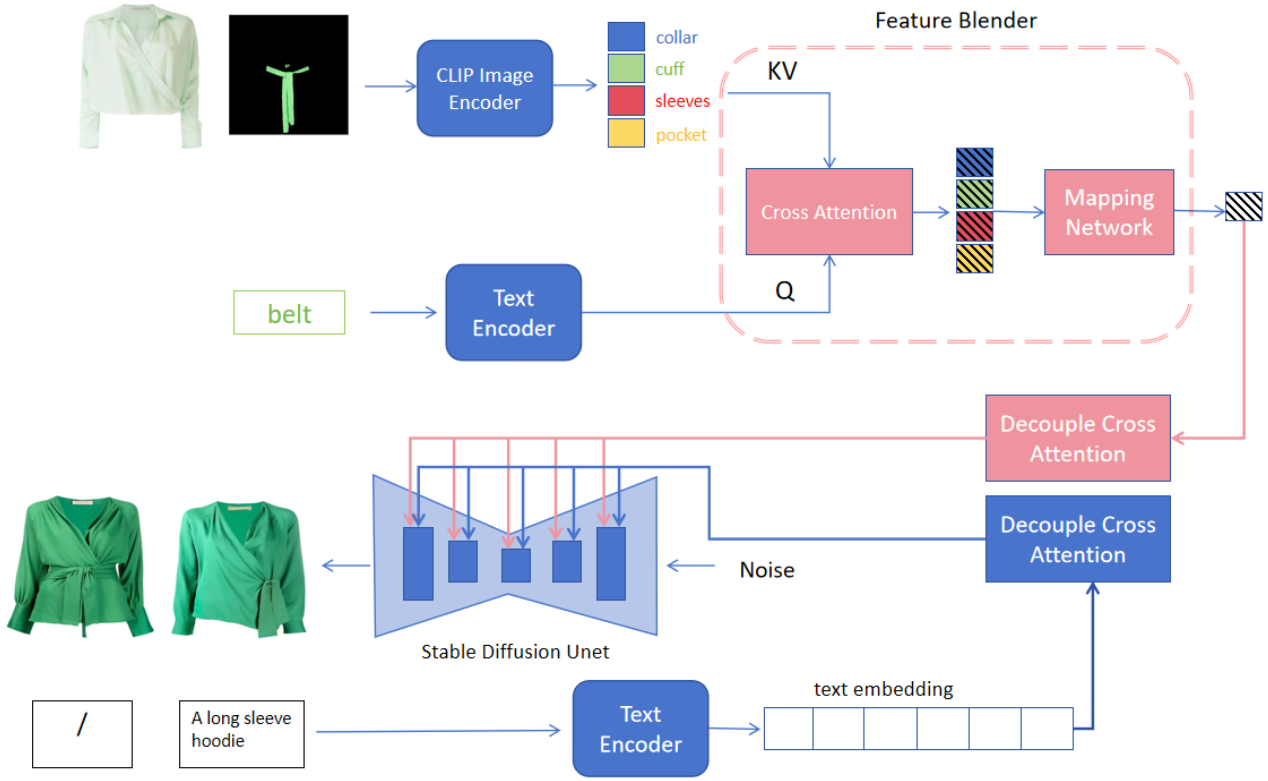
id	3803
mask_id	3
mask_type	collar
mask_image	./cloth_data_\id_00 0104851\id_000104 851_seg\id_000104 851_3_collar.bmp
cloth_id	104851

…

Functional Description of Components

➢ Clothing Generation Model: Fashion Adapter

We designed a fashion adapter based on the IP-adapter structure, adding a Feature Blender to enable feature fusion.



The CLIP image encoder encodes images of clothing parts, which are then fused with text features generated by the text encoder. The resulting text-image features are fed into a decoupled cross-attention module for image generation.

Functional Description of Components

➢ Fashion Adapter: Feature Blender

The Feature Blender contains a cross-attention module and a mapping network. The specific structure is shown below.

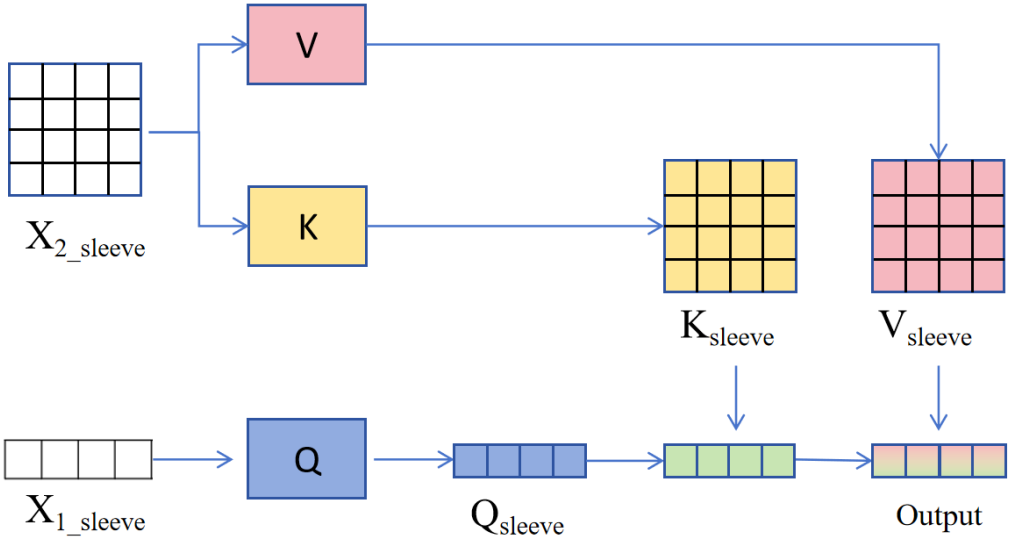
Let *W* represent the weight parameters of the linear projection, X_1 represent the input text sequence, and X_2 represent the input image sequence, then we have:

$$Q = X_1 W^Q \qquad K = V = X_2 W^K$$

Compute the cross-attention output as follows:

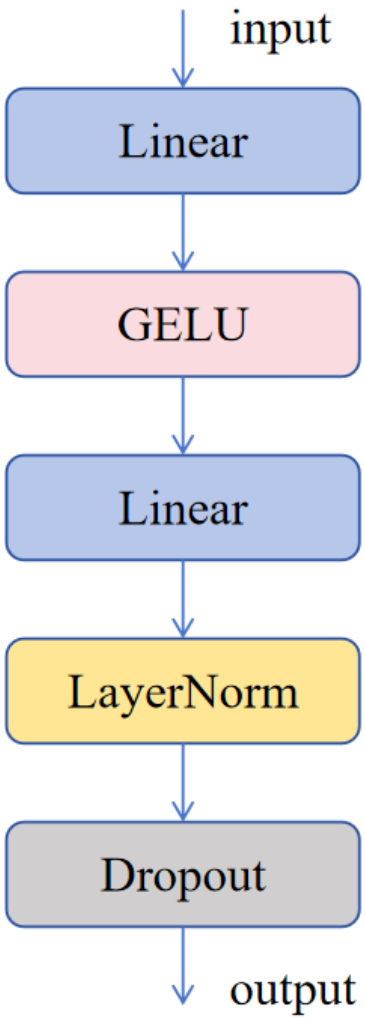
$$\text{CrossAttention}(X_1, X_2) = \text{Softmax}(\frac{QK^T}{\sqrt{d_2}})V$$

where $X_1 \in R^{4\times d_1}, X_2 \in R^{64\times d_2}, \ W^Q \in R^{d_1\times d_k}, W^K \in R^{d_2\times d_k}$



Functional Description of Components

➢ Fashion Adapter: Feature Blender

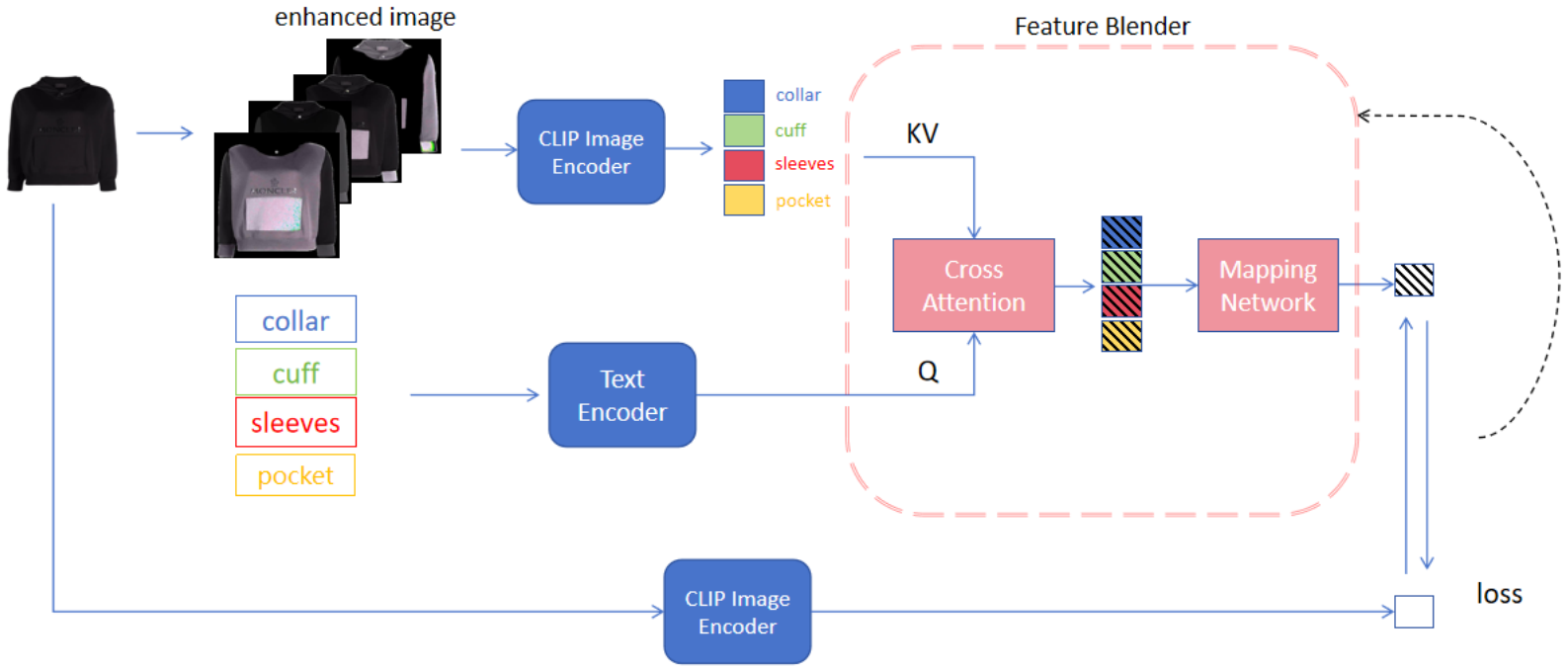


The structure of mapping network is shown in the figure.

•**Linear Layer**: Applies a linear transformation to the input feature data by learning adjustable weights and biases, mapping it to the output feature space and completing feature extraction and transformation.
•**GELU Function**: Maps feature values between 0 and 1, enhancing the model's non-linear capability and improving its expressiveness and performance.
•**LayerNorm Layer**: Normalizes the input to ensure similar mean and variance across layers, reducing internal covariate shift, which helps accelerate network convergence and improve model generalization.
•**Dropout Function**: Randomly sets the output to zero with a certain probability to prevent overfitting on training samples.

Functional Description of Components

➢ Fashion Adapter: Training Process



After data augmentation, each segmented clothing part is encoded by the CLIP image encoder to obtain feature encodings, while part names are encoded by the text encoder to produce text features.
Image features serve as the key and value, and text features as the query in cross-attention, allowing automatic extraction of part features from the image based on text.
The combined image and text features are linearly represented to obtain fused part features.
During training, the fused features and the original image encoding are optimized using mean squared error as the loss function.

Functional Description of Components

➢ AI Virtual Try-on Model: CatVTON

To better showcase the generated clothing as if worn on a person, we use CatVTON to implement the try-on function.
CatVTON is easy to use; by specifying the try-on area of the clothing, it automatically generates corresponding masks for the model's specified body parts using DensePose and SCHP.
Then the edited garment will be inpainted into the mask.

Model Image | Agnostic Mask | Result Image

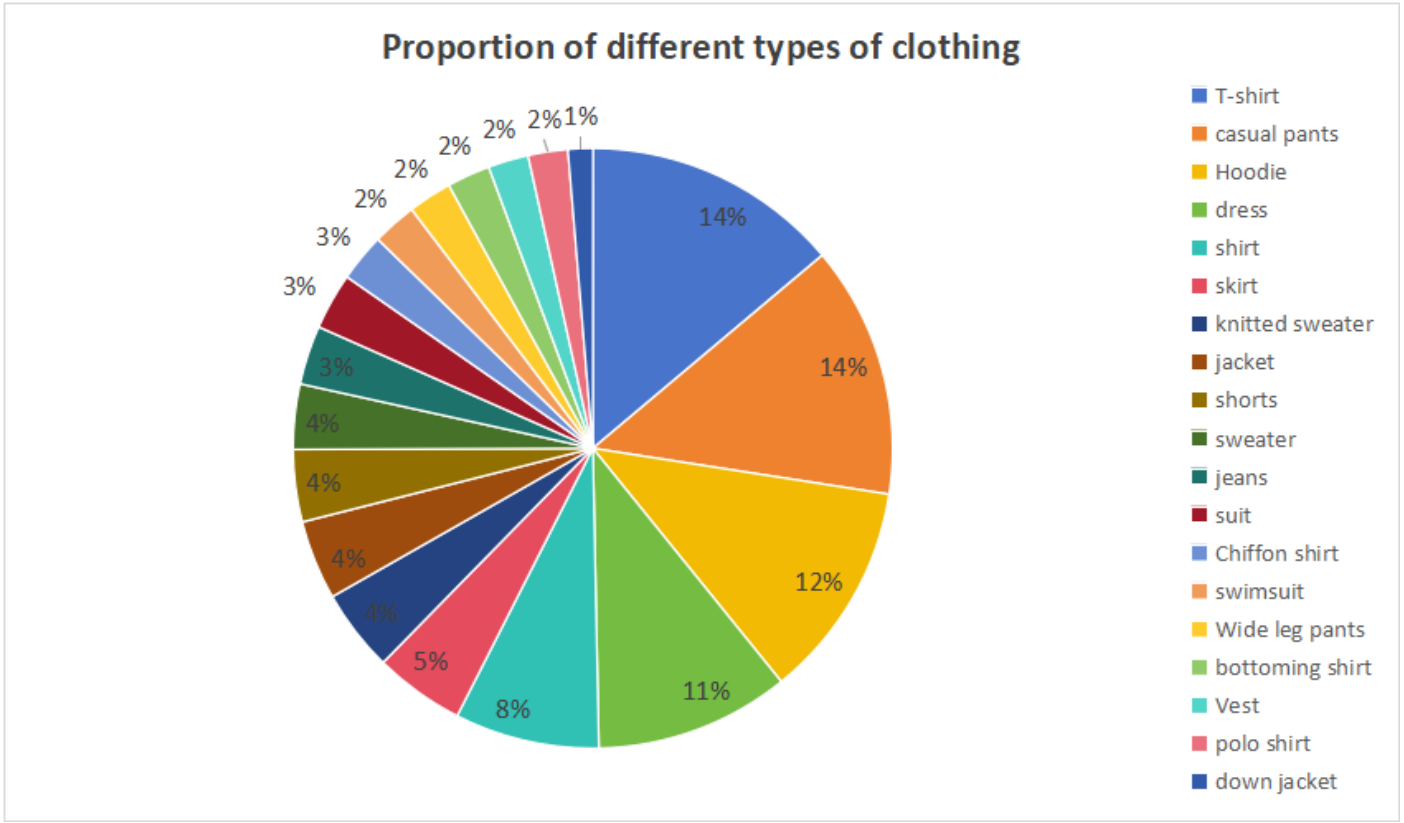
4 SYSTEM DEVELOPMENT & IMPLEMENTATION

- Development Process

Development Process

➢ Data Collecting and Preprocessing

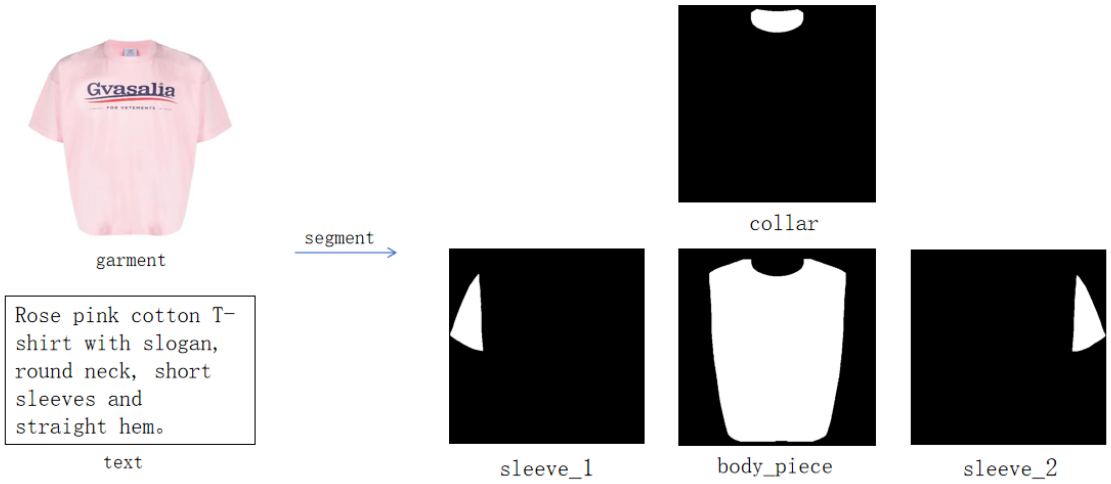
- We collected data from the open-sourced CM-Fashion dataset. The dataset contains 444,214 clothing images and their corresponding text descriptions in natural language, and a total of 49 different clothing categories, in which 19 clothing categories with a data volume greater than 5000, a total of 402,933 pieces, accounting for about 90% of the total data volume. The distribution of the data is shown in the figure below.



Development Process

➢ Data Collecting and Preprocessing

- For Fashion Adapter model training and the subsequent development of clothing modification function, complete clothing images need to be segmented by parts to obtain different components of each garment. We utilised the PointRend technology in the Detectron2 framework to achieve high-precision image segmentation, making the segmented edges more closely match the contours of the original image, ensuring clear details. PointRend performs fine-grained sampling and prediction on image boundaries, effectively improving segmentation accuracy at the edges.
- In practice, the CM-Fashion dataset is segmented to generate segmentation masks for 12 types of clothing components (such as collar, sleeve, body, etc.), with each part accompanied by a corresponding text description. The figure below shows an example of segmented parts. Additionally, statistical analysis of the segmented data reveals an uneven distribution of components, with parts like sleeves, buttons, and bodies being the most frequent, while skirts, hats, and belts are less common. This imbalance could impact model training effectiveness, thus requiring appropriate dataset partitioning to balance the training data.



Development Process

➢ ETL Engineering

- To normalize data, minimize redundancy, and enable efficient data extraction, we opted to extract clothing features and store them in a structured format within the database, rather than simply relying on natural language descriptions.

- We developed an automated process to achieve this goal:

Extract

- Extracted unstructured clothing description data from the downloaded CM-Fashion dataset using Python scripts.

Transform

- Transformed the extracted data into structured JSON format using ChatGPT

Details:

- We feed raw text description into ChatGPT in batch by using relevant API with a carefully crafted prompt that asks the model to extract attributes (e.g., color, material, type) and output them in JSON format.
- Example: For instance, the description was “Pink short-sleeve T-shirt with logo”, and the output was “{"color": "pink", "sleeve": "short", "type": "T-shirt", “pattern": "logo"}”. We would set a default value for each feature if it’s not mentioned in the description text.

Load

- Loaded the data into our self-constructed database.
- This work goes hand in hand with the database construction, and we needed to ensure the JSON structure aligns with the target database schema.

Development Process

➢ Database Construction

Database Initialization and Environment Setup

- We started by setting up the Django environment and configuring the SQLite3 database. In the settings.py file, we defined the database connection.
- This configuration establishes a connection to an SQLite3 database file (cetaImages.sqlite3) in the project's base directory. This database is used throughout the development process.

Creating Data Models and Tables

- Using Django ORM, we defined data models as Python classes that correspond to database tables in models.py.
- After defining the model, we run Django migration commands. This generates and applies SQL statements to create the User table in the SQLite3 database.
- We did the same thing to deploy the db_mask table.

Development Process

➢ Database Construction

Creating Functions for “inserting, deleting, and querying”

Using Django ORM, we defined functions for inserting, deleting, and querying data in our two tables. For instance, for db_cloth table, we defined query_cloth_by_dict function to receive querying dictionary and return the querying result in insert_images.py, delete_images.py, query_images.py.

Creating API for Interacting with the Other Project Components

- After finishing constructing the project database, we created different API to realize our objectives. For instance, we defined get_recommendation function for users to get a recommended clothing image based on their demand in get_recommendation.py.
- The function will receive the user’s natural language query, and return a dictionary including error detection flag, prompt engineered user’s query (if applicable) and recommended image path (if applicable).
- The function involves the interaction with ChatGPT API that will be introduced later. We also left the room for interacting with the Memory module (not finish developing due to the time limitation) in the future.

Development Process

➢ Fashion Adapter Model Training

- The training processes for Fashion Adapter are implemented based on PyTorch and trained on a single RTX 3090 GPU, with the Feature Blender and IP Adapter trained separately.
- The Feature Blender module is trained with a learning rate of 1e-6 and a batch size of 50 for 3 epochs, spending approximately 40 hours.
- The IP Adapter is trained with a learning rate of 1e-4 and a batch size of 8 for 3 epochs, taking about 120 hours.
- During both training processes, the weights of the image and text encoders are frozen.
- The division of the training data is shown in the figure below, where *D* represents the entire dataset.

Training Module	Training Set	Training set representation	Amount of training data (thousand)
IP Adapter	D_a	40%D	160
Feature Blender	D_f	10%(D-D_a)	24

Development Process

➢ Catvton Implement

Image Processing and Loading

- The full-body photos of users and clothing images are resized to a fixed dimension (768x1024) to ensure consistency in input data.

Mask Generation

- The DensePose and SCHP models are used to generate masks, accurately segmenting the user's body contour and background.
- AutoMasker is employed to automatically generate masks and adjust them to match the dimensions of the person image.

Pipeline Initialization

- The virtual try-on pipeline is initialized using pre-trained Stable Diffusion and CatVTON model weights.
- Half-precision (bf16) and TF32 are enabled to optimize computation efficiency and GPU resource utilization.

Inference and Image Generation

- A random seed is set to ensure the reproducibility of the generated results.
- The inference process is carried out with 75 steps, using a guidance scale of 7.0 to maintain a balance between image realism and conditional input.

Output and Saving

- The generated try-on result images are saved to a specified path for users to preview and utilize.

Development Process

➢ ChatGPT API Deployment

Deploying ChatGPT in Telegram Bot

Obtain an OpenAI API key

↓

Write a function in the Telegram Bot to capture user messages

↓

Use the OpenAI API to generate responses

↓

Return the responses via the Telegram API for real-time interaction.

Deploying ChatGPT Prompt

Write a system Prompt covering key features. will be used for processing of all user inputs, ensuring that the GPT model generates structured feature information in JSON format as required by the template.

↓

Write a function to call OpenAI's GPT model and take user input, construct the Prompt, and send the Prompt along with the user input to OpenAI's API.

↓

Write a lookup function that traverses the generated dictionary data to construct query conditions and search for records that match the dictionary conditions.

Development Process

➢ Telegram Bot Deployment

Function Modules:
Five distinct core functions were created, including */search, /searchmask, /generate, /tryon*, and a fashion design assistant powered by ChatGPT. Each module serves different user needs, providing a diverse and interactive experience.

Text and Image Handlers:
Text and image handlers were implemented to ensure the bot accurately captures and processes both user text and image inputs.

User State Management:
Each user is assigned a unique state based on their ID, allowing the bot to distinguish which function each user is currently using, ensuring independent function operation without user interference.

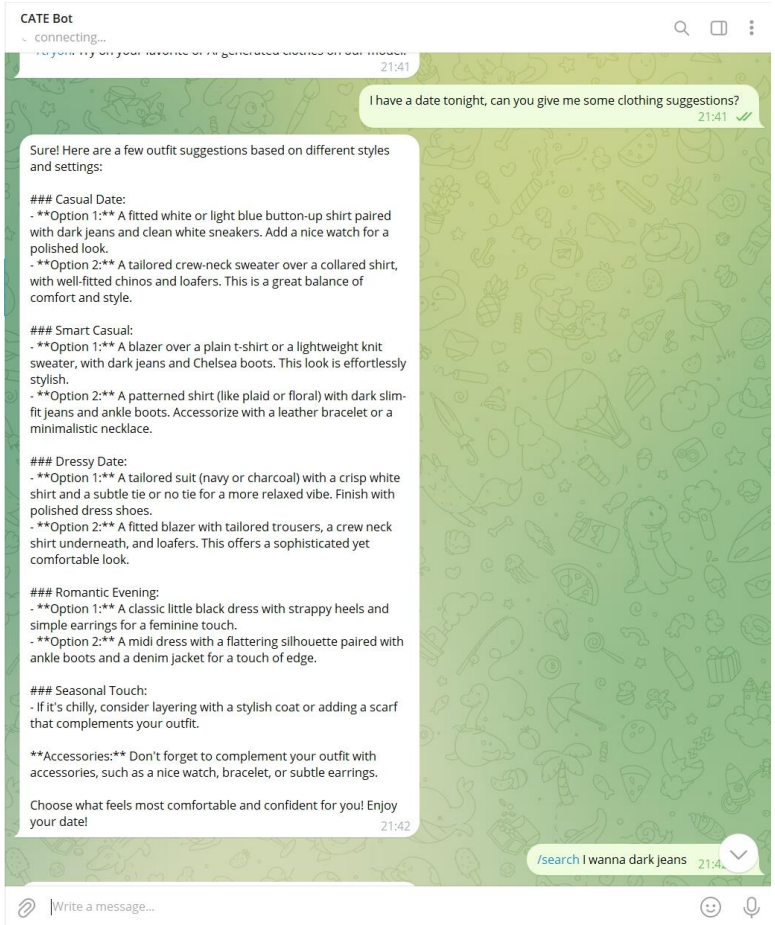
ChatGPT and Model Integration:
ChatGPT was integrated with various AI models, enabling the bot to interpret a single user input, retrieve suitable images from the database, and pass them to the model for generating the requested output, enriching the user experience.

5 FINDINGS & DISCUSSION

- Presentation of Project Results
- Interpretation & Discussions of the results

Presentation of Project Results

➢ UC01 Getting Recommendation Demo

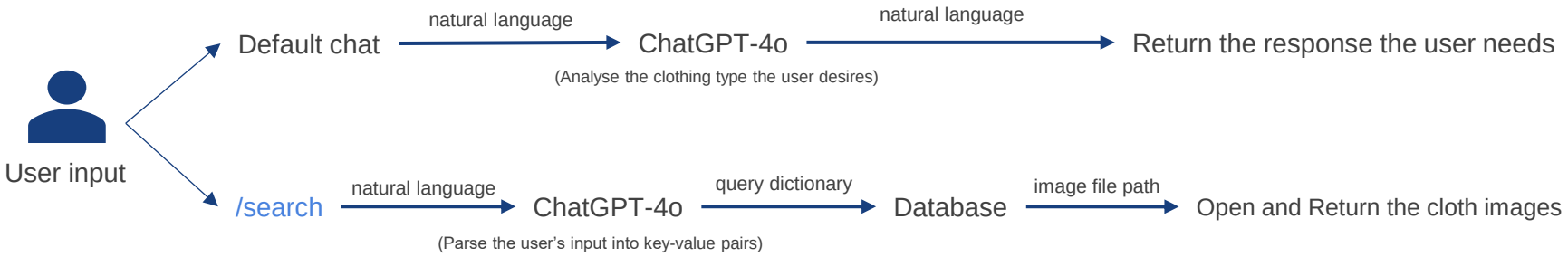


- The user provides clothing preferences or descriptions in natural language, and the system returns outfit recommendations by ChatGPT or images of suggested clothing items from the database. Pure text communication doesn't need order, but image recommendation function need to type /search in the beginning.

Presentation of Project Results

➢ UC01 Getting Recommendation Details

- **Use Case Workflow:**



- **Probable Errors:**
  - Too complex demand: If the user types in a very complex demand (i.e. I want a Prussian-blue, mint-green, and purple Arjuna color-block sports bra from ERES, designed with a color-block pattern…), the system may not be able to find a qualified piece of clothing in the database, and will respond “No recommendation found in the database” instead.
  - Failed to connect to ChatGPT: If there is a problem in network or API-KEY, ChatGPT will not be available, and the system will respond “Failed to connect to ChatGPT.”
  - Failed to open the image: If the path of the image stored in the database is not correct, the system will fail to open and respond the corresponding picture to the user, and will respond “There’s a problem in our database” instead.

Presentation of Project Results

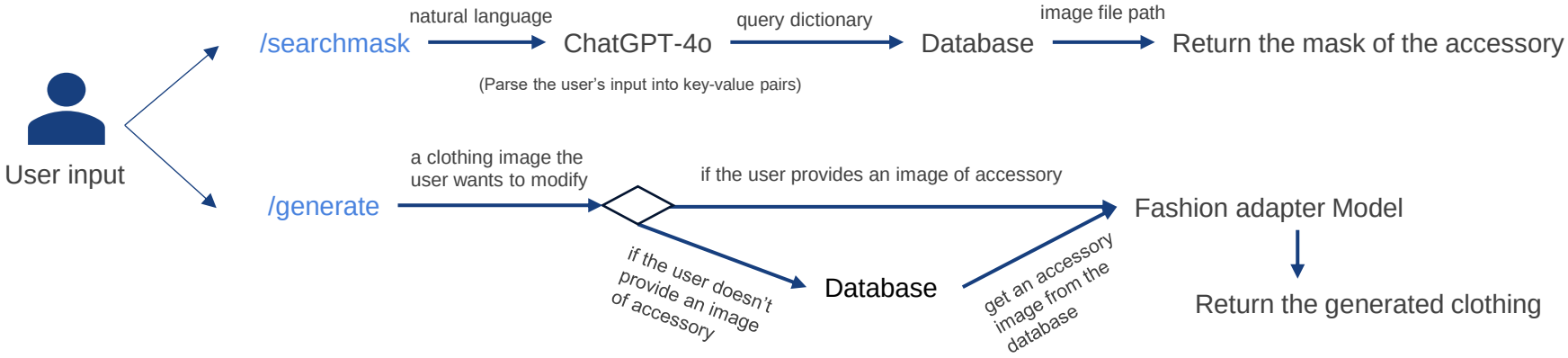
➢ UC02+03 Component-Based Garment Generation Demo

- The user provides clothing image and the type of the component (accessory) they want to add on it by using /generate order. The system will then return images of the updated outfit with different modification effect. The user can choose the one they feel the best by typing the number 1~8.
- The user can choose another way to modify their clothing under the /generate order. They can send an image of accessory instead of sending the type of the accessory to the system, which can achieve the same goal.
- The user can also search the existing mask image (i.e., component image) from the database by /searchmask order.

Presentation of Project Results

➢ UC02+03 Component-Based Garment Generation Details

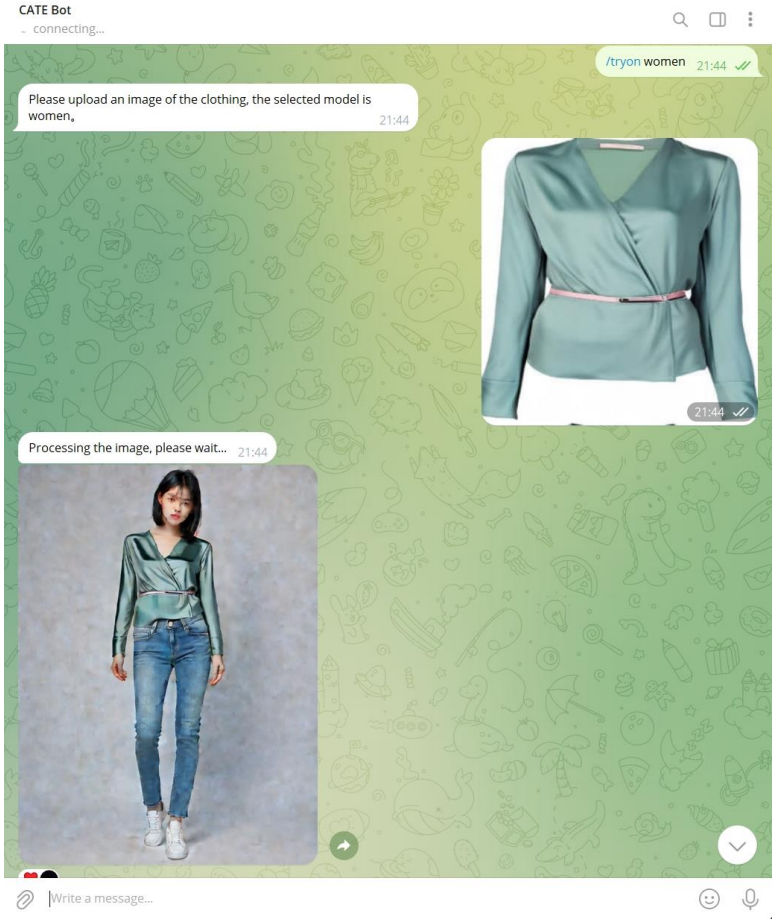
- **Use Case Workflow:**

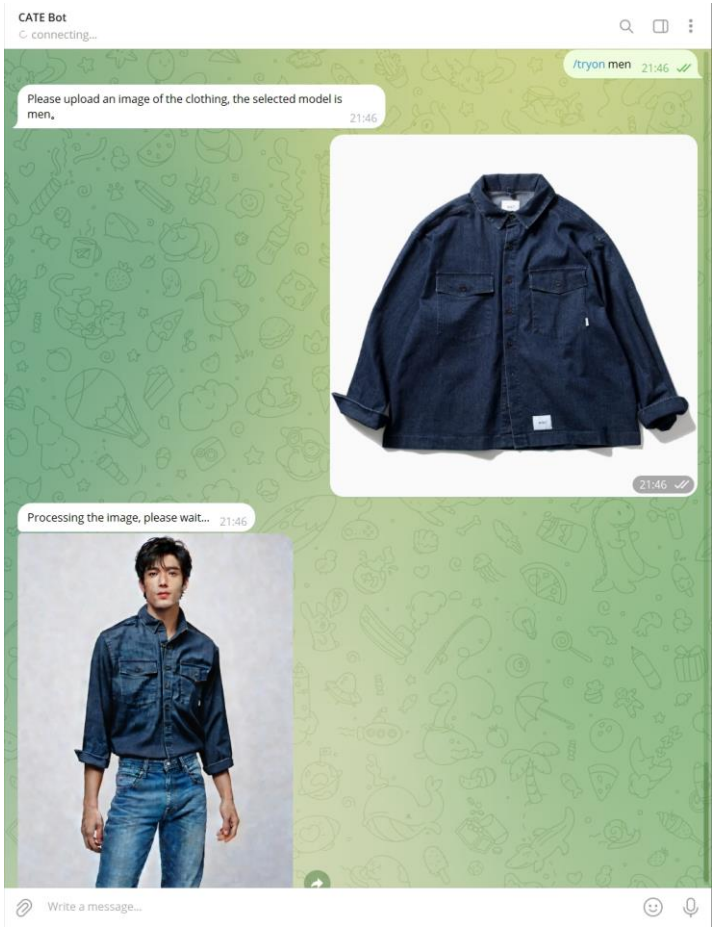


- **Probable Errors:**
  - Failed to connect to ChatGPT: If there is a problem in network or API-KEY, ChatGPT will not be available, and the system will respond “Failed to connect to ChatGPT.”
  - Failed to open the image: If the path of the image stored in the database is not correct, the system will fail to open and respond the corresponding picture to the user, and will respond “There’s a problem in our database” instead.

Presentation of Project Results

➢ UC02+04 Virtual Try-On Demo





- Under the /tryon order, the user can request the system to generate a model wearing the designated clothing. The user may specify the clothing items, the model, or even use themselves as the model (not realized in our project due to the time limitation). The system processes the input and returns images of the model dressed in the selected clothing.

Presentation of Project Results

➢ UC02+04 Virtual Try-On Details

- **Use Case Workflow:**

- **Probable Errors:**
  - None

Interpretation & Discussions of the results

➢ Result of Fashion Adapter

- Using the Fashion Adapter, clothing can be edited by inputting the clothing image and the image of the desired part to be edited.
- The tests were conducted both with part names as text prompts and without any text input. The specific generation results are shown in the figure.
- When combining the clothing to be edited with the part images, the generated images effectively blend the features of both.
- When using part features as text prompts, the generation results are better than those without any text input, with the added clothing part features being well preserved.

Interpretation & Discussions of the results

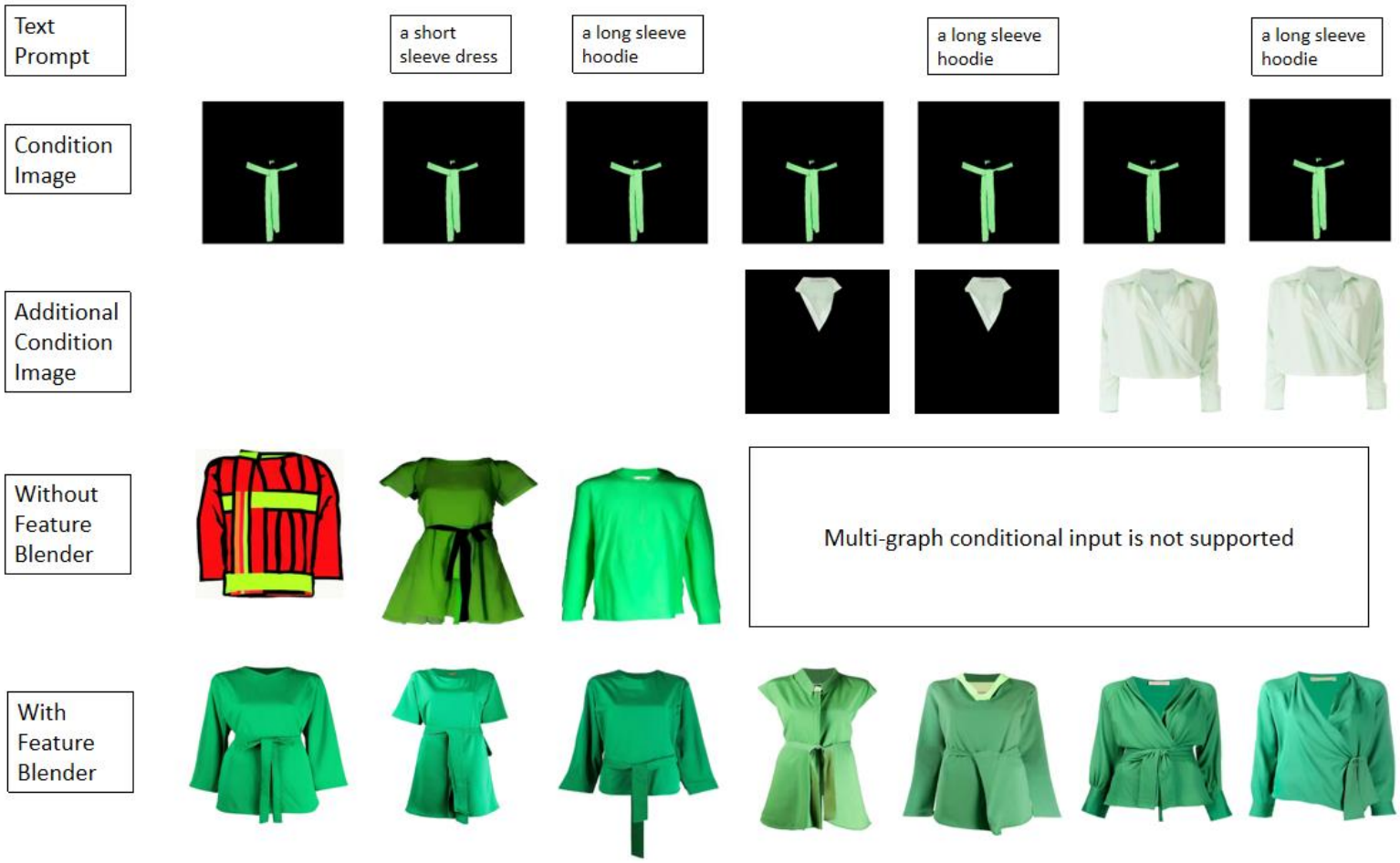
Result of Fashion Adapter

- It is also possible to generate a complete garment with the characteristics of a single clothing part.
- In the experiments, both scenarios were tested: inputting text prompts in the UNet and not inputting any text prompts. The specific generation results are shown in the figure.
- The generation results indicate that the model can generate complete garment images with the characteristics of a single part, regardless of whether text is inputted or not.

Interpretation & Discussions of the results

➢ Ablation Experiments for Feature Blender

Based on our designed fashion adapter, we conducted ablation experiments to verify the effectiveness of the Feature Blender.



Interpretation & Discussions of the results

➢ Feature Blender

- Experimental results show that without the feature fusion module, the IP-adapter can only accept single-image inputs.
- While it can extract color information, it fails to associate it with the corresponding parts, leading to less realistic clothing images.
- When there is a significant discrepancy between text and image prompts, the generated images cannot exhibit features from both.
- With the feature blender module, multiple images can be used as conditions, effectively maintaining the features of each, regardless of whether text is presented. This approach results in more natural and realistic clothing images.
- Thus, it demonstrates that the proposed feature fusion module can effectively extract and integrate various clothing features, generating realistic clothing images.

6 CONCLUSION & FUTURE WORK

- Our Key Takeaways
- Potential Improvements & Next Steps

Our Key Takeaways

Telegram Bot Platform:
Provides a convenient user interface through Telegram Bot, allowing users to interact anytime, anywhere.

ChatGPT Semantic Analysis:
Uses ChatGPT to analyze user needs and retrieve the required clothing or accessories from the database accurately.

AI Clothing Generation and Try-On:
Generates clothing designs using the Fashion Adapter and CatVTON models, with try-on effects displayed on AI models.

Integrated Fashion Generation Platform:
Combines Telegram Bot, ChatGPT, and AI models to build a stable and usable platform for clothing generation and virtual try-on, enabling a complete workflow from request to display.

Potential Improvements & Next Steps

Improved Intent Recognition:

Enhance GPT's ability to recognize user intent accurately. The goal is to allow users to simply upload an image or express an idea, enabling the bot to automatically initiate search, clothing generation, or virtual try-on without the need to manually select functions.

Enhanced User Memory for Personalization:

Implement temporary storage of user conversations in the database, and, with user consent, save their preferences. This will allow ChatGPT to remember previous interactions, offering a more personalized experience during future conversations with the fashion design assistant.

Optimized AI Try-On Feature:

Improve the virtual try-on functionality by allowing users to upload their own model photos, providing a more realistic and customizable experience.

Project Commercialization:

Move towards commercialization by introducing a subscription-based service, offering premium features and exclusive designs for subscribers.

7 APPENDIX

- Installation and User Guide
- Knowledge Map
- Proposal

Installation and User Guide

First, you need to prepare your own ChatGPT API key.

You can install the virtual environment by executing the following instructions:

- **conda create -n CETA python==3.9.0**
- **conda activate CETA**
- **pip install diffusers==0.22.1**
- **pip install git+https://github.com/tencent-ailab/IP-Adapter.git**
- **cd CETA/Catvton**
- **pip install -r requirements.txt**

Then you can download the required models and put them into CETA/models:

- CLIP-ViT-H-14:https://huggingface.co/laion/CLIP-ViT-H-14-laion2B-s32B-b79K
- Fahion-sd-2.1:https://huggingface.co/Zhangwq76/fashion-adapter/tree/main/fashion-sd-2.1
- Catvton: https://huggingface.co/zhengchong/CatVTON
- stable-diffusion-inpainting:https://huggingface.co/booksforcharlie/stable-diffusion-inpainting

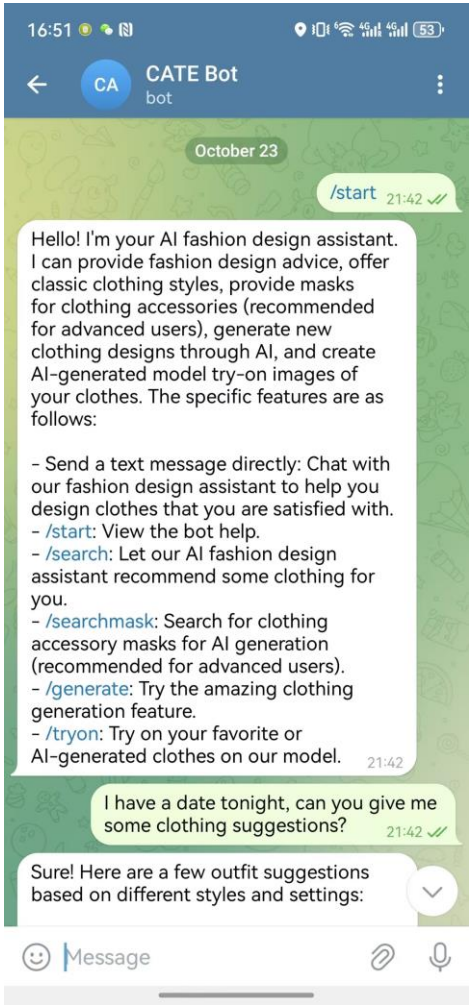
Installation and User Guide

Now you can start the robot by running:

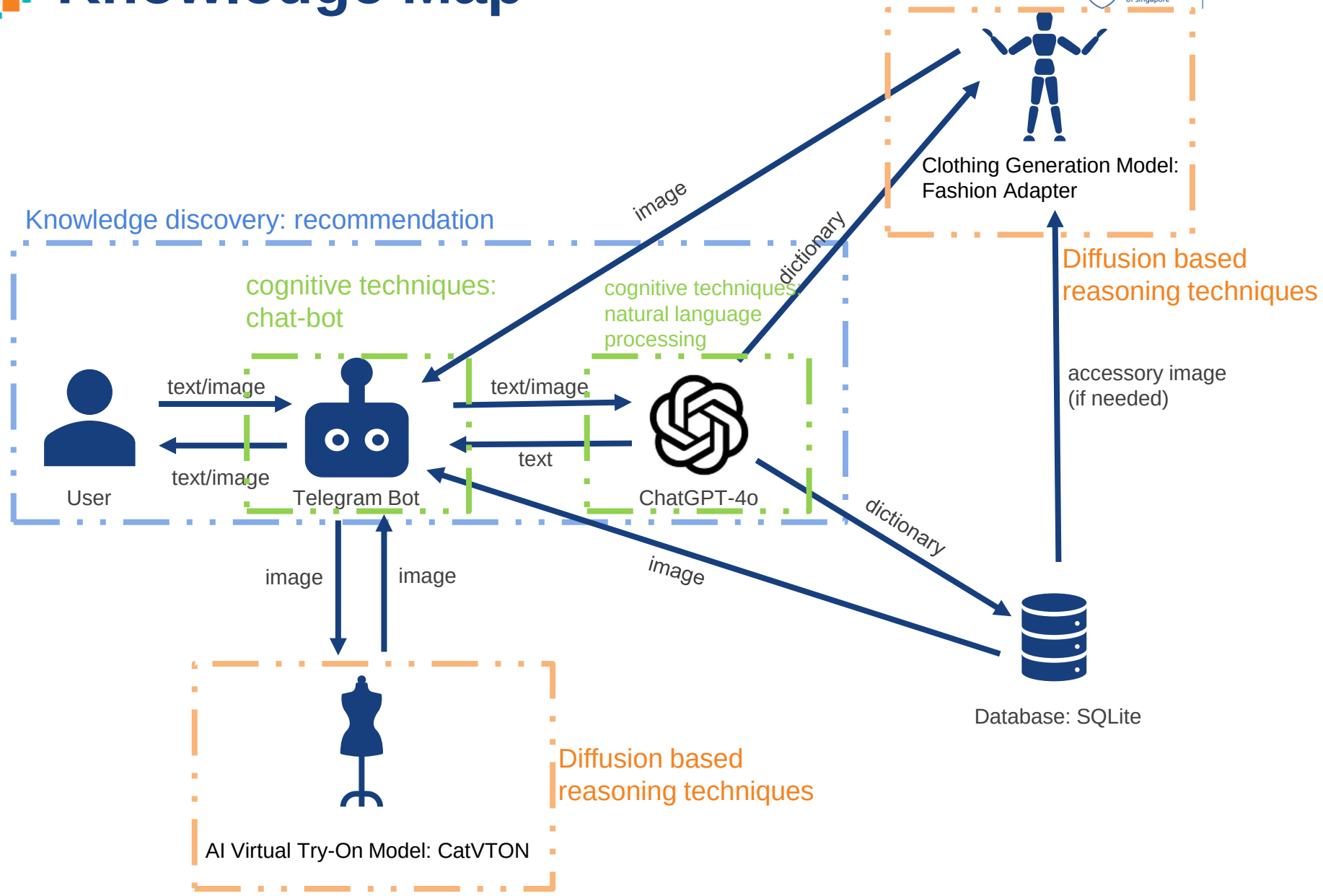
- **cd CETA/TelegramBot**
- **python bot.py**
- **Then open your Telegram and search @CodeAllNight_bot**

If you run it successfully, you will see the screen shown on the right. The robot will automatically send you instructions.

Detailed use cases please refer to *Presentation of Project Result* in "5 Findings & Discussion" chapter.



Knowledge Map



PROPOSAL OF CETA

Fashion agent for

Clothing Editing and Try-on Application

Group Number: 16

Team Name: Code All Night

Rain all day, code all night

Contents

PROPOSAL

1 INTRODUCTION

- Proposal Overview
- Importance and Relevance
- Project Overview

1.1 Proposal Overview

Project Background Analysis: Analyze current consumer demand for personalized recommendations and virtual try-on experiences and identify market pain points and needs.

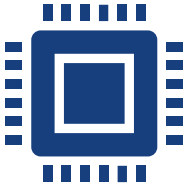
Literature Review: Study existing technologies (such as recommendation systems based on large language models, clothing image generation technology, and virtual try-on technology) to provide a theoretical foundation for system design.

Project Scope Analysis: Explain the computing resources used for CETA. Define the scope of input and output of each step in detail, and analyze the specific workflow of different use cases.

Data Collection and Preparation: Present statistical insights and an overview of the CM-Fashion[1] dataset distribution. Outline the data augmentation methods applied during the process.

System Design: The overall system architecture is designed, and the input and output defined in the scope section are combined with the used models to draw the model flow chart.

1.2 Importance and Relevance



Modern consumers' demand for personalized recommendations and virtual try-on experiences is growing, but current image generation models struggle to accurately generate specific clothing components.

Therefore, a fashion agent that can accurately understand user intent to recommend clothing, edit outfits based on user preferences, and showcase the results using AI models is of considerable importance.

CETA improves the accuracy of recommendations and the realism of component-level clothing generation. With a try-on model, it can visualize the final result more realistic, **meeting consumers' needs for personalized** and **real-time interactive experience**s. It has broad application prospects in fashion and e-commerce, significantly boosting user satisfaction and purchase decision efficiency.

1.3 Project goals

The purpose of the CETA is to enable precise and controllable fashion editing by image of a given photo by integrating various pre-trained models while adhering to human interaction habits. It contains three components:

Recommendation System Based on LLM: The system generates personalized clothing recommendations based on the user's text input, ensuring that the recommended items align with the user's description and needs.

Component-Based Garment Generation Model: Using cross-attention mechanisms and a clothing adapter, the system automatically generates accurate clothing components, ensuring high consistency between the text prompts and image features, resulting in natural-looking clothing images.

Virtual Try-On Model: The generated clothing images are integrated with a virtual model to showcase how the selected clothing would appear on the user, enhancing the virtual try-on experience.

2 PROJECT BACKGROUND/ MARKET CONTEXT

- Demand In Clothing Industry
- Market Context
- Overview of Market Landscape
- Problem Addressed

2.1 Demand In Clothing Industry

Consumer personalization needs: Apparel consumers expect a personalized shopping experience, as well as dress recommendations for specific occasions and temperatures. However, traditional recommendation systems are often based on the user's historical purchase data, making it difficult to analyze and respond to the different dynamic needs of individuals in real time.

High return rate: Due to the lack of online trying on experience in online shopping, consumers are not accurate in judging the suitability of the clothes, resulting in a high return rate, which leads to a large waste of time costs and logistics resources (McKinsey & Company, 2023)[2].

Slow innovation: As fashion cycles shorten, brands need to respond to trends more quickly. Traditional design and production processes are relatively slow, but AI can accelerate the cycle of design, production, and market feedback.

2.2 Market Context

- **Digital transformation of fashion industry**: The fashion industry is gradually transforming from brick-and-mortar stores to e-commerce platforms. ai technology is being widely used in recommending products and virtual fitting. (Intelistyle,2023)[3]
- **Component-based Garment Generation and Editing**: Existing work on cross-modal garment synthesis is mainly based on a two-phase process of generic generation of transformers, but this ignores the structural correspondence between garment images and input textual cues, which can lead to problems of imprecise cross-modal semantic alignment and poor quality of semantic combinations.
- **Natural Language Recommendation**: The current marketplace relies heavily on simple keyword searches and cannot handle more complex descriptions provided by users. For example, when users use fuzzy language or multi-dimensional requirements, the existing system's recommendation is not ideal for personalized, dynamic and accurate matching.

2.3 Overview of Market Landscape

- **Promoting customized design and innovation:** With component-level apparel generation and editing, our system helps brands accelerate the design process by providing users with customized apparel options as well as faster feedback from the market during the design and production cycle, resulting in faster product iterations.
- **The Advantage of Personalized Recommendation：** With natural language processing technology, our system is able to handle more complex and multi-dimensional user descriptions, no longer relying on simple keyword matching, but deeply analyzing consumers' personalized linguistic inputs.
- **The Business Potential of Virtual Try-On：** Based on AI model try-on virtual try-on technology can dramatically reduce the return rate of online purchases and increase user confidence in online shopping.

2.4 Problem Addressed by CETA

- **Achieving seamless interaction**: In traditional AI-powered clothing platforms, users may need to switch between multiple platforms or applications to complete the entire process from naturally language-based recommendations for clothing styles, designing personalized outfits, then previewing try-on effects. By integrating these functions, we have streamlined and made the entire process flow smoothly and seamlessly, significantly enhancing the user's interactive experience. This addresses the issue of poor algorithm model interaction and coherence with users.
- **Adapters based on clothing dataset:** Most of the existing visual grand models and adapters are trained based on generic images, and using these models and adapters for clothing image generation may result in cluttered background of the generated images and unreasonable generation of clothing parts. In order to make the model's image generation results more natural and closer to reality, CETA trains a clothing adapter based on a specific clothing dataset;
- **Addressing limitations in user recommendation system:** Current recommendation systems often rely on basic keyword matching, which can fail to capture the full complexity of user preferences and descriptions. Our system leverages advanced natural language processing (NLP) to analyze multi-dimensional user inputs, enabling deeper understanding of personalized preferences.

3 LITERATURE REVIEW

- Intelligent Agent
- Fashion Editing Task
- Base Methods

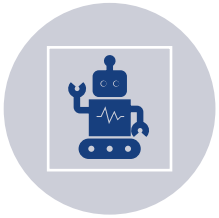
3.1 Intelligent Agent

3.2.1 Garment Image Editing

AutoGPT[4] designed an intelligent agent system that receives plans from users via natural language, breaks these plans into several sub-tasks, and automatically uses the internet or other tools to accomplish these tasks.

HuggingGPT[5] proposed a system that utilizes Large Language Models (LLMs), such as ChatGPT, to connect various AI models within the HuggingFace community to solve AI tasks.

Minidalle3[6] combines LLMs with image segmentation and generation models to create an interactive text-to-image system.



In LLM-driven AI agent construction, these models play a central role by employing mechanisms such as **Chain-of-Thought (CoT)**[7]**, React**[8], and others, to reason about specific objects and achieve desired outcomes through connecting external tools.

3.2 Fashion Editing Task

3.2.1 Garment Image Editing

Text-to-Clothing Methods:

HieraFashDiff[9], Picture[10], Unihuman[11]and some other methods focus on generating and manipulating clothing on human figures.

Armani[1], Diffcloth[12] and Dresscode[13] directly convert given descriptions into in-store clothing images.

Pre-trained Models:

Picture[10], Multimodal Garment Designer[14], Dreampaint[15] use pre-trained text-to-image latent diffusion models[16] as a backbone.

They adapt these models to the text-to-clothing domain by adjusting descriptions or text embeddings, or by fine-tuning the backbone.

Component-Level Importance:

Diffcloth[12] and **ARMANI**[1] utilize semantic segmentation of segmented descriptions and clothing images for component-level alignment.

HieraFashDiff[9] implements hierarchical design concepts to edit clothing components worn by models.

Fashion-Diffusion[17] introduces a dataset annotated at the component level

3.2 Fashion Editing Task

3.2.2 Portrait Editing

Previous Work:

Viton-hd[18], CP-VTON[19], Pasta-gan++[20] focused on virtual try-on techniques for image-to-image synthesis.

Stylegan-human[21] addressed unconditional human generation.

These approaches had limited image detail and control during the generation process.

Text2Human[22]:

Proposes a method to first generate a human parsing map from a given pose.

Creates the final human image by providing a text description of clothing and additional information about the clothing texture.

HumanDiffusion[23]:

Introduces a text-to-image fashion editing technique.

Enables generation of portraits based on text or label-guided pose or semantic segmentation maps.

FICE[24]:

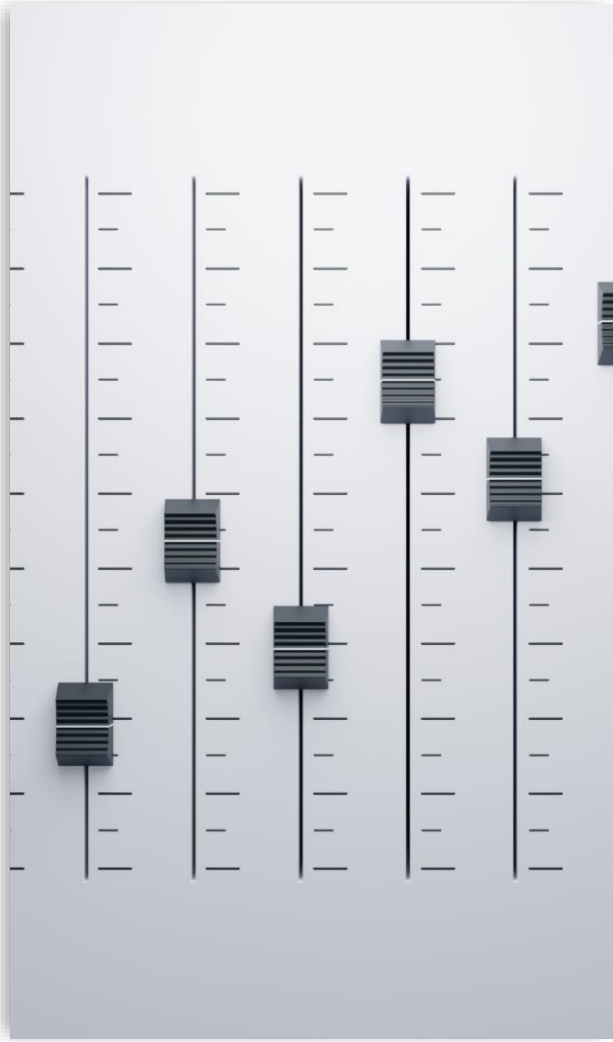
Uses GAN inversion techniques to modify portrait photos based on text prompts.

Maintains personal features while applying textual modifications.

3.3 Base Methods

3.3.1 Shortcomings of Image editing Methods

- Large pre-trained models can generate images based on conditional inputs effectively through fine-tuning, but due to their vast number of parameters, fine-tuning them requires a significant amount of time and computational resources.
- To address the this issue, an efficient approach is to use adapters by adding a small number of trainable parameters while freezing the original large model's parameters during training.
- ControlNet[25] and Uni-ControlNet[26], which utilize edge detection, depth maps, and other conditions, can effectively preserve the edge information of the conditional image when generating images. However, they perform poorly in retaining color and texture information.
- While the T2I-adapter[27] can accept color and style conditions, it also struggles to generate good results when the conditional image is incomplete.



3.3 Base Methods

3.3.1 Shortcomings of Image editing Methods

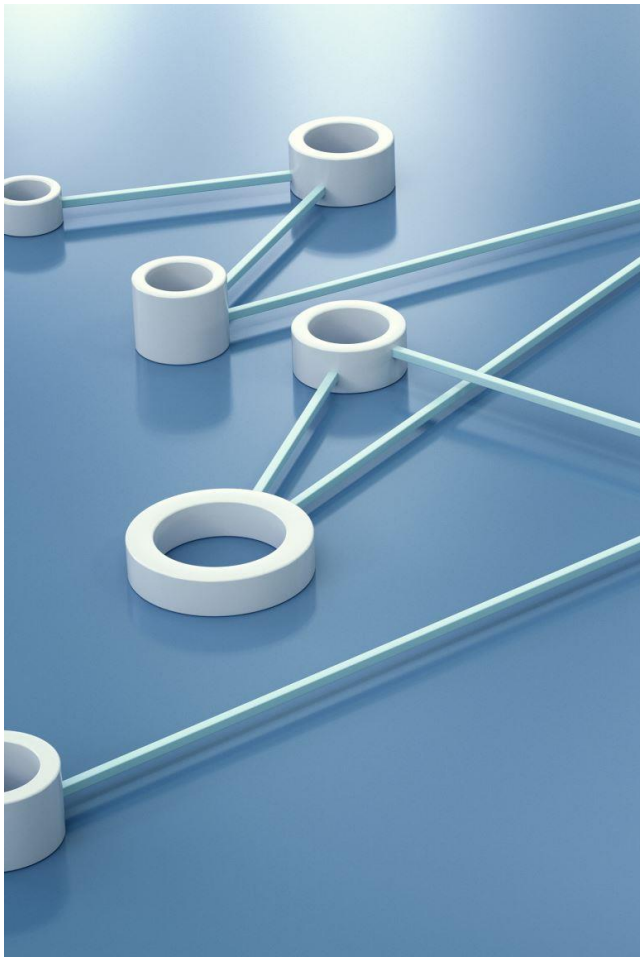
- To address the above issue and enable the model to generate high-quality results from both complete and incomplete conditional images, we proposes a feature extraction method based on a cross-attention mechanism. By combining a small mapping network and adapters fine-tuned on a text-to-garment dataset, the generated images can better preserve the structure, color, and texture information of the conditional images.
- Then we use stable diffusion as the generative model to generate and edit clothing images.

3.3 Base Methods

3.3.2 Selection of LLM

We use **Yi-Large** as recommender in CETA, with benefits below.

- **Economic Efficiency**: **Yi-Large** can be easily integrated into existing systems via APIs without requiring expensive infrastructure and development investments, reducing the development and maintenance costs for businesses. It is suitable for small and medium-sized enterprises, as it doesn’ t rely on complex algorithms and massive data processing like traditional recommendation systems.
- **Real-Time Updates**： With its ability to dynamically learn, **Yi-Large** can quickly adapt to new information and make corresponding adjustments to recommendations, making the results more aligned with users' real-time needs.
- **Intelligent Feedback Mechanism**: It adjusts recommendation results in real-time based on user feedback, offering a personalized interaction that allows users to feel the improving accuracy of recommendations, thus enhancing the overall user experience.



3.3 Base Methods

3.3.3 Try-on Model

- Among the many try-on models, we selected Catvton[28], a newly released, efficient, and excellent model.
- With an edited clothing image we provide, LLM will automatically pass the clothing image into Catvton, so that the virtual model will change into this clothing to better show the effect of clothing editing.

4 PROJECT SCOPE

- Resources Requirement

- Use case scope

4.1 Resources Requirement

- **Hardware proposed for consideration:**
  - Local/cloud system with RTX 3090 GPU
- **Software proposed for consideration**:
  - Front end:
    - Telegram Bot / Web Browser (limited by development time, may not achieve)
  - Middleware:
    - Flask
  - Back end:
    - Relational DB: SQLite or MySQL
    - Pre-trained Model: Stable Diffusion V1-5, Catvton
  - Programming Language:
    - Python
    - SQL

4.2 Use Case Scope

Use Case ID	UC 01
Use Case Name	Getting Recommandation
Description	The user provides clothing preferences or descriptions, and the system returns outfit recommendations along with images of suggested clothing items.
Preconditions	None
Trigger	The user makes a natural language query related to clothing, which is recognized and processed by the system.
Basic Flow	1. The user asks the system for clothing recommendations using natural language. 2.The system captures and processes the input. 3. The system invokes the pre-trained model API to generate and return outfit recommendations and retrieves corresponding images from the database.
Alternative Flows	1a. If the query is not related to clothing, the system prompts the user to refine the request. 2a. If the system fails to understand the user’ s query (e.g., due to unclear or inappropriate language), it prompts the user to rephrase the question. 3a. If there are no suitable images in the database, the system returns only text-based recommendations.
Postconditions	The system successfully generates and delivers recommendations to the user. **OR** The user terminates the process voluntarily.
Special Requirement	Accurate input recognition
Business Rules	None
Open Issues	None

4.2 Use Case Scope

Use Case ID	UC 02
Use Case Name	Picture Uploading
Description	The user can upload their own images of clothing or select images returned by the system. The system stores these images and waits for further instructions from the user (if not provided).
Preconditions	None / The system has already returned relevant images.
Trigger	The user makes a natural language request to upload an image or select one from the system’s suggestions, which is recognized and processed.
Basic Flow	1. The user uploads their own images of clothing/outfits to the system, and reminds the system to store these pictures for the later processing. 2. The system captures and processes the input. 3. The system stores the uploaded image and notifies the user that it is awaiting further instructions.
Alternative Flows	1a. The user can also choose to select pictures in the previous steps for processing. 1b. If the query is not related to clothing, the system prompts the user to refine the request. 2a. If the system cannot understand the user’s request (e.g., unclear or inappropriate language), it prompts the user to rephrase the query. 3a. If the user also provide further instructions, the system will not wait for instructions and will turn to UC 03 or UC04 directly.
Postconditions	The system stores the image and notifies the user that it is ready for additional instructions. **OR** Turn to UC 03 or UC 04 directly. **OR** The user terminates the process voluntarily.
Special Requirement	- Accurate input recognition - Can be integrated into UC03 & UC04 if the user provides instructions along with the image.
Business Rules	Notify the user: "Ensure that the uploaded image meets the system's requirements; otherwise, results may be inaccurate or of low quality."
Open Issues	Handling illegal inputs (e.g., if the user uploads images of clothing that could cause inappropriate outcomes in model-generated images in UC04).

4.2 Use Case Scope

Use Case ID	UC 03
Use Case Name	Clothes Modification (Component-Based Garment Generation)
Description	The user can modify the clothing they have uploaded or selected by providing images of components. The system processes the input and returns images of the updated outfit.
Preconditions	None / The system has already stored images for modification. / The system has already returned relevant images.
Trigger	The user makes a natural language request to modify an outfit with components, which is recognized and processed by the system.
Basic Flow	1. The user requests clothing modifications after uploading pictures of components. 2. The system captures and processes the input. 3. The system calls the pre-trained model API to generate and return the modified outfit images.
Alternative Flows	1a. If the query is unrelated to clothing, the system prompts the user to refine the request. 1b. If the user has not uploaded clothing images before this step, the system notifies the user that it is waiting for relevant images. The system needs to have the ability to remember previous conversations. 2a. If the system fails to understand the user's request (e.g., unclear or inappropriate language), it prompts the user to rephrase the query.
Postconditions	The system successfully generates and delivers modified outfit images to the user. **OR** The user terminates the process voluntarily.
Special Requirement	Accurate input recognition
Business Rules	Notify the user: "Ensure that the uploaded image meets the system's requirements; otherwise, results may be inaccurate or of low quality."
Open Issues	- Currently, only accessory modifications are supported. - Consider developing functionality to allow users to modify clothing using natural language commands. - Explore adding features for modifying other aspects such as colors or patterns.

4.2 Use Case Scope

Use Case ID	UC 04
Use Case Name	Model Dress Map Generation (Virtual Try-On)
Description	The user can request the system to generate a model wearing the designated clothing. The user may specify the clothing items, the model, or even use themselves as the model. The system processes the input and returns images of the model dressed in the selected clothing.
Preconditions	None / The system has already stored images for modification. / The system has previously returned relevant images.
Trigger	The user makes a natural language request to generate a model dress map, which is recognized and processed by the system.
Basic Flow	1. The user requests model dress map generation using natural language. 2.The system captures and processes the input. 3.The system calls the pre-trained model API to generate and return the model dress map.
Alternative Flows	1a. If the query is unrelated to clothing, the system prompts the user to refine the request. 1b. If the user has not uploaded images prior to this step, the system notifies the user and waits for the relevant images. The system should also remember prior interactions to facilitate this process. 2a. If the system cannot understand the user’ s query (e.g., due to unclear or inappropriate language), it prompts the user to rephrase the request.
Postconditions	The system successfully generates and delivers the model dress map to the user. **OR** The user voluntarily terminates the process.
Special Requirement	Accurate input recognition
Business Rules	Notify the user: "Ensure that the uploaded image meets the system's requirements; otherwise, results may be inaccurate or of low quality."
Open Issues	None

5 DATA COLLECTION AND PREPARATION

- Data Collection

- Data Preparation

5.1 DATA COLLECTION

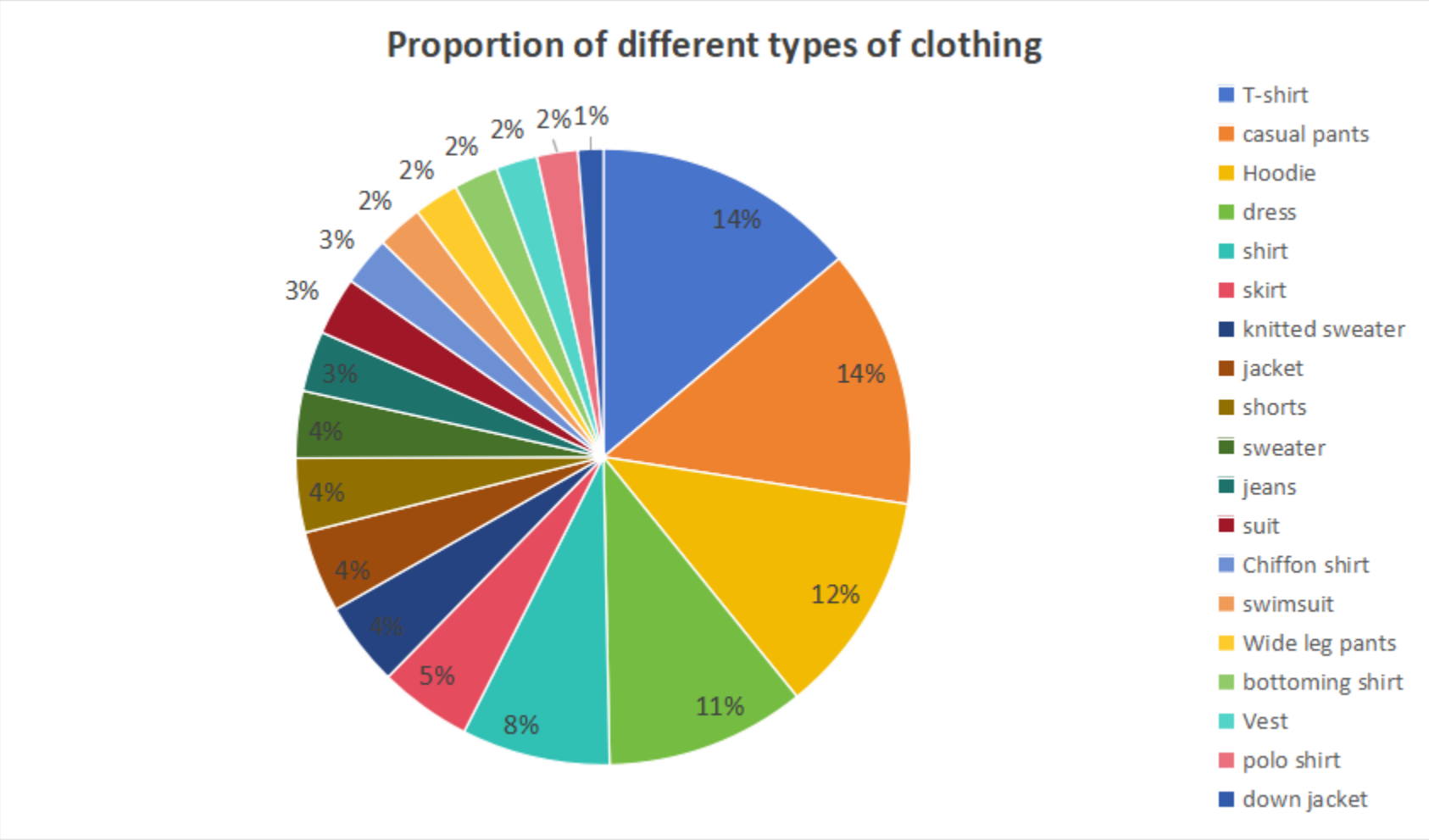
Considering the need to use a certain amount of clothing image datasets, we chose to use the CM-Fashion clothing dataset.

The dataset used in our project contains 444,214 clothing images and their corresponding text descriptions, spanning a total of 49 different clothing categories.

19 clothing categories have a data volume greater than 5,000 images, accounting for a total of 402,933 images, which represents about 90% of the total data volume.

Its data distribution diagram is shown in next page

5.1 DATA COLLECTION

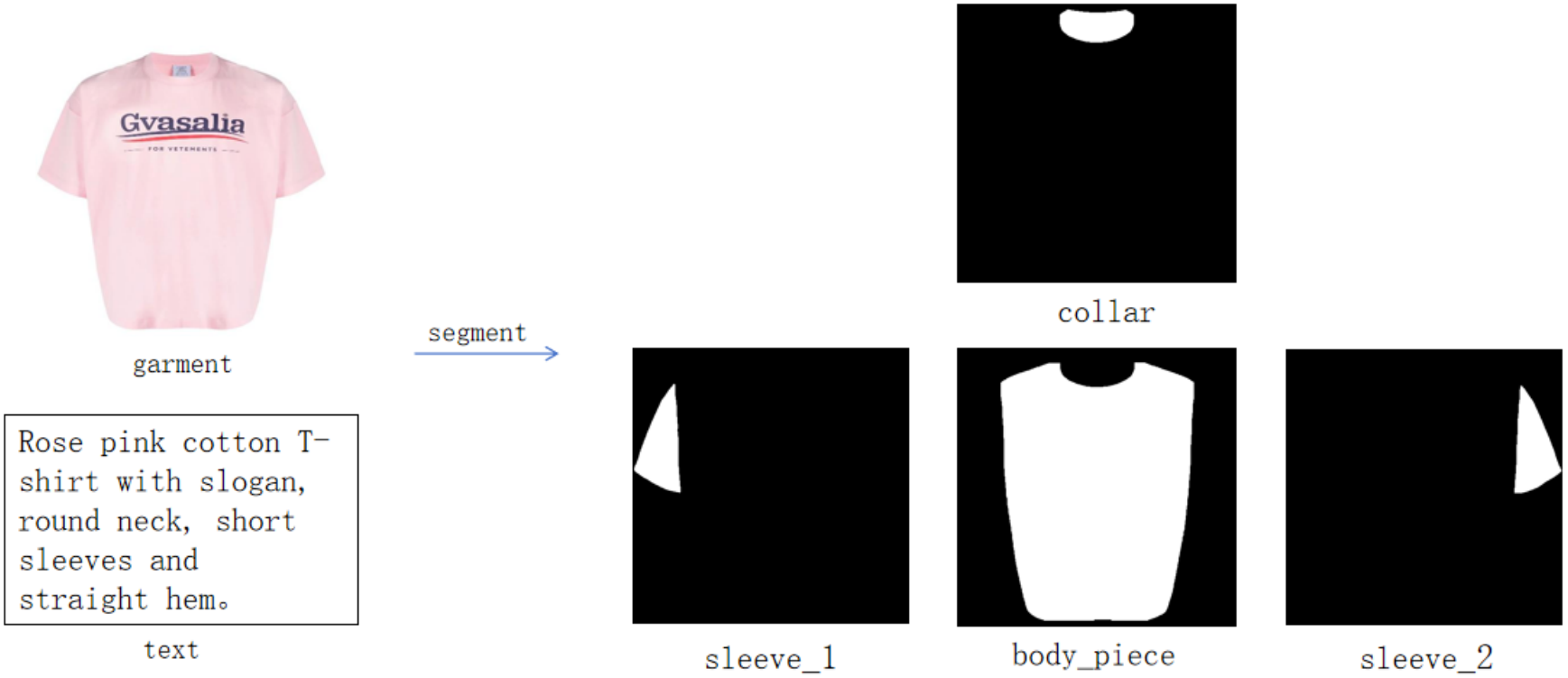


According to the data distribution, the CM-Fashion dataset contains the most data of T-shirts, slacks, hooded shirts and dresses, accounting for 50% of the total data. Overall, tops accounted for more than pants and skirts.

5.2 DATA PREPARATION

5.2.1 MASK GENERATION

- Use Detectron2's PointRend[29] to generate masks for clothing parts to achieve the purpose of getting segmentation of clothing parts.



5.2 DATA PREPARATION

5.2.2 DATA AUGMENTATION

- Obtain segmentation maps $S=\{S_1,S_2,\ldots,S_n\}$ for different parts of an original image *p.*
- Randomly select several segmented parts for data augmentation. For the parts S_i that need augmentation, take their complement in S:

$$S_{aug} = S - S_i$$

- In S_{aug}, randomly change contrast and brightness for the parts. With a 10% probability, set the part to null. Then, add the processed part to the original part:

$$S_i' = S_i + \sum_{j=1}^{n-1} E\left(S_j\right)$$

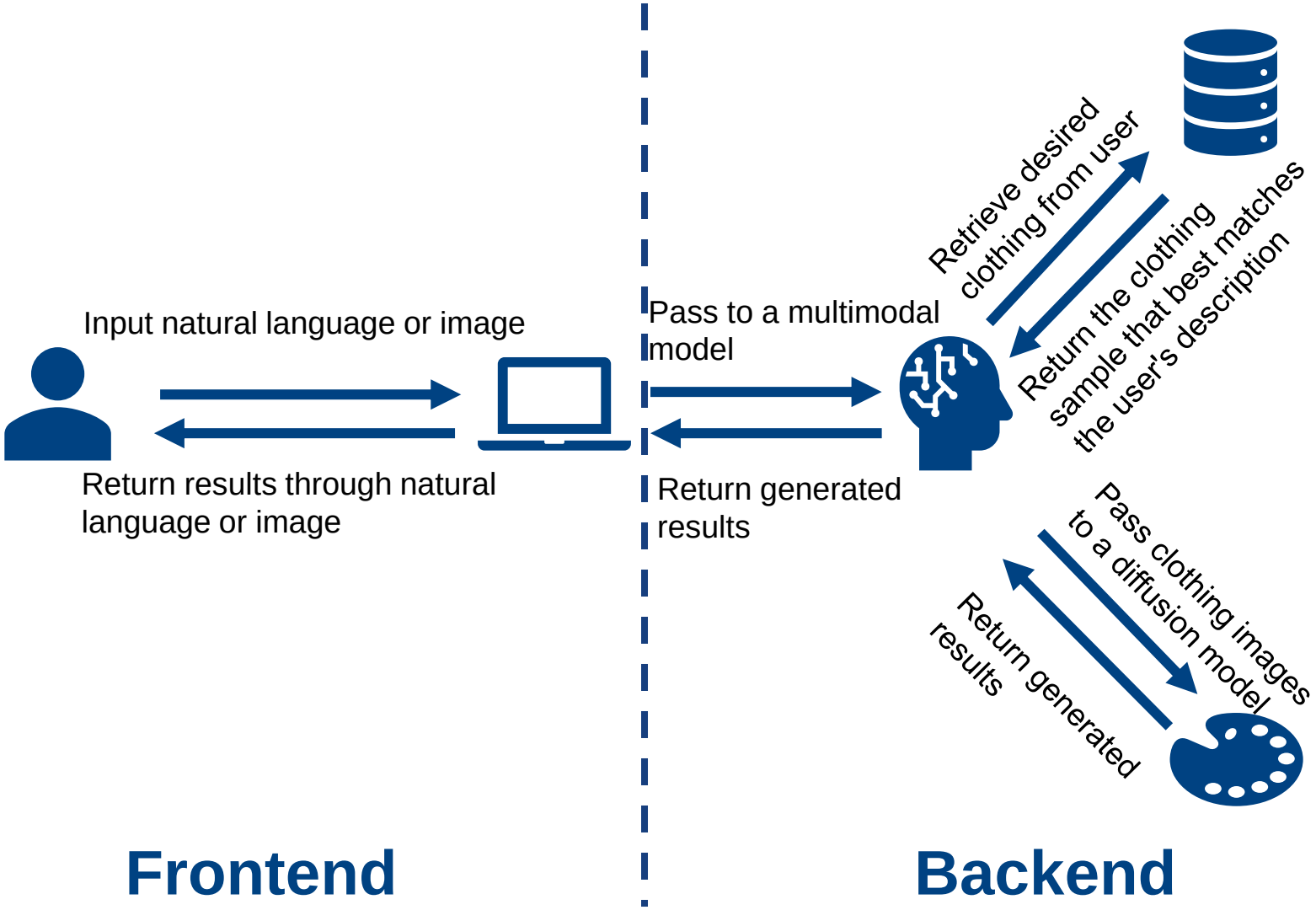
- With $E(S) = \begin{cases} \mathbf{convertScaleAbs}(S), p = 0.9 \\ null, \quad p = 0.1 \end{cases}$

*The convertScaleAbs function is used to change the contrast and brightness of the image

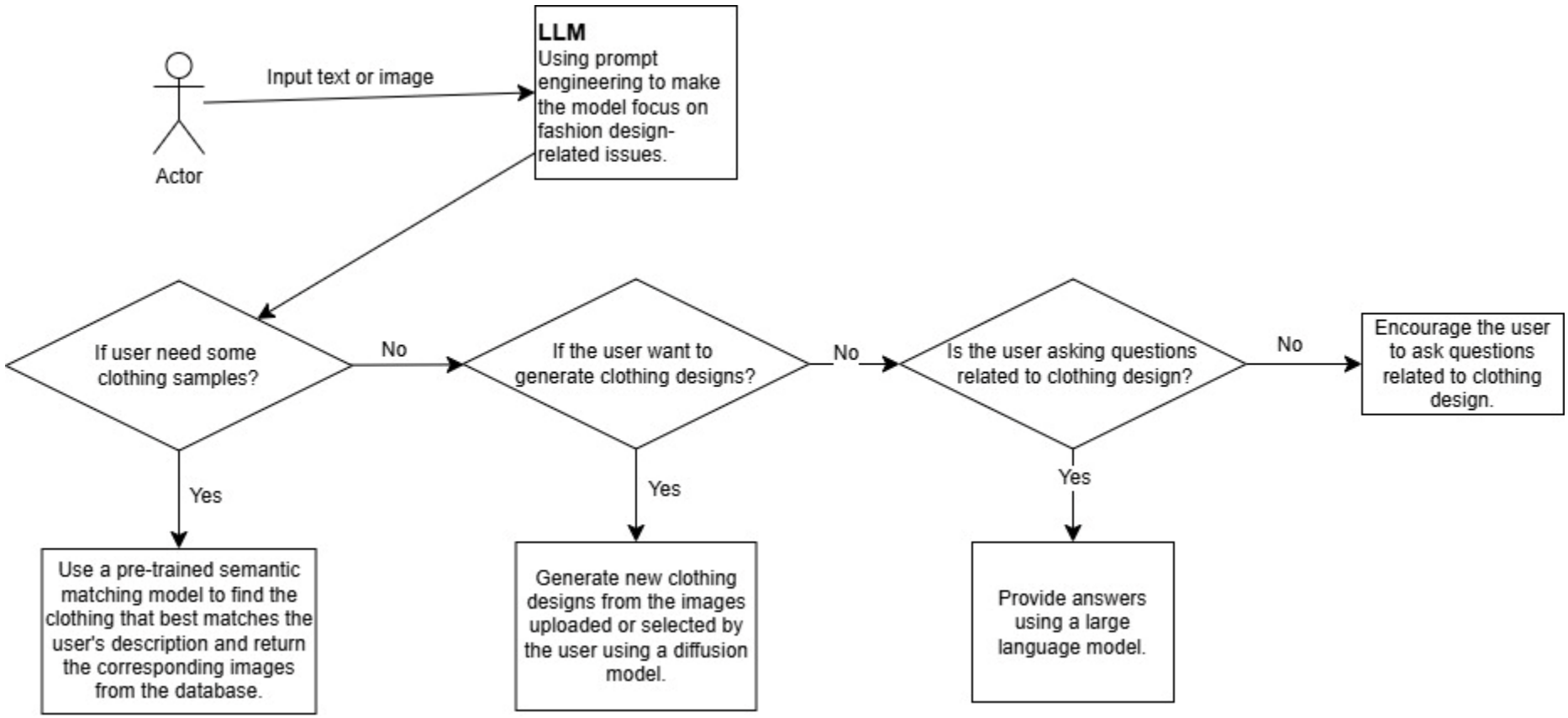
6 SYSTEM DESIGN

- Basic System Design
- System Workflow Design
- System Architecture

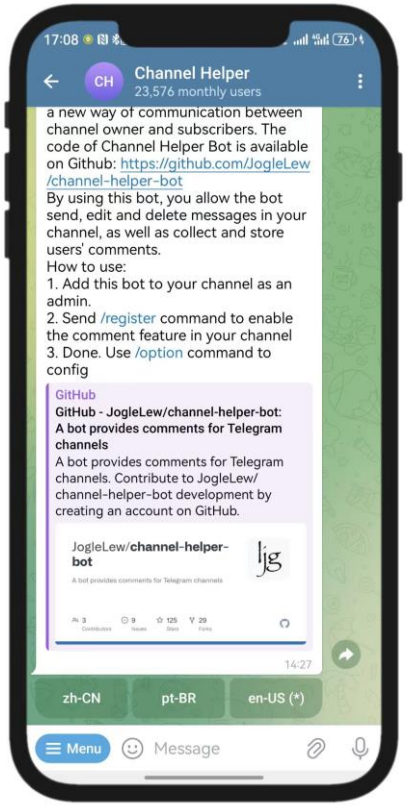
6.1 Basic System Design Introduction



6.2 System Workflow Design



6.3 System Architecture Design



Frontend

Telegram Bot

Database: SQL server

Drawing Model: Stable Diffusion V1-5, Catvton

LLM: Yi-Large

Semantic Matching Model: NLTK, Scikit-Learn, Sentence-BERT

7 REFERENCE

Reference

[1] Zhang X, Sha Y, Kampffmeyer M C, et al. Armani: Part-level garment-text alignment for unified cross-modal fashion design[C]//Proceedings of the 30th ACM International Conference on Multimedia. 2022: 4525-4535.

[2] McKinsey & Company. (2023). Generative AI: Unlocking the future of fashion. Retrieved from https://www.mckinsey.com/industries/retail/our-insights/generative-ai-unlocking-the-future-of-fashion

[3]Intelistyle. (2023). Fashion AI in 2023: What should we expect to see this year?. Retrieved from https://intelistyle.com/fashion-ai-in-2023-what-should-we-expect-to-see-this-year/

[4] Toran Bruce Richards. Auto-gpt: An autonomous gpt-4 experiment, 2023.

[5] Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. arXiv preprint arXiv:2303.17580.

[6] Zeqiang L, Xizhou Z, Jifeng D, et al. Mini-dalle3: Interactive text to image by prompting large language models[J]. arXiv preprint arXiv:2310.07653, 2023.

[7] Zhang, Z., Zhang, A., Li, M., & Smola, A. (2022). Automatic chain of thought prompting in large language models. arXiv preprint arXiv:2210.03493.

[8] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629.

[9] Xie Z, Ding H, Li M, et al. Hierarchical Fashion Design with Multi-stage Diffusion Models[J]. arXiv preprint arXiv:2401.07450, 2024.

[10] Ning S, Wang D, Qin Y, et al. PICTURE: PhotorealistIC virtual Try-on from UnconstRained dEsigns[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 6976-6985.

Reference

[11] Li N, Liu Q, Singh K K, et al. UniHuman: A Unified Model For Editing Human Images in the Wild[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 2039-2048.

[12] Zhang X, Yang B, Kampffmeyer M C, et al. Diffcloth: Diffusion based garment synthesis and manipulation via structural cross-modal semantic alignment[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 23154-23163.

[13] He K, Yao K, Zhang Q, et al. DressCode: Autoregressively Sewing and Generating Garments from Text Guidance[J]. ACM Transactions on Graphics (TOG), 2024, 43(4): 1-13.

[14] Baldrati A, Morelli D, Cartella G, et al. Multimodal Garment Designer: Human-Centric Latent Diffusion Models for Fashion Image Editing Supplementary Material[J].

[15] Seyfioglu M S, Bouyarmane K, Kumar S, et al. DreamPaint: Few-Shot Inpainting of E-Commerce Items for Virtual Try-On without 3D Modeling[J]. arXiv preprint arXiv:2305.01257, 2023.

[16] Rombach R, Blattmann A, Lorenz D, et al. High-resolution image synthesis with latent diffusion models[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10684-10695.

[17] Yu J, Zhang L, Chen Z, et al. Quality and Quantity: Unveiling a Million High-Quality Images for Text-to-Image Synthesis in Fashion Design[J]. arXiv preprint arXiv:2311.12067, 2023.

[18] Li B, Qi X, Lukasiewicz T, et al. Manigan: Text-guided image manipulation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 7880-7889.

[19] Wang B, Zheng H, Liang X, et al. Toward characteristic-preserving image-based virtual try-on network[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 589-604..

Reference

[20] Xie Z, Huang Z, Zhao F, et al. Pasta-gan++: A versatile framework for high-resolution unpaired virtual try-on[J]. arXiv preprint arXiv:2207.13475, 2022.

[21] Fu J, Li S, Jiang Y, et al. Stylegan-human: A data-centric odyssey of human generation[C]//European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022: 1-19.

[22] Jiang Y, Yang S, Qiu H, et al. Text2human: Text-driven controllable human image generation[J]. ACM Transactions on Graphics (TOG), 2022, 41(4): 1-11.

[23] Zhang K, Sun M, Sun J, et al. Humandiffusion: a coarse-to-fine alignment diffusion framework for controllable text-driven person image generation[J]. arXiv preprint arXiv:2211.06235, 2022.

[24] Pernuš M, Fookes C, Štruc V, et al. Fice: Text-conditioned fashion image editing with guided gan inversion[J]. arXiv preprint arXiv:2301.02110, 2023.

[25] Zhang L, Rao A, Agrawala M. Adding conditional control to text-to-image diffusion models[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 3836-3847.

[26] Zhao S, Chen D, Chen Y C, et al. Uni-controlnet: All-in-one control to text-to-image diffusion models[J]. Advances in Neural Information Processing Systems, 2024, 36.

[27] Mou C, Wang X, Xie L, et al. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(5): 4296-4304.

[28] Chong Z, Dong X, Li H, et al. CatVTON: Concatenation Is All You Need for Virtual Try-On with Diffusion Models[J]. arXiv preprint arXiv:2407.15886, 2024.

[29] Kirillov A, Wu Y, He K, et al. Pointrend: Image segmentation as rendering[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 9799-9808.