Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

TAILOR
Text to image AI generator

By:

Amgad Abd El Baset	CS Department
Andrew Nassef Amin	CS Department
Andre Adel Ibrahim	CS Department
Maria Maurice Ayoub	CS Department
Mayve Ehab Rasmy	CS Department
Mina Sami Anwar	CS Department

Under Supervision of:

Dr. Salsabil Amin
Basic Sciences Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

T.A. Mohamed Essam
Bioinformatics Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

July 2023

Acknowledgement

We offer our sincerest gratitude to our supervisors, Dr.Salsabil Amin and T.A. Mohamed Essam who supported us throughout our thesis with their patience, knowledge and experience.

Abstract

T*AI*LOR AI Art Generator is a mobile application designed to provide users with the ability to generate unique and creative fashion designs using artificial intelligence.

In recent times, there has been a growing interest in creating models that can generate high-quality images from input text. We present a new model that uses a large clothing dataset to generate images of casual clothes and wedding dresses, leveraging stable diffusion finetuning and state-of-the-art language models.

The text-to-image generator project is essential because it enables the creation of high-quality images from input text, which can have multiple practical applications. Our model is focused on the fashion industry which could use this technology to generate images of clothing items to showcase their designs to customers or create virtual try-on systems.

To ensure coherence between the input text and the generated images, we fine-tuned our model on a vast dataset of clothing images. Our model can produce visually appealing images of clothing items that are semantically coherent with the input text. Our evaluations indicate that our approach outperforms other text-to-image models.

Our model's flexibility allows it to produce images of diverse clothing items, including shirts, dresses, and other types of apparel. Our user study reveals that the generated images are of high quality and accurately represent the input text.

While our model has limitations, such as its fixed resolution, we suggest exploring techniques to generate images at various resolutions and using GANs to enhance image quality. In conclusion, our model represents a significant step towards generating high-quality images of clothing items and has the potential to be useful in fashion design and virtual try-on systems.

Table of Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
APK	Android Package Kit
cGANs	Conditional Generative Adversarial Networks
CNNs	Convolutional Neural Networks
FID	Fréchet Inception Distance
NLP	Natural Language Processing
RNNs	Recurrent Neural Networks
T5-Transformer	A Transformer-based language model
UI	User Interface
UNDP	United Nations Development Programme

.

1- Introduction

1.1 Motivation

Time and cost savings: the average cost of a wedding dress design depending on whom you select to make your dress but typically it can range from $2,000 - $8,000, and the average cost of casual wear design $200 - $250, which can be a significant expense for many people. By providing a tool that allows users to create their own unique designs, that can help save them time and money.
Customization: According to a survey by The Wedding Report, 83% of brides want a personalized wedding dress. By using an AI text to image generator, users can create a dress that is tailored to their specific preferences and style, rather than having to settle for a pre-made design.
Accessibility: Not everyone has access to high-end fashion designers or the budget to purchase expensive clothing. By providing a tool that allows users to create their own designs, you can help make fashion more accessible to a wider range of people.
Additionally, according to a report by the UNDP, Egypt has a high youth unemployment rate, which can make it difficult for young people to afford high-end fashion or to find employment in the fashion industry. By providing a tool that allows users to create their own designs, your AI art generator can help make fashion more accessible and provide opportunities for young people to develop their skills and creativity.

1.2 Problem Definition

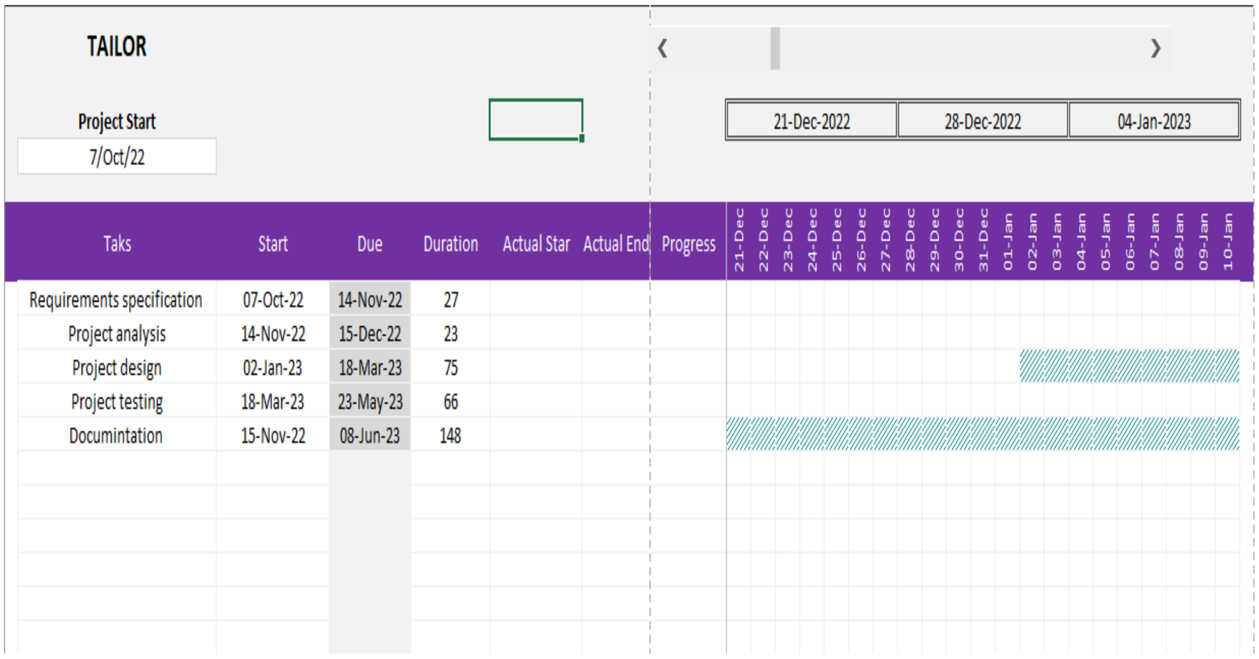
The need for an efficient and accessible way for users to generate custom images of wedding dresses or casual wear based on their own text inputs. Traditionally, designing custom clothing can be a time-consuming and expensive process, requiring the skills of a professional designer or costly software. This project aims to provide a solution that is accessible to a wider range of users, allowing them to generate unique and personalized clothing designs quickly and easily. By using AI technology to generate these designs, the project also aims to improve the accuracy and consistency of the design process, reducing the need for manual adjustments and revisions.

1.3 Objective

Build a maintainable and usable application that can take clothes descriptions as text and generate a new image containing the clothes with the desired description that the user provided in the text input**.**
To achieve these objectives, the following steps were taken:

1. Research and development of the AI model and algorithms that can generate high-quality images based on user inputs.
2. Collection and analysis of a large dataset of wedding dress and casual wear images to train the AI model.
3. Design and development of a user-friendly interface that allows users to input their desired design specifications and view the generated images.
4. Testing and refinement of the AI model and user interface to ensure that the generated images are accurate and visually appealing.
5. Integration of the ability for users to save and download their generated designs.

1.4 Time Plan



TAILOR

Project Start
7/Oct/22

21-Dec-2022 | 28-Dec-2022 | 04-Jan-2023

Taks	Start	Due	Duration	Actual Star	Actual End	Progress	21-Dec	22-Dec	23-Dec	24-Dec	25-Dec	26-Dec	27-Dec	28-Dec	29-Dec	30-Dec	31-Dec	01-Jan	02-Jan	03-Jan	04-Jan	05-Jan	06-Jan	07-Jan	08-Jan	09-Jan	10-Jan
Requirements specification	07-Oct-22	14-Nov-22	27																								
Project analysis	14-Nov-22	15-Dec-22	23																								
Project design	02-Jan-23	18-Mar-23	75																▨	▨	▨	▨	▨	▨	▨	▨	▨
Project testing	18-Mar-23	23-May-23	66																								
Documintation	15-Nov-22	08-Jun-23	148				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Figure 1-1- Time Plan

1.5 Document Organization

Chapter 2: the background of text-to-image generation, which is a field of research in artificial intelligence (AI). Over the past 20 years, advancements in deep learning, computer vision, and natural language processing have led to the development of complex AI models and applications, including text-to-image models. These models can generate images from a text prompt and have recently seen a rise in development. The chapter then introduces diffusion, which is the foundation of text-to-image models. Diffusion involves training a model to denoise a noisy image and return it to its original condition. By including a text prompt to the noisy image through a text encoder, the model can be trained to denoise the image guided by the text provided. The chapter also describes the decoding model, which attempts to decode the noise into an original representation, producing a new image. Finally, the chapter mentions existing similar systems, such as DALL·E 2 from Open AI, which can create realistic images and art from a text description and combine concepts, attributes, and styles.

Chapter 3: the document titled "Analysis and Design" discusses the analysis and design phase of the project. In section 3.1, the system overview is provided, which includes a system architecture diagram and a description of the intended users and their characteristics. The system is designed for fashion designers, brides, and anyone who has a keen sense of fashion and can describe their model. The user characteristics include being creative, imaginative, having a sense of fashion, being descriptive, and being able to use and understand modern technology like mobile phones or browsers.
In section 3.2, the system analysis and design are discussed in detail. This includes a use case diagram and fully dressed use cases that describe each function of the project. A class diagram is also provided, which includes screens classes, widget classes, and provider classes. Additionally, a sequence diagram is presented to show the sequence of actions that occur when a user interacts with the system. Lastly, a database diagram is included to illustrate the database schema and description of tables if the system includes a database.

Chapter 4: of the document titled "Implementation and Testing" describes the implementation and testing phase of the project. The system functions are detailed, including user registration and login, text input, image generation, image storage, image display, and image saving and sharing. Additionally, the techniques and algorithms implemented are discussed, such as deep learning models like CNNs and RNNs, cGANs diffusion model, and NLP techniques.
New technologies used in the implementation are also described, including Flutter framework for mobile application development, TensorFlow for building and training deep learning models, and Firebase for user authentication, real-time database storage, and cloud storage for saving and retrieving generated images. The UI design and wireframes are also included, with a focus on the login and registration screens, home screen, image display screen, and history screen.

Chapter 5: the user manual provides a step-by-step guide on how to use the tAIlor mobile application to generate unique and creative fashion designs using artificial intelligence. The chapter starts with an installation guide, which outlines the steps to download and install the application on your mobile device. The chapter then provides instructions on how to sign up with your Google account, then guides the user on how to use the application, including entering text and selecting a category for the fashion design. The user is then instructed to submit their request and wait for the results to appear. Once the results are generated, the user can view and save the image to their gallery or previous work. The chapter also includes instructions on how to access your history of saved images and how to log out of the application.

Chapter 6: of the document titled "Conclusion and Future Work" provides a summary of the entire project and the results obtained. The virtual clothing customization system presented in this project successfully integrates natural language processing and image generation techniques to provide users with a seamless and interactive platform to personalize clothing items based on their preferences. The diffusion model and algorithm employed in the project have demonstrated impressive performance in generating clothing images that align with the provided textual descriptions.

2- Background

Text to image generation

Over the past 20 years, artificial intelligence (AI) research has increased dramatically, with advancements in fields like deep learning, computer vision, and natural language processing leading to the development of ever-more complex AI models and applications. text-to-image models, which are AI models that can generate images from a text prompt, have recently seen a spike in development. As the study of these models advances, the models become more complex and can produce stunning pictures that may be applied in a variety of contexts.

Diffusion

Diffusion is where the science of text-to-image models begins. By gradually and frequently introducing noise to an image, diffusion involves training a model to take the noise out of the image and return it to its original condition. The model can learn to denoise a noisy image until it resembles one of the images in the initial training set by starting with a noisy image. The goal of the model is to convert the noise into a picture that closely reflects the statistics and patterns present in real images.

However, when including a text prompt to the noisy image through a text encoder, the model can be trained to denoise the image-guided by the text provided. The model learns to generalize text into images by converting the noisy image back to its original representation. In this way, the model learns how to match a text prompt to an image.

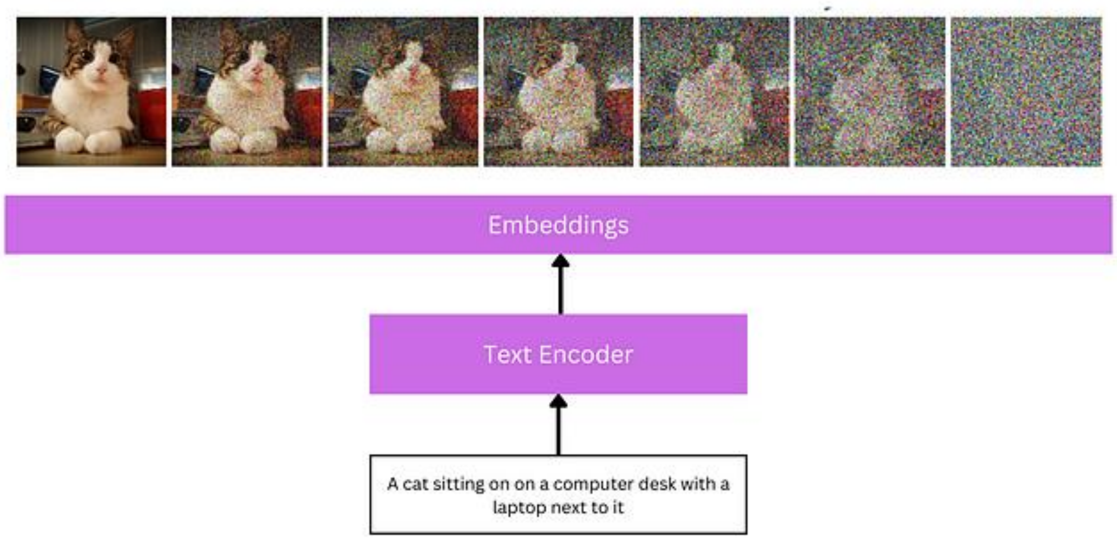

Embeddings
Text Encoder
A cat sitting on on a computer desk with a laptop next to it

Figure 2-1- Applying a text prompt to a noisy image

The decoding model will attempt to decode the noise into an original representation, producing a new image, starting with random noise and an unread bit of text. The original image will be modest, but it can be improved (upscaled) with additional models to add fine details and a greater resolution.

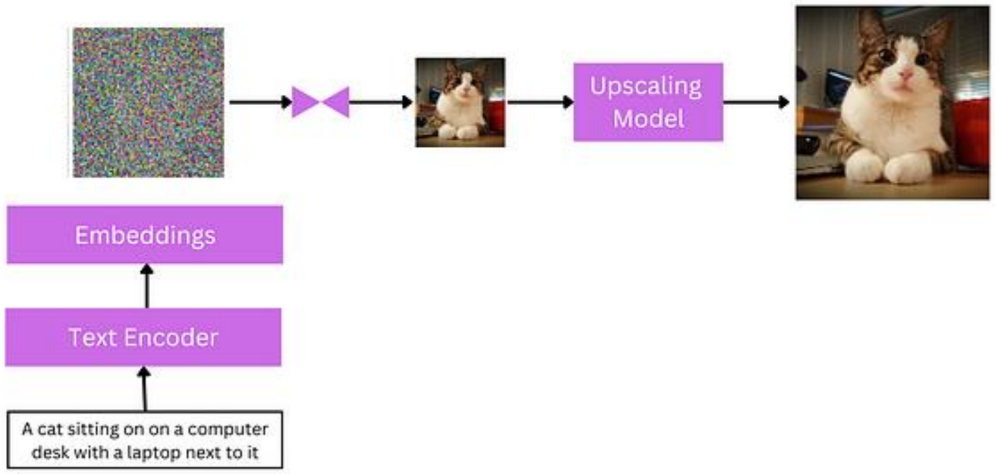

Upscaling Model
Embeddings
Text Encoder
A cat sitting on on a computer desk with a laptop next to it

Figure 2-2- Decoding a noisy image based on a text prompt and upscaling to a larger size

Forward Diffusion Process
The forward diffusion process adds Gaussian noise to the input image step by step. Nonetheless, it can be done faster using the following closed-form formula to directly get the noisy image at a specific time step *t*:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$$

Reverse Diffusion Process
Since the reverse diffusion process is not directly computable, we train a neural network *εθ* to approximate it.
The training objective (loss function) is as follows:

$$\boxed{x_t = \sqrt{\bar{a}_t}\, x_0 + \sqrt{1 - \bar{a}_t}\, \varepsilon}$$

$$L_{\text{simple}} = \mathbb{E}_{t, x_0, \varepsilon}\left[\left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\boxed{x_t}, t) \right\|^2 \right]$$

Stable Diffusion

Most of the recent AI art found on the internet is generated using the Stable Diffusion model. Since it is an open-source tool, any person can easily create fantastic art illustrations from just a text prompt.
The original name of Stable Diffusion is “Latent Diffusion Model” (LDM). As its name points out, the Diffusion process happens in the latent space. This is what makes it faster than a pure Diffusion model.

Departure to Latent Space

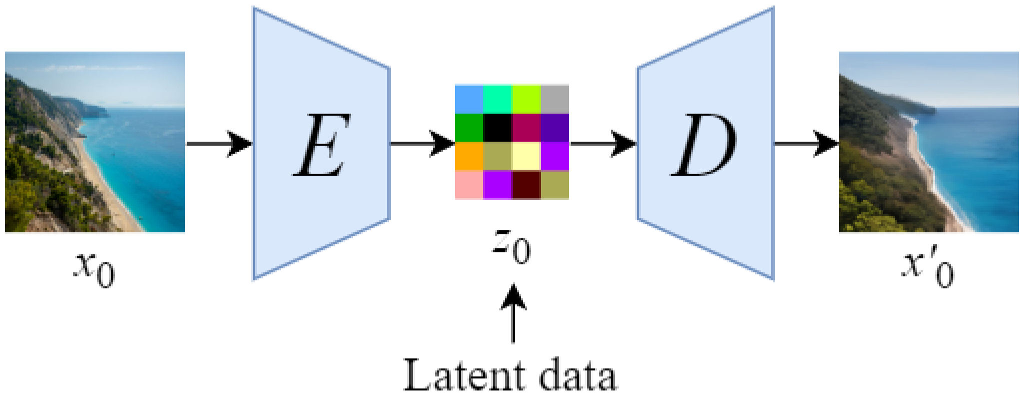


Figure 2-3- Autoencoder

Conditioning

The true power of the Stable Diffusion model is that it can generate images from text prompts. This is done by modifying the inner diffusion model to accept conditioning inputs.

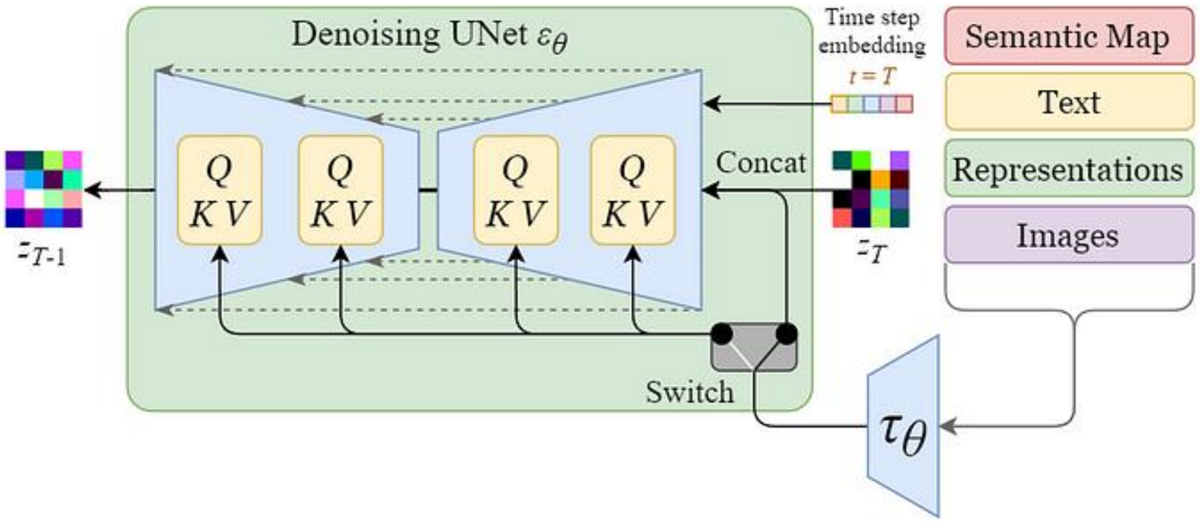


Figure 2-4- Conditioning mechanism details

To quickly summarize:

- Stable Diffusion (Latent Diffusion Model) conducts the diffusion process in the latent space, and thus it is much faster than a pure diffusion model.
- The backbone diffusion model is modified to accept conditioning inputs such as text, images, semantic maps, etc.

Existing similar systems

1) DALL·E 2 from Open AI: DALL·E 2 can create original, realistic images and art from a text description. It can combine concepts, attributes, and styles.

2) MidJourney: An independent research lab that produces a proprietary artificial intelligence program that creates images from textual descriptions.

Figure 2-5- Drawing made by Midjourney that won a contest.

3) Stable Diffusion: An Artificial Intelligence (AI) Art Generator app that allows you to create your own amazing artworks.

3- Analysis and Design

3.1 System Overview

3.1.1 System Architecture

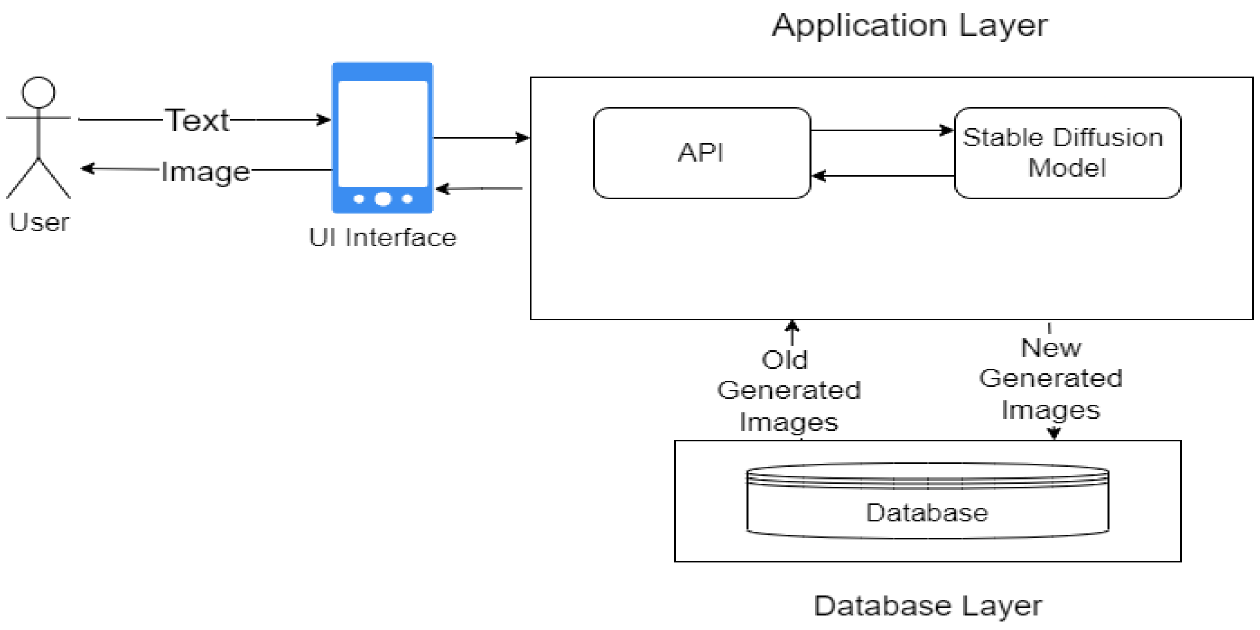


Figure 3-1- System Architecture

3.1.2 System Users

A. Intended Users:

- Fashion Designers: Designers can use our program to help them design new creative and distinctive clothing and give them new ideas including but not exclusive: shirts, shoes, pants, dresses…etc.
- Brides: Any bride that has a creative sense of fashion and wants to imagine what her wedding dress would look like or has a specific design in mind can just type the description she wants.
- Other People: Technically, any user that has keen sense of fashion or has a fashion design in mind that he can describe the model will generate it.

B. User Characteristics

- Creative.
- Pays attention to details.
- Has a sense of fashion.
- Imaginative.
- Can use and understand modern technology like mobile phones or browsers.
- Descriptive.

3.2 System Analysis & Design

3.2.1 Use Case Diagram

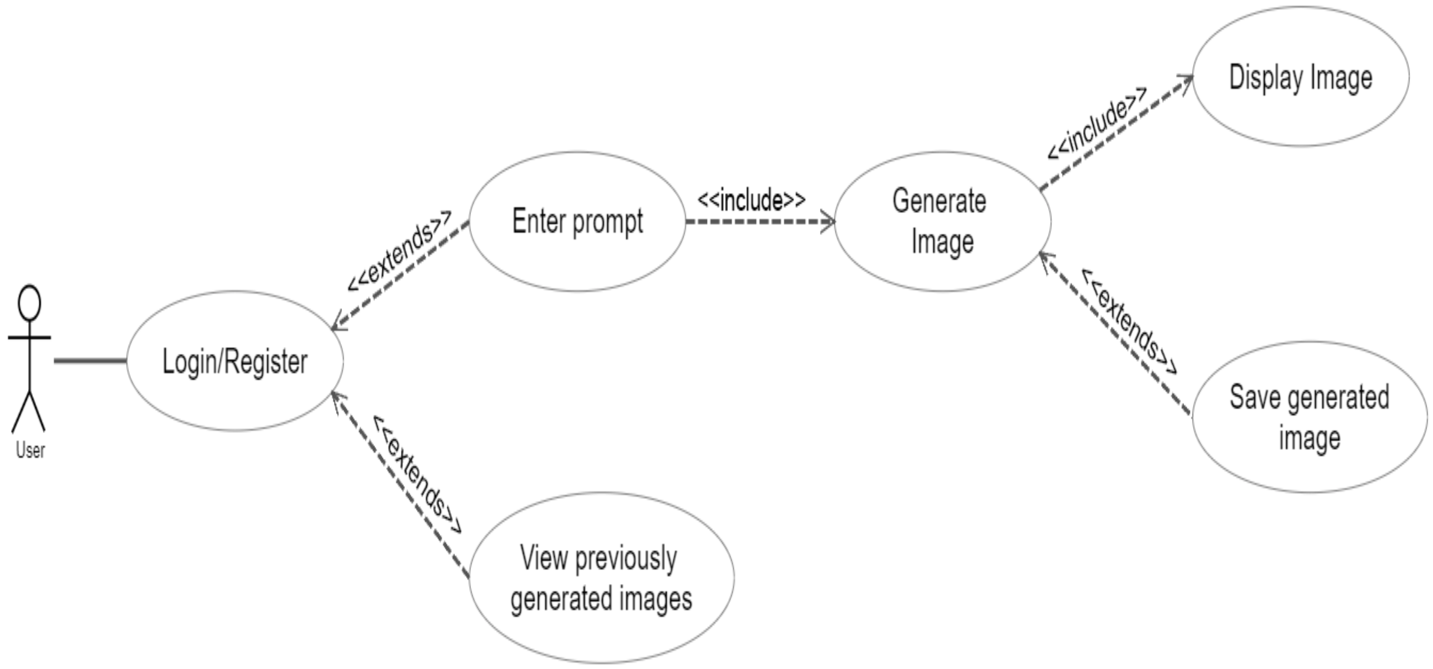


Figure 3-2- Use Case Diagram

3.2.2 Class Diagram

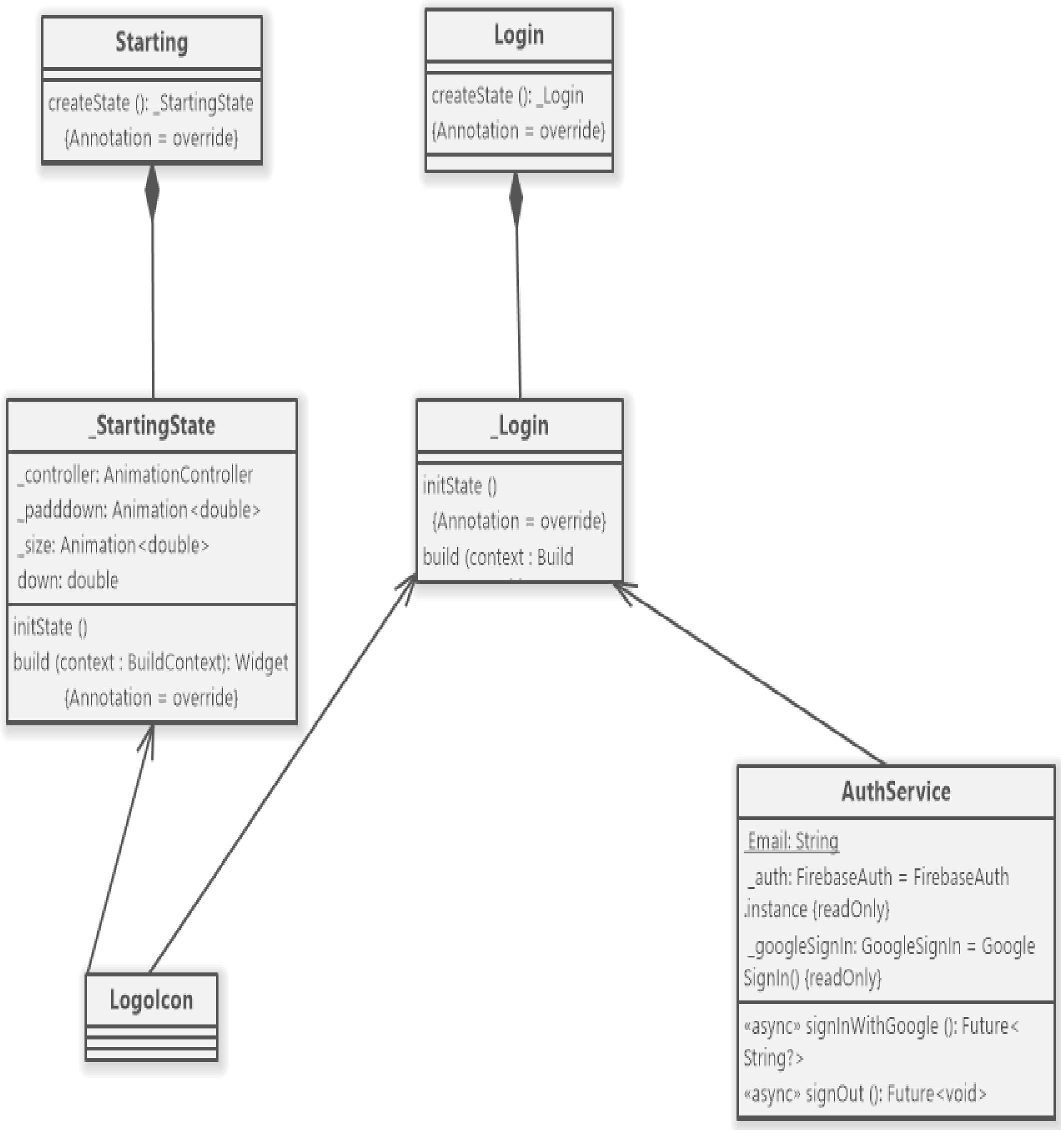

Starting
createState (): _StartingState
{Annotation = override}
Login
createState (): _Login
{Annotation = override}
_StartingState
_controller: AnimationController
_padddown: Animation<double>
_size: Animation<double>
down: double
initState ()
build (context : BuildContext): Widget
{Annotation = override}
_Login
initState ()
{Annotation = override}
build (context : Build
AuthService
Email: String
_auth: FirebaseAuth = FirebaseAuth .instance {readOnly}
_googleSignIn: GoogleSignIn = Google SignIn() {readOnly}
«async» signInWithGoogle (): Future< String?>
«async» signOut (): Future<void>
LogoIcon

Figure 3-3- Class Diagram(1)

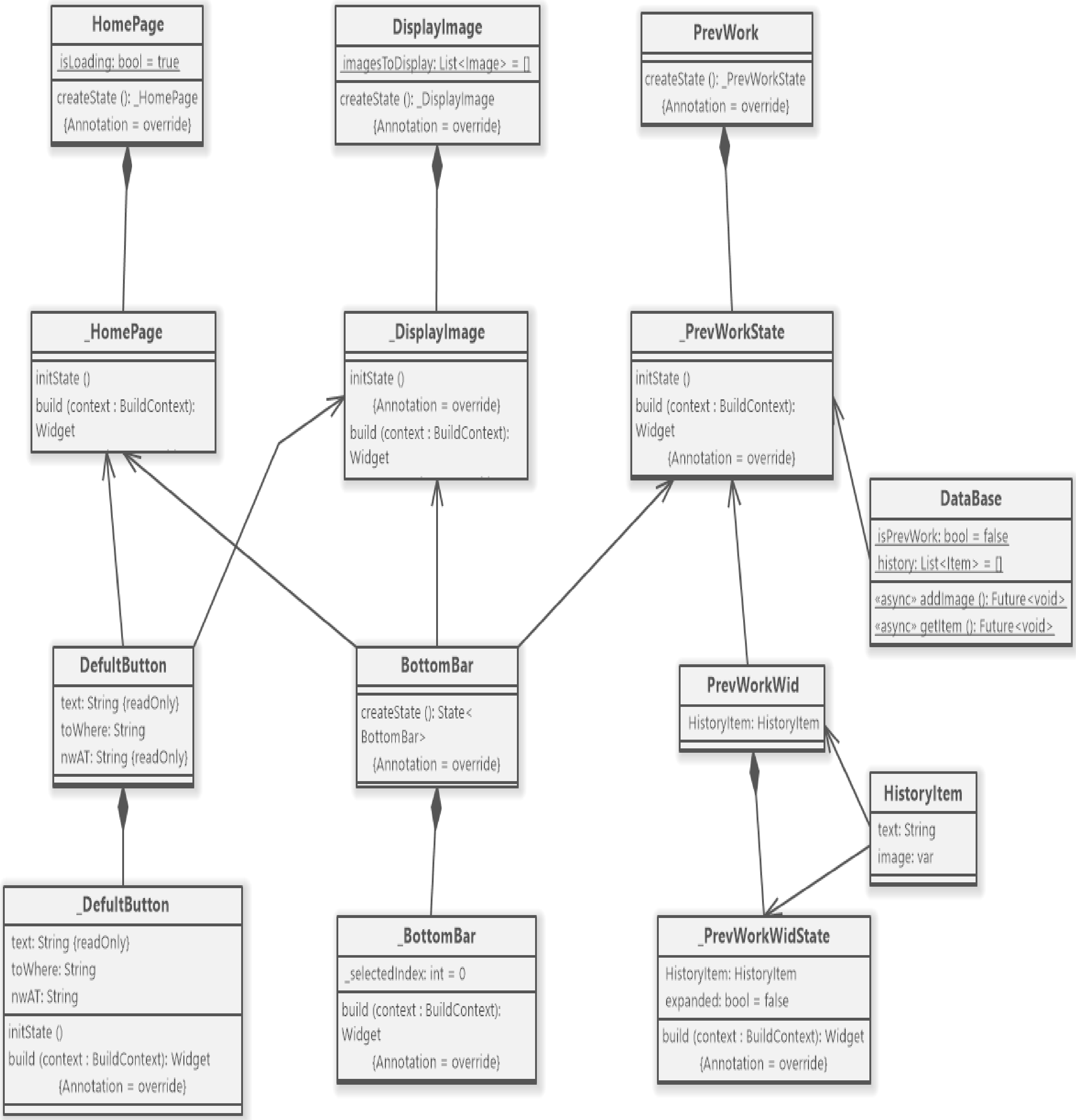


Figure 3-4- Class Diagram(2)

Screens classes:

The Starting class: This class likely contains methods for displaying the animation and checking the user's login status.

The Login class: This class likely contains methods for handling the login process and communicating with the AuthService.

The HomePage class: This class likely contains methods for handling user input and communicating with other classes to generate the designs.

The DisplayImage class: This class likely contains methods for communicating with the API and database, as well as methods for displaying the images to the user.

The PrevWork class is responsible for displaying the user's previously saved work.

Widget classes: including BottomBar, DefultButton, Logo, and PrevWorkWidget
provider classes: including AuthService, DataBase, and historyItem.

4- Implementation and Testing

Datasets:

1) First Dataset:
   - fashion product dataset, Kaggle. 21K of fashion products images divided into two subcategories (top wear, bottom wear)
   - 10 features describing the product such as gender, base color, year,
2) Second Dataset:
   - 500 collected wedding dresses images with detailed description.
   - Preprocessing done on images: face segmentation, background.
   - Preprocessing done on text: summarize the text prompt by getting the most important sentences in the description, enter the output of step 1 to T5-transformer to summarize the description according to the meaning of the entered sentences.

System Function:

- **User Login with Google**: Users can create an account by providing their details and logging in using their credentials.
  user using their Google account. It first prompts the user to select their account using GoogleSignIn class, and then retrieves the authentication tokens using the authentication property of the GoogleSignInAccount class. It then creates an AuthCredential object using the retrieved tokens and signs in the user using signInWithCredential() method of the FirebaseAuth instance.

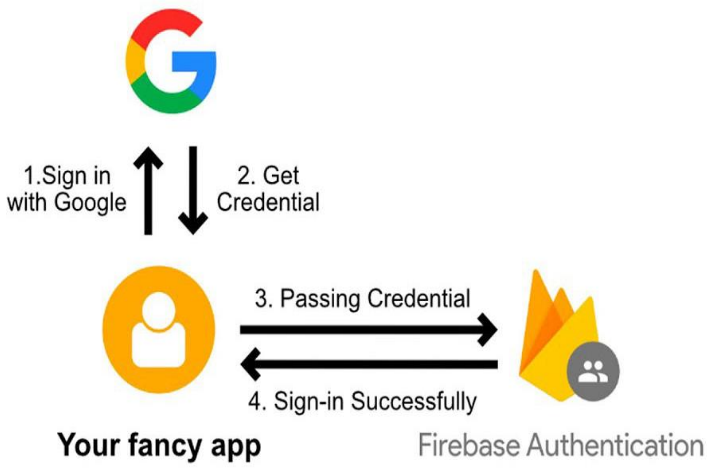


Figure 4-1- Flutter Firebase Auth Login

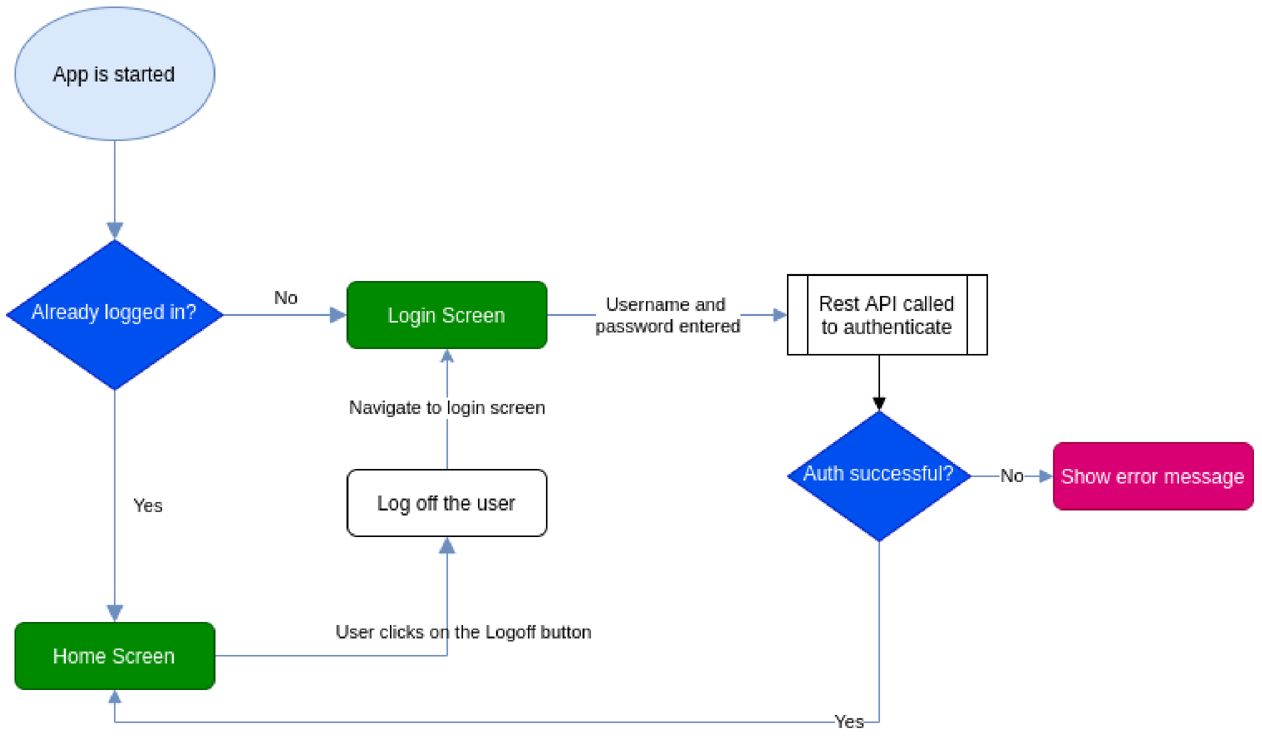


Figure 4-2- Sign in with google

- **Text Input**: Users can enter their desired clothing description as text input, specifying various attributes such as color, style, fabric, and pattern.

- **Image Generation:** The system utilizes AI algorithms to generate images based on the provided text input. These algorithms convert the text description into visual representations of the clothing items by connecting the app to the API of the model.

  1. The Flutter app sends a request to an API endpoint to retrieve some data.
  2. The API endpoint processes the request and sends back a response containing the requested data.
  3. The response is received by the Flutter app and parsed into a model object.
  4. The model object is used by the Flutter app to display the retrieved data in the user interface.

- **Image Storage**: The generated images are stored in a database for future retrieval and display.
  It first retrieves the email of the authenticated user using the AuthService class. The function then iterates over the list of image paths and converts the images to a format that can be stored in the Firebase Storage. The function then uploads the images to the Firebase Storage and retrieves their download URLs. Finally, the function stores the download URLs and any corresponding text data in the Firebase Realtime Database under the Images node for the authenticated user.

- **Image Display**: The system allows users to view the generated images on the application interface.
  This function is responsible for retrieving the image data for the authenticated user from the Firebase Realtime Database and storing it in the history list. It first constructs the URL for the Images node of the authenticated user using the AuthService class. The function then sends a GET request to the Firebase Realtime Database to retrieve the image data. The response data is parsed into a Map object using the json.decode() method. The function then iterates over the Map object and extracts the text and image data for each image. The extracted data is then stored in an Item object and added to the history list.

- **Image Saving**: Users can save the generated images to their device's gallery.
  The function takes two parameters, the Uint8List representation of the image data and an optional quality parameter that specifies the quality of the saved image. The function returns a Future that resolves to a dynamic value.

Techniques and Algorithms Implemented:

- **Deep Learning**: The system utilizes deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to generate images based on the provided text input. These models are trained on a large dataset of fashion images to learn the mapping between textual descriptions and visual representations.
- **Conditional Generative Adversarial Networks** (cGANs) Diffusion Model: cGANs are employed to generate images conditioned on the input text descriptions. The generator network generates images based on the text input, while the discriminator network evaluates the quality and authenticity of the generated images.
- **Natural Language Processing** (NLP): NLP techniques are used to process and understand the text descriptions provided by users. This involves tasks such as text preprocessing, feature extraction, and semantic understanding.

Technical Specifications:

Preprocessing Techniques:

1) **Image preprocessing:**
   - **Remove background:**
     remove the background from the images and saves the output to improve the quality and accuracy of image generation.
     Frist iterate through each file in the directory and reads the images. It then splits the filename to extract the name and extension separately. Then remove background using rembg lib.then converts the output image to the RGBA color format. Finally, save the output image, with the filename formatted.

- **Face Segmentation:**
  using face segmentation, which can help improve the quality and accuracy of image generation using deep learning models and cGANs**.**
  first initializes a counter variable to keep track of the number of images processed. Then set the path to the folder containing the images. Then iterates through each image in the folder and reads the image using the cv2.imread() method. It then resizes the image to a maximum width of 400 pixels. Next, the code loads a pre-trained SSD model for face detection. It resizes the image to a maximum width of 375 pixels and generates a blob from the image. The SSD model is then used to detect faces in the image, and the code iterates through each detected face. For each face with a confidence greater than 0.8, the code computes the bounding box coordinates.

2) Text preprocessing:

- **Summarize using the spaCy:**
  Summarize the text prompt by getting the most important sentences in the description using the spaCy natural language processing library.
  first loads the spaCy English language model and tokenizes the input text into sentences and individual words. It then calculates the frequency of each word in the text, excluding stop words and punctuation, and normalizes the frequencies by dividing by the maximum frequency. Next, the function scores each sentence based on the frequency of the words it contains, and selects the top sentences based on the desired summary length. Finally, the function constructs the summary by concatenating the selected sentences into a single string.

- **Summarization using T5-transformer:**
  The function takes a data generated from summarition using the spaCy containing the summarized text descriptions and generates a new summary for each description using the T5-transformer model. The first load T5-transformer model and tokenizer from the Hugging Face model hub. The code then iterates through each row in the data and generates a new summary for each description. It first checks if the length of the description is greater than 65, which is the maximum length of the generated summary as the Keras model which used to generate image that will be usen in following steps does not accepts larger description . If the length of the description is greater than 65, the function tokenizes the input text using the T5 tokenizer and encodes it as input to the T5-transformer model.
  The model.generate() method generates a new summary based on the encoded input, using the T5-transformer model.

Deep learning model:

- Implementation of Trainer class for Text-to-Image Generation using Diffusion Models and VAEs in TensorFlow.
  The Trainer class inherits from tf.keras.Model, and is responsible for training the diffusion model and fine-tuning the VAE for text-to-image generation.
  The Trainer class takes several parameters, including the diffusion model, the VAE, the noise scheduler, and the maximum gradient norm. It also has an option for mixed precision training.
  The train_step method of the Trainer class implements the training loop for the diffusion model. It takes input images and encoded text as inputs and uses the VAE to project the images into the latent space and sample from it. It then adds noise to the latents according to the noise magnitude at each timestep, and predicts the noise residual using the diffusion model.
  The get_timestep_embedding method of the Trainer class generates a timestep embedding for the diffusion model, which is used to encode the timestep information into the model.
  The sample_from_encoder_outputs method of the Trainer class samples from the encoder outputs of the VAE to generate the latent representation for the images.
  The save_weights method of the Trainer class overrides the default method to only checkpoint the diffusion_model since that's what is being trained during fine-tuning.

- Implementation of Text-to-Image Fine-Tuning using Diffusion Models and VAEs in TensorFlow.
  defines an instance of the ImageEncoder and DiffusionModel classes, loads a pre-trained diffusion model, and initializes a Trainer instance for fine-tuning the model.
  The Trainer instance takes several parameters, including the diffusion model, the VAE, the noise scheduler, and a flag for mixed precision training. It also compiles the model with an Adam optimizer with specified hyperparameters and a mean squared error loss function.
  The USE_MP flag is used to enable mixed-precision training if the underlying GPU has tensor cores, which can help speed up the training process.
  The ImageEncoder class is responsible for encoding the input images into a latent representation, while the DiffusionModel class

is responsible for predicting the noise residual for text-to-image generation. The pre-trained diffusion model is loaded using the load_weights() method.

Defines a checkpoint path for saving the weights of the model during training, and initializes a ModelCheckpoint callback to save the model after each epoch.
The fit method of the Trainer instance is called with the training dataset and specified number of epochs, along with the ModelCheckpoint callback as a list of callbacks.
During training, the diffusion model is fine-tuned on the training dataset to improve its performance for text-to-image generation.
The ModelCheckpoint callback saves the model weights after each epoch, allowing the model to be restored and continued from where it left off in case of interruption or failure.

New Technologies used in Implementation:

- **TensorFlow**: TensorFlow, an open-source machine learning framework, is used for building and training the deep learning models used in text-to-image generation. It provides efficient computation and optimization capabilities for neural networks.
- **Google Colab Pro**: used for developing and training models. colab is a cloud-based platform that provides a Jupyter notebook interface to run and execute Python code. Colab provides GPUs, which can significantly speed up the training of machine learning models and users can collaborate on a notebook and share their work with others.
- **Flutter Framework**: The mobile application is developed using the Flutter framework, which allows for cross-platform development of high-quality mobile applications with a single codebase. Flutter provides a rich set of UI components and facilitates seamless integration with backend services.
- **Firebase**: Firebase is utilized for user authentication, real-time database storage, and cloud storage for saving and retrieving the generated images. It provides a scalable and secure backend infrastructure for the application.

Setup Configuration:

Code was trained and tested using Google Colab with python 3.9, Tensorflow, Keras, Torchvision, It was run on GPU: NVIDIA V100, System RAM: 83.5GB, GPU RAM: 40 GB, Disk Storage: 166.8 GB

UI Design and Wireframes:

The user interface (UI) of the application is designed to be **intuitive**, **user-friendly**, and **visually appealing**. The UI design includes the following key components:

- **Login and Registration Screens:** have been designed to enable users to access the application securely. Users can either log in using their existing credentials or create a new account by providing their details. The design of these screens has been kept simple and straightforward to facilitate ease of use.

- **Home Screen**: is the primary screen of the application, and it displays the main functionality of the application. The screen comprises a text input field where users can describe the clothing they want to generate images. The design of the home screen aims to ensure that users can easily navigate through the application and access the features they need.

- **Image Display Screen:** is where the generated images are displayed. The screen has been designed to enable users to view and interact with the images effectively. It includes options for saving the images, which are easily accessible to users.

- **History Screen**: is where users can access their previously generated images. The screen displays a collection of saved images and allows users to easily retrieve and view them.

- **Feedbacks Screen:** is designed specifically for administrator users. It enables them to access all the feedback entered by users. This screen has been designed to enable administrators to manage user feedback effectively.

Model Testing Procedures:

We used Fréchet Inception Distance which is a measure of similarity between two datasets of images. It was shown to correlate well with the human judgment of visual quality and is most often used to evaluate the quality of samples of Generative Adversarial Networks. FID is calculated by computing the Fréchet distance between two Gaussians fitted to feature representations of the Inception network.

These two datasets are essentially the dataset of real images and the dataset of fake images (generated images in our case). FID is usually calculated with two large datasets. However, we couldn’t use large dataset with the wedding dresses model.

In the casual wear model, we had a huge dataset, so we split the dataset and we choose unique prompts selected randomly from the entire dataset, while in the dresses model, we had to collect another 60 images of wedding dresses as it is a self-collected dataset and used them.

We took the prompts and started generating images in their respective models using the prompts from the images.

1) Casual Wear Model:
   - test Dataset 2k image and training dataset 21k.
   - Model FID: 72.333
2) Wedding Dresses Model:
   - test Dataset 60 image and training dataset 600.
   - Model FID: 94.666

Mobile APP Testing Procedures:

To ensure the quality and reliability of the system functions, various testing procedures and levels are used. Below are the testing procedures and levels that applied to the system functions:

1. **Unit Testing**: This level of testing involves testing individual functions or methods in isolation to ensure they work as expected.
   The functions responsible for user login with Google, text input, image generation, image storage, image display, and image saving all be tested individually to ensure they work as intended.

2. **Integration Testing**: This level of testing involves testing the interactions between different components of the system to ensure they work together seamlessly.
   The integration between the Flutter app and the API endpoint used for image generation can be tested to ensure that the data is being transmitted correctly and that the response is being parsed and displayed appropriately.

3. **System Testing**: This level of testing involves testing the system as a whole to ensure that it meets the requirements and specifications outlined for the application.
   The system testing for this system include testing the user login, text input, image generation, image storage, image display, and image saving functions in conjunction with each other to ensure that they work together as intended.

4. **Acceptance Testing**: This level of testing involves testing the system against the user requirements to ensure that it meets their needs.
   The acceptance testing for this system involves testing the user login with Google, text input, image generation, image storage, image display, and image saving functions to ensure that they meet the user's expectations.

5. **Regression Testing**: This level of testing involves retesting the system after changes have been made to ensure that the previously functioning features still work as intended.
   Regression testing is used to ensure that changes made to one function do not adversely affect other functions in the system

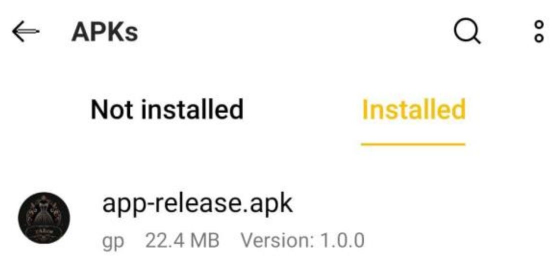
5- User Manual

The tAIlor is an AI Art Generator is a mobile application developed, which allows users to generate unique and creative fashion design with the help of Artificial Intelligence. This user manual will guide you through the process of using the application and generating your own AI-generated artwork.

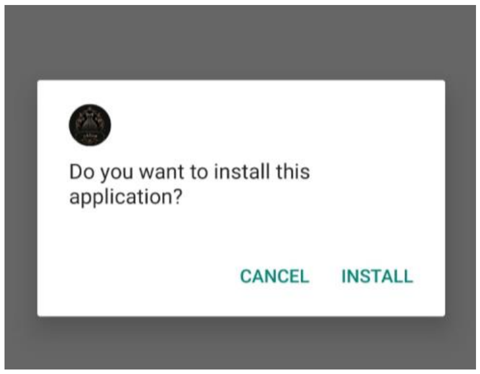
Installation Guide:

To install the AI Art Generator mobile application, follow the steps below:

1. Download the APK file from the provided link.
2. Once the download is complete, open the APK file on your mobile device.



3. Follow the on-screen instructions to install the application.

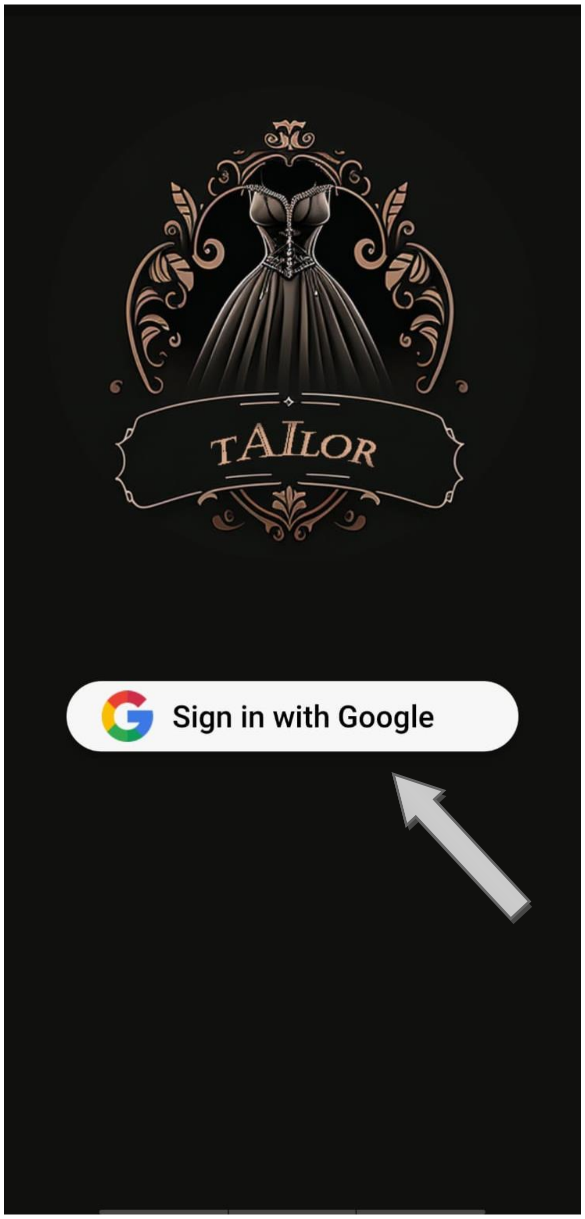

Do you want to install this application?

4. Once the installation is complete, open the application.

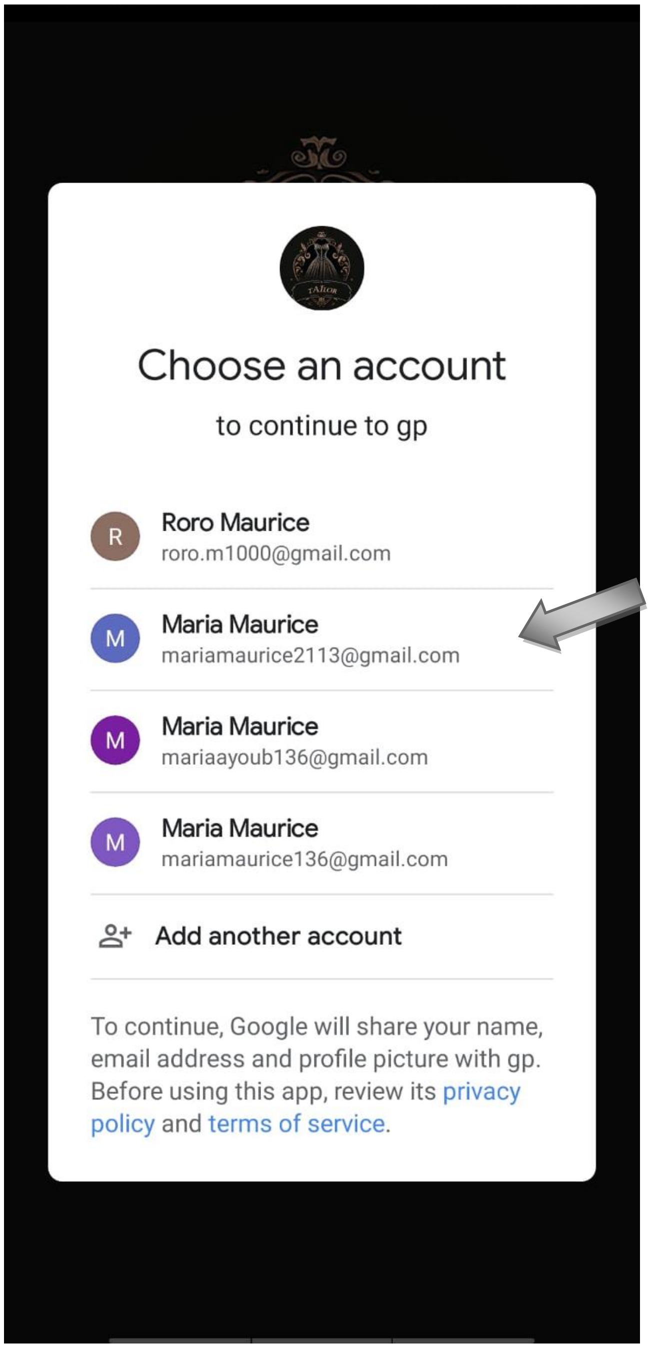
Note: You must have an active internet connection to use the application.

After successfully installing the application, follow the steps below to generate your own AI-generated fashion design:

Step 1: Sign Up with Google On the home screen of the AI Art Generator application, tap on the "Sign Up with Google" button. This will take you to a screen where you can sign in with your Google account.



Step 2: Grant Permissions
After tapping on the "Sign Up with Google" button, you will be prompted to grant permissions for the application to access your Google account. Follow the on-screen instructions to grant the necessary permissions.


Choose an account

to continue to gp

Roro Maurice
roro.m1000@gmail.com

Maria Maurice
mariamaurice2113@gmail.com

Maria Maurice
mariaayoub136@gmail.com

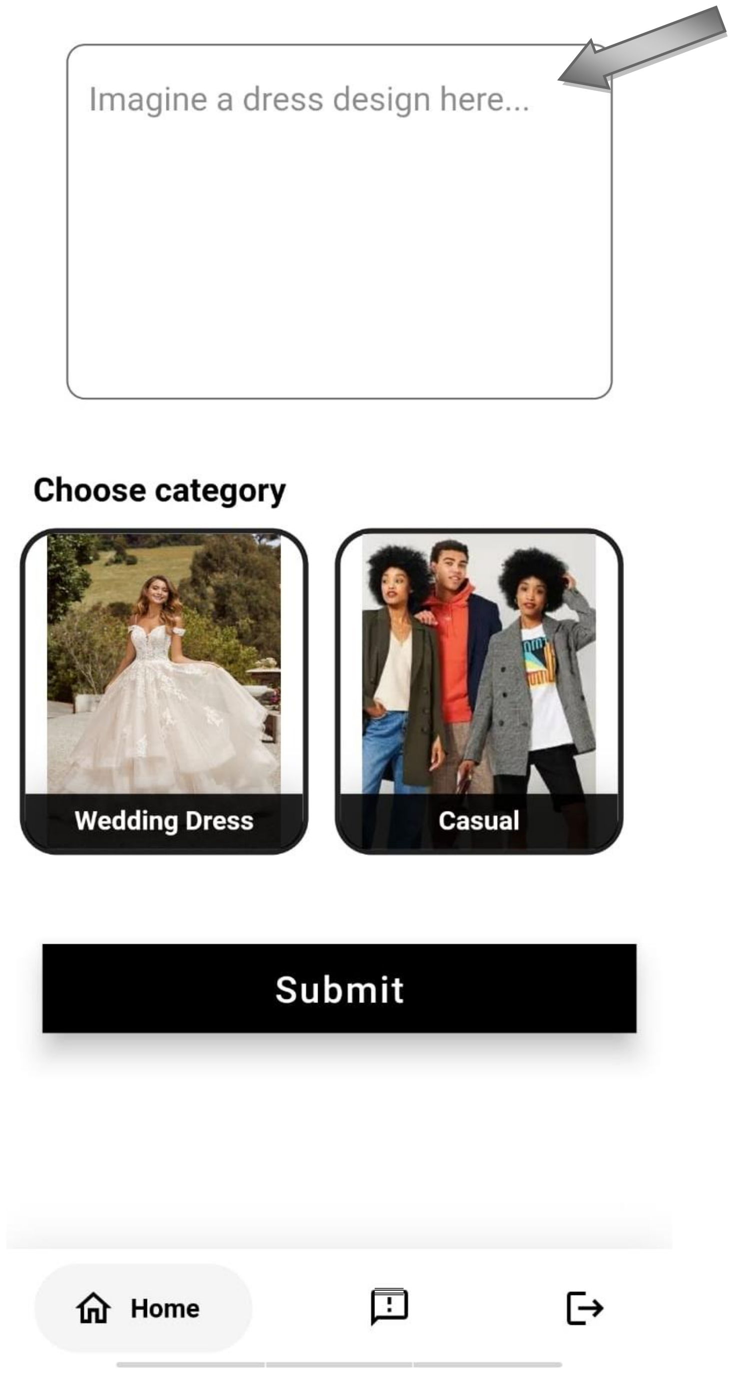
Maria Maurice
mariamaurice136@gmail.com

Add another account

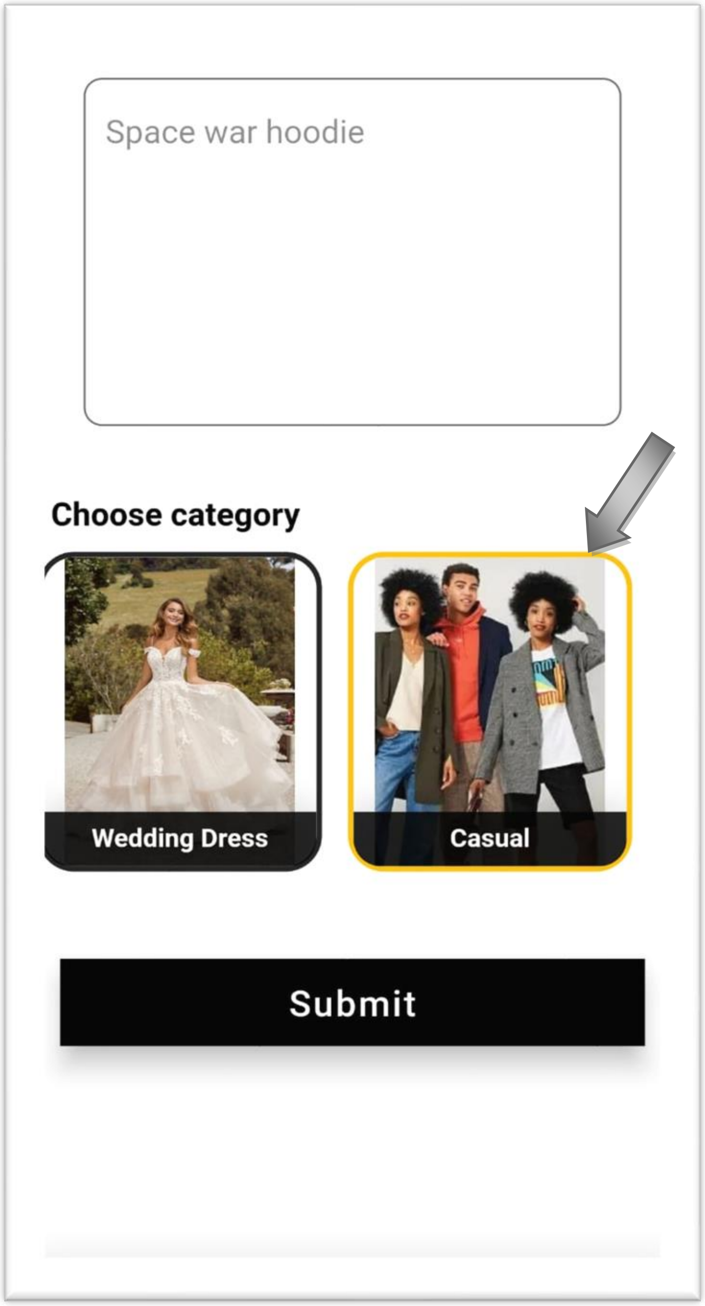
To continue, Google will share your name, email address and profile picture with gp. Before using this app, review its privacy policy and terms of service.

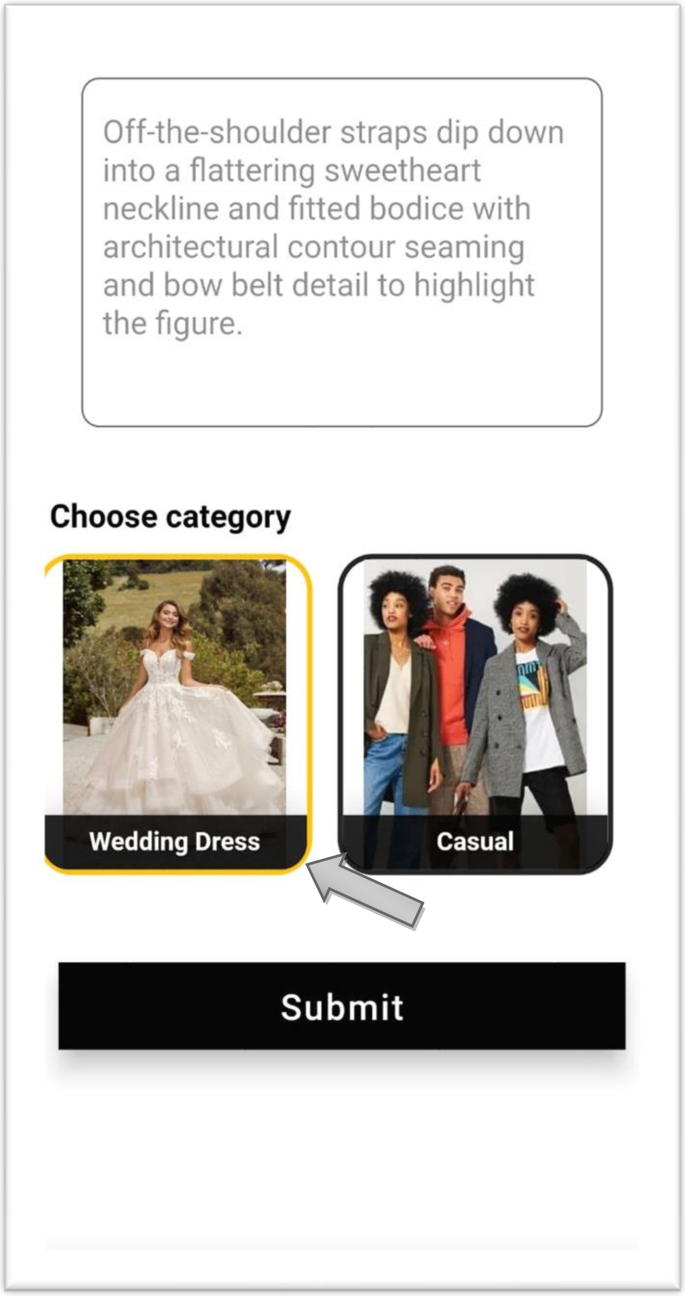
Step 3: Start Using the App

On the home screen of the application, you will see a text field where you can enter the text you want to generate the fashion design from.



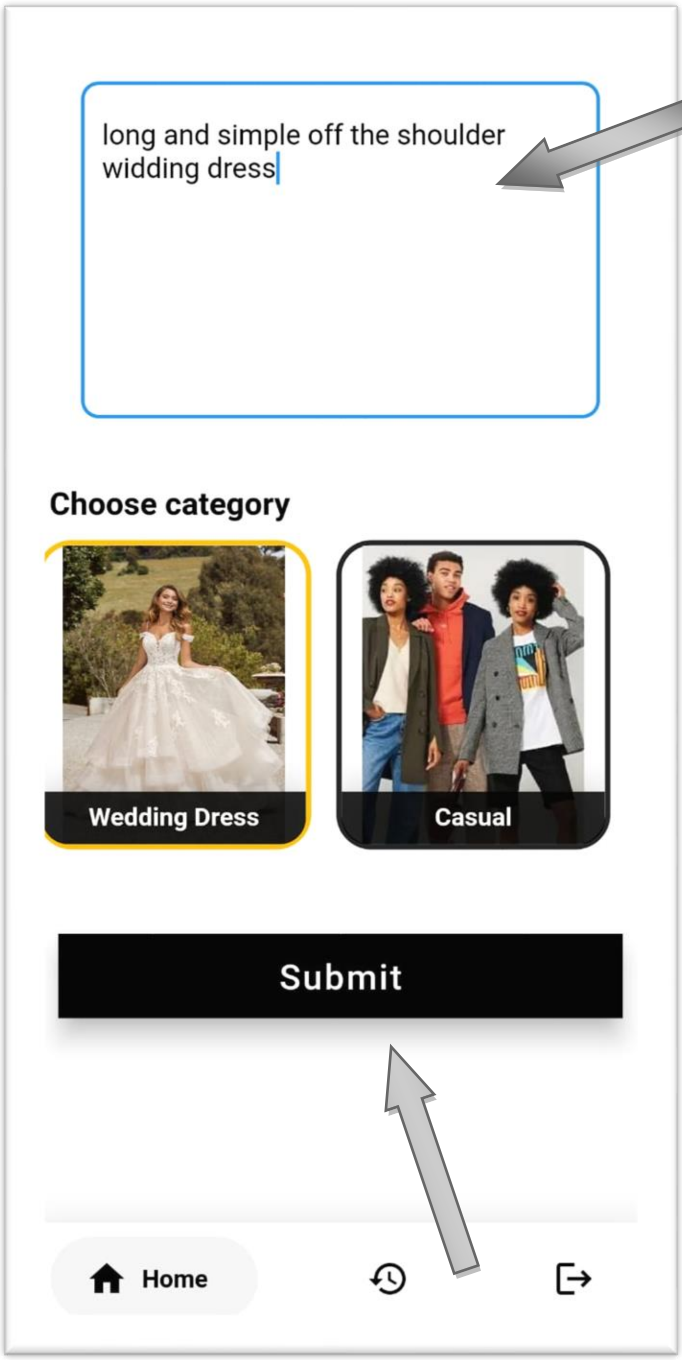
Choose a category for the fashion design from the provided list by tapping on the category field.
Once you Choose a category for the fashion design from the provided list you will see the hint of text field changed to guide you what you could type for best results

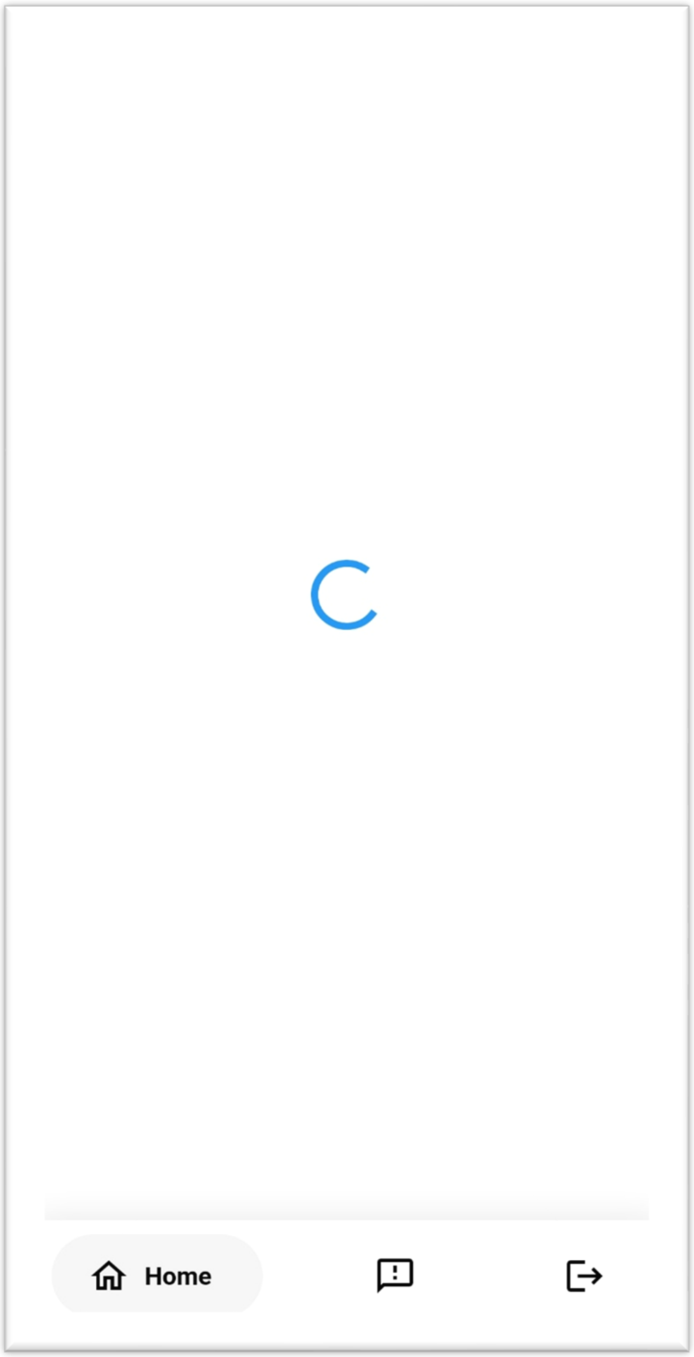

Space war hoodie


Off-the-shoulder straps dip down into a flattering sweetheart neckline and fitted bodice with architectural contour seaming and bow belt detail to highlight the figure.

Step 4: Submit and Wait for Results

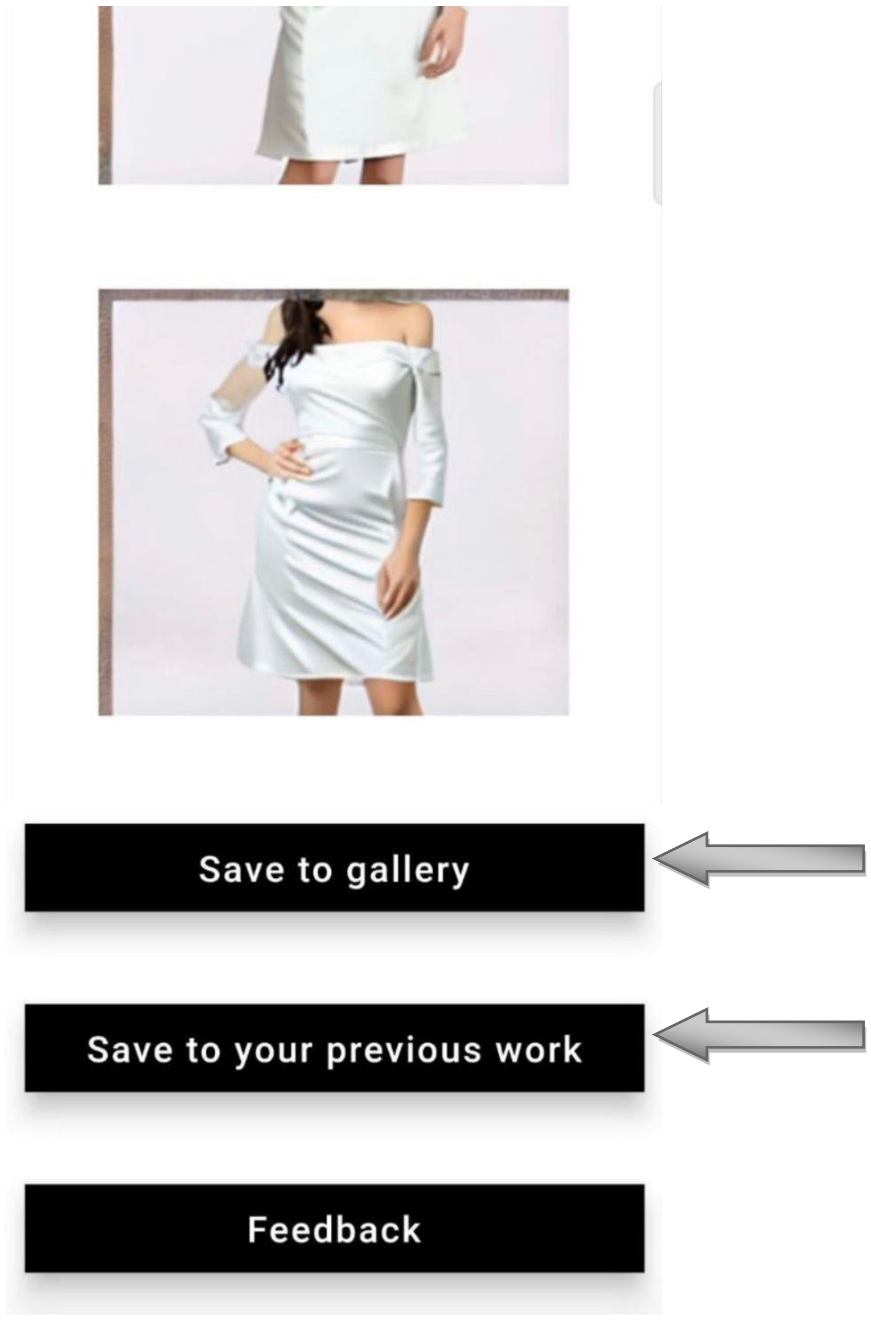
After entering the text and selecting a category, tap on the "Submit" button at the bottom of the screen. This will take you to a new screen where you will see a progress bar indicating that the AI algorithm is processing your request. Wait for a few moments and the results will appear.


long and simple off the shoulder widding dress

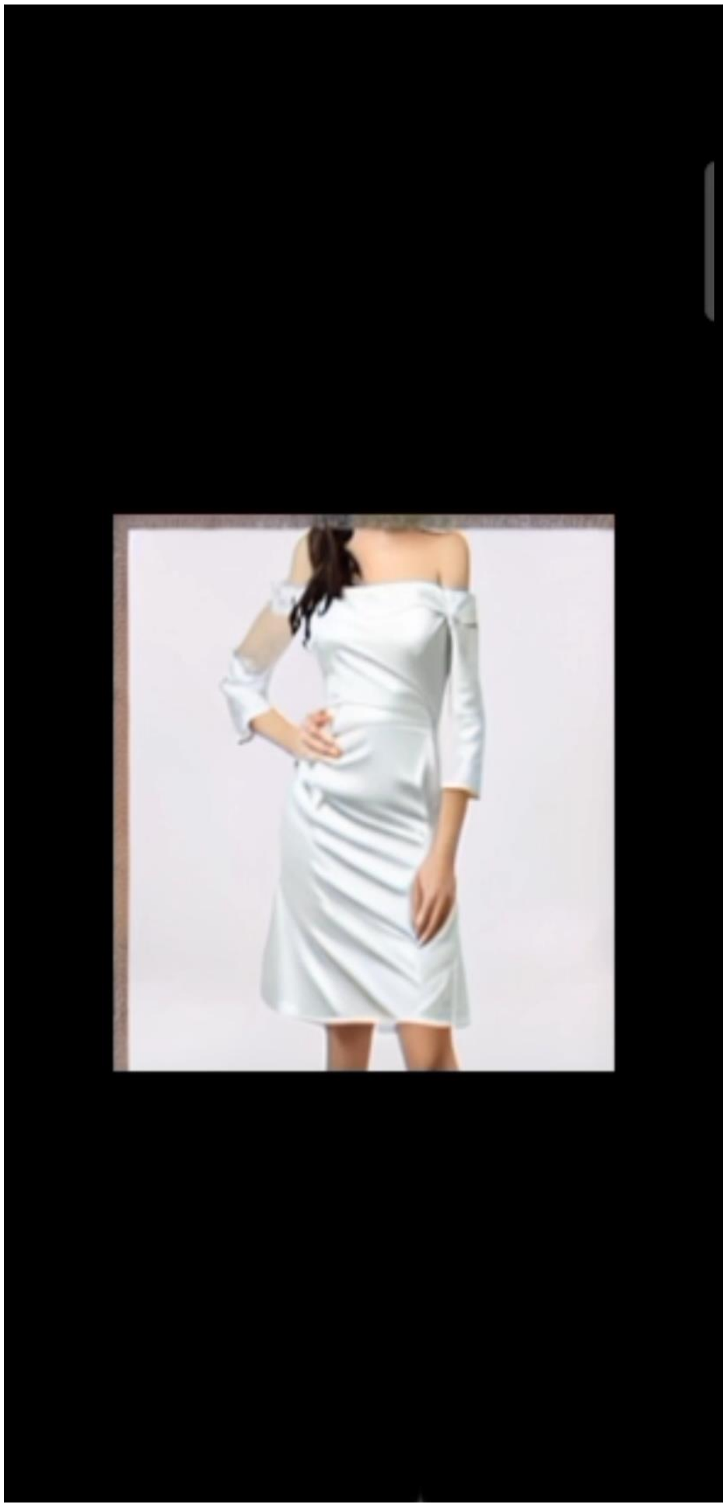


Step 5: View and Save Result

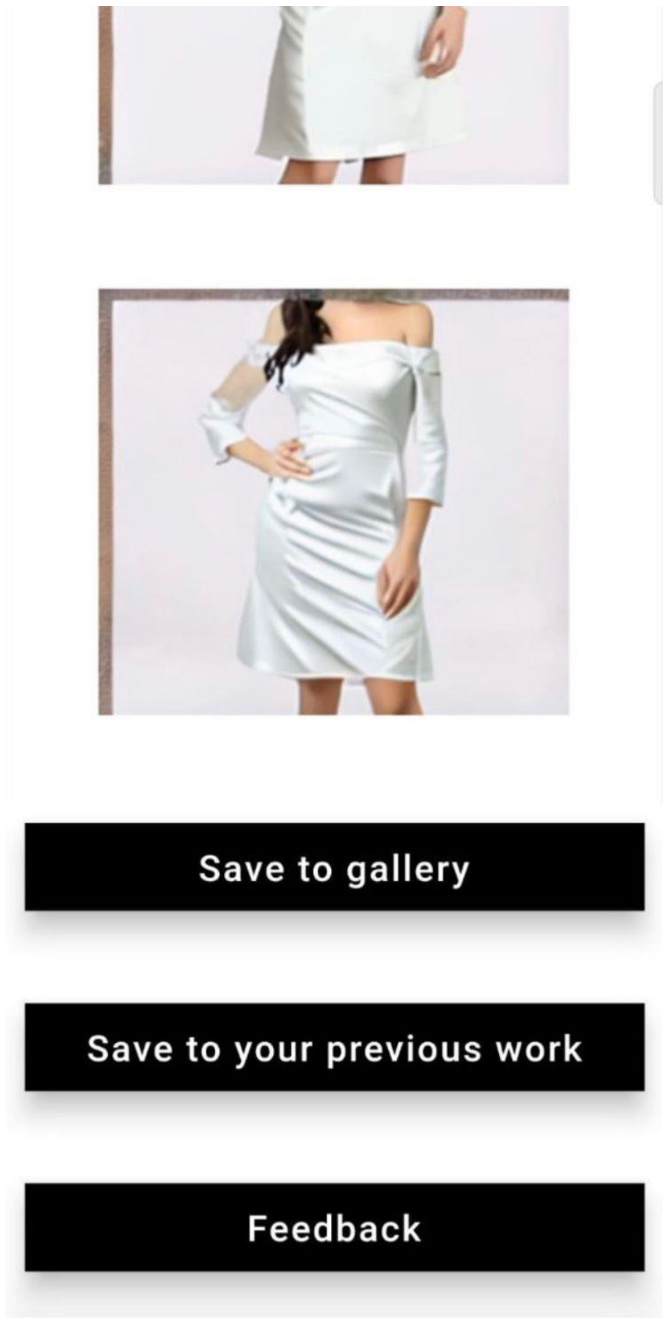
Once the progress is completed, you will see the generated fashion design image on the screen.



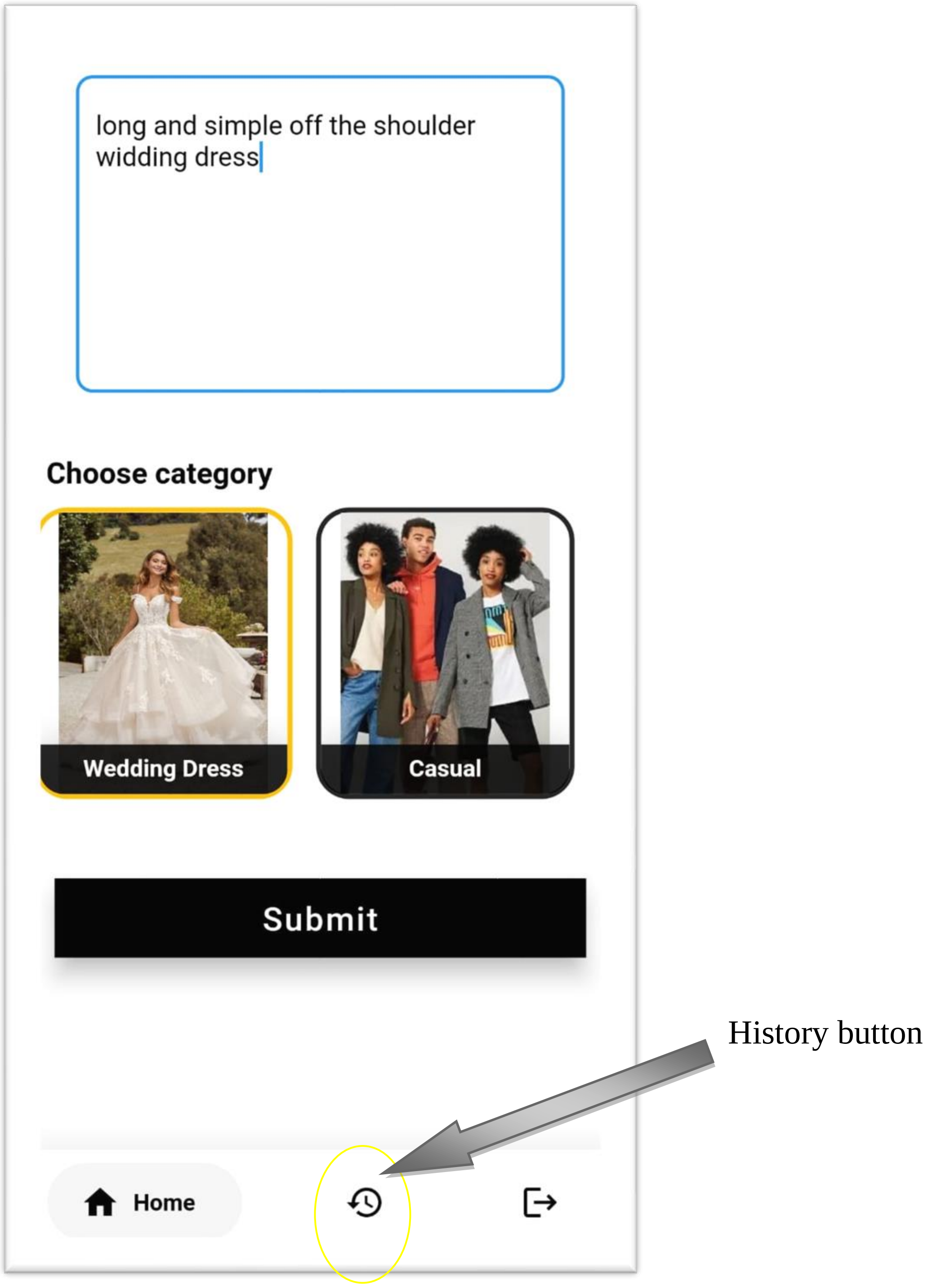
Tap on any of the generated images to view it on full screen.



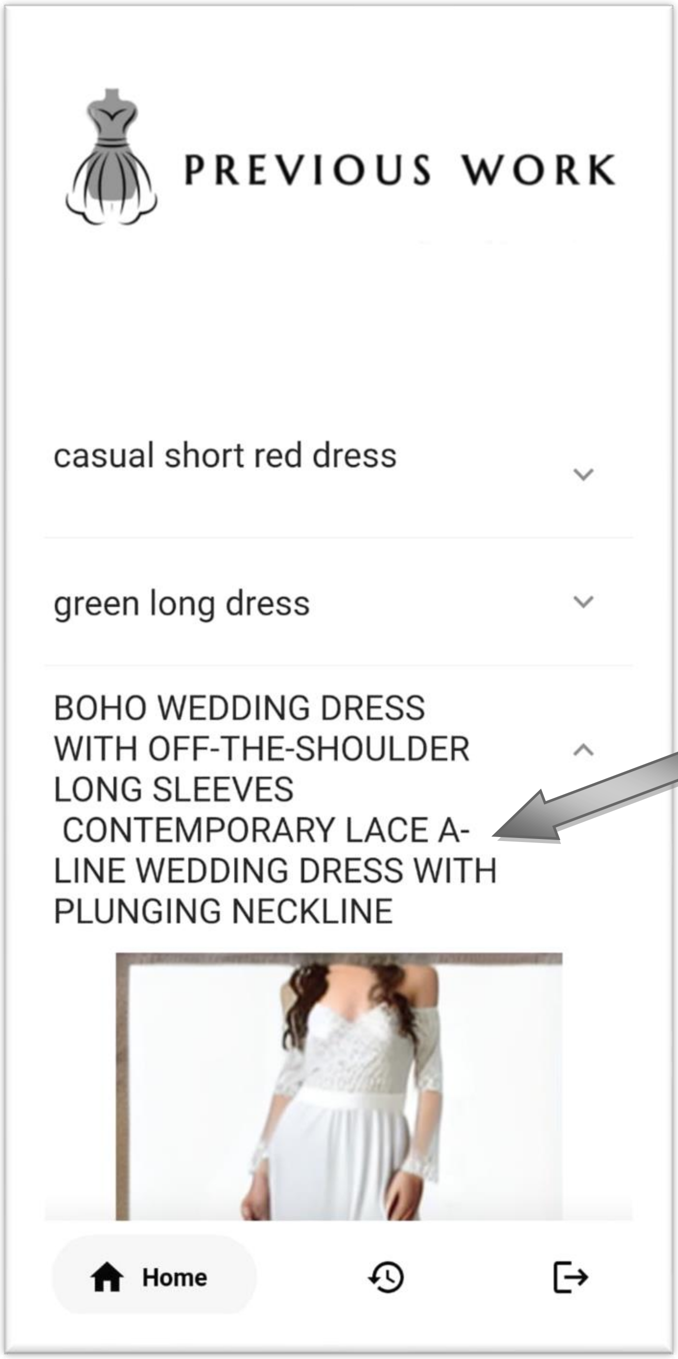
To save the fashion design image to gallery, tap on the Save To Gallery button at the bottom of the screen. You can also save the generated fashion design image to your history by tapping on the Save To previous work button. Press on the Feedback button to send your feedback to the generated images.



Step 6: Access Your History
To access your saved generated fashion design images, tap on the "History" button in the bottom navigation bar.


long and simple off the shoulder widding dress

This will take you to a screen where you can see all of your previously saved generated fashion design images with the text you entered to generate this image. Tap on any fashion design image to view it in full screen.

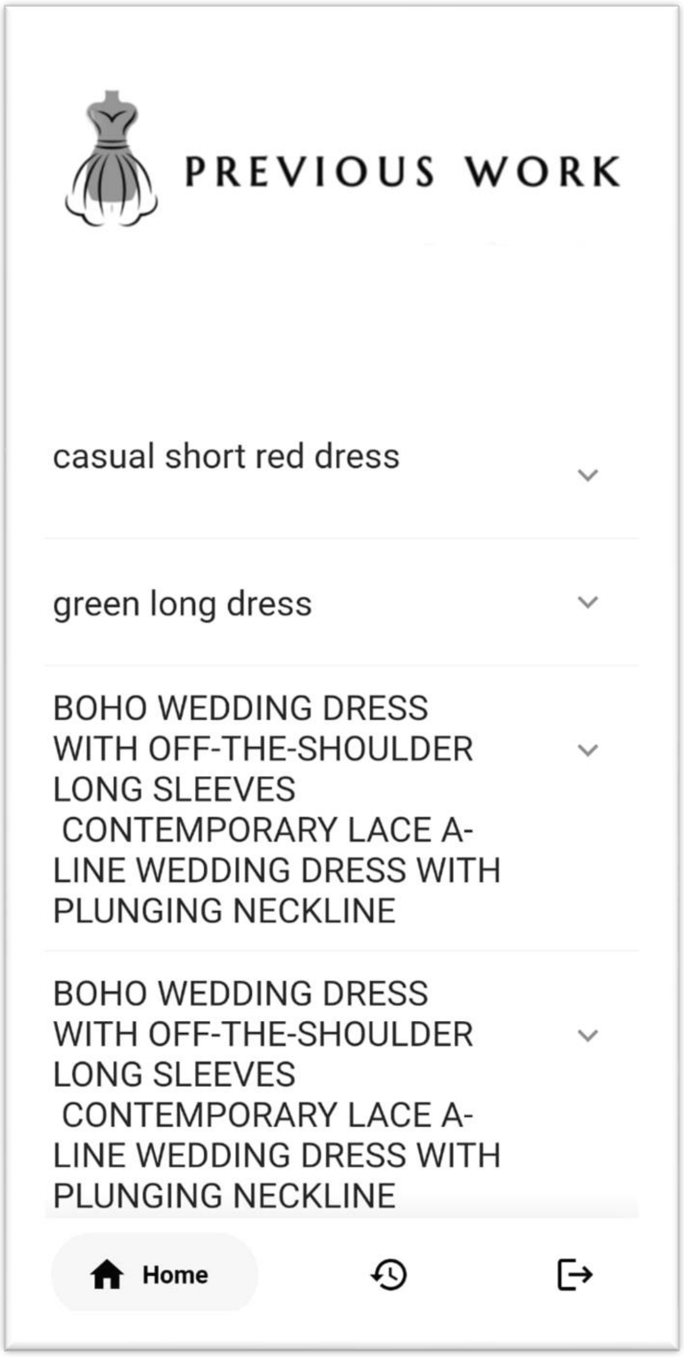


PREVIOUS WORK

casual short red dress

green long dress

BOHO WEDDING DRESS WITH OFF-THE-SHOULDER LONG SLEEVES
CONTEMPORARY LACE A-LINE WEDDING DRESS WITH PLUNGING NECKLINE



PREVIOUS WORK

casual short red dress

green long dress

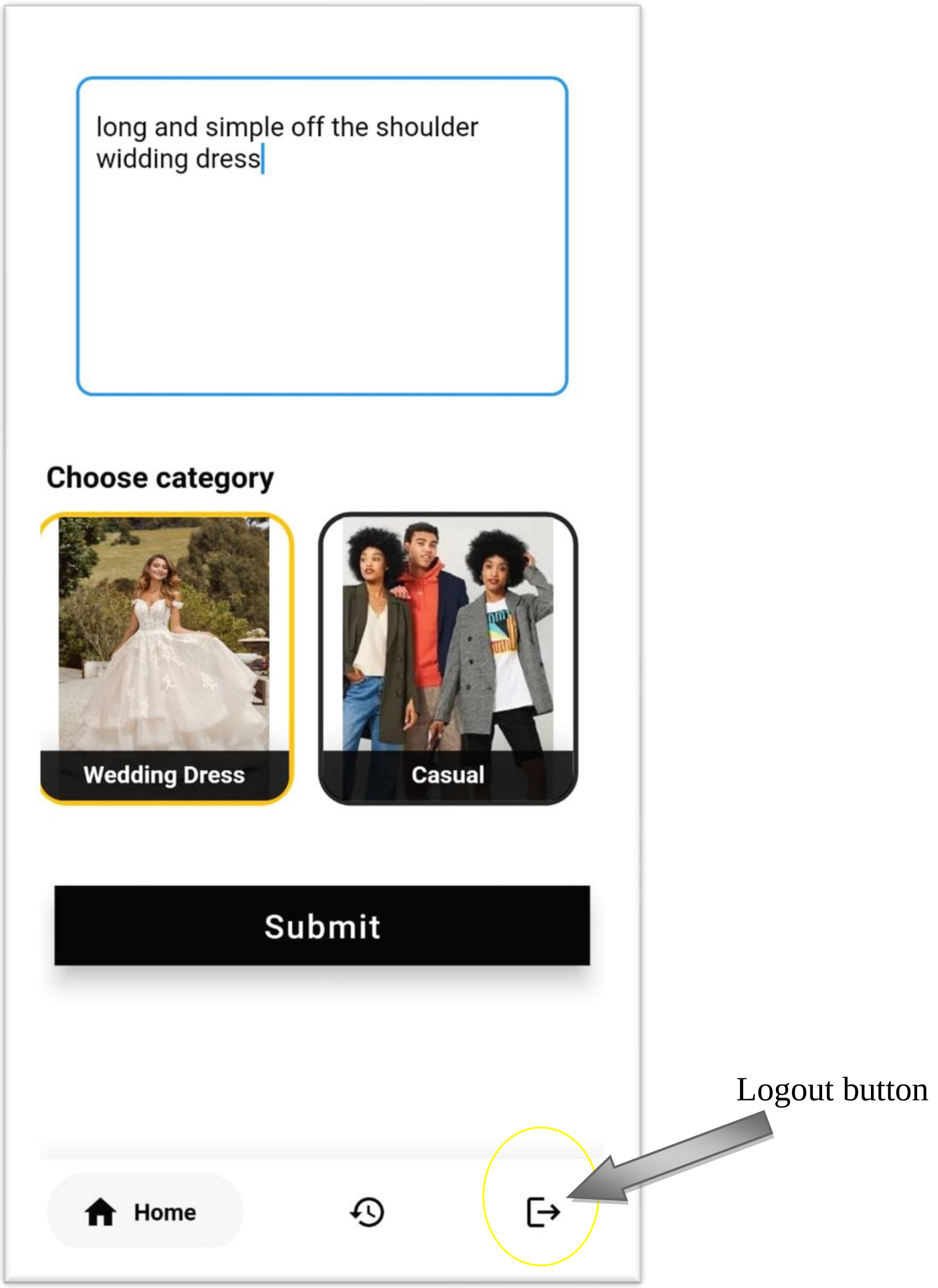
BOHO WEDDING DRESS WITH OFF-THE-SHOULDER LONG SLEEVES
CONTEMPORARY LACE A-LINE WEDDING DRESS WITH PLUNGING NECKLINE

BOHO WEDDING DRESS WITH OFF-THE-SHOULDER LONG SLEEVES
CONTEMPORARY LACE A-LINE WEDDING DRESS WITH PLUNGING NECKLINE

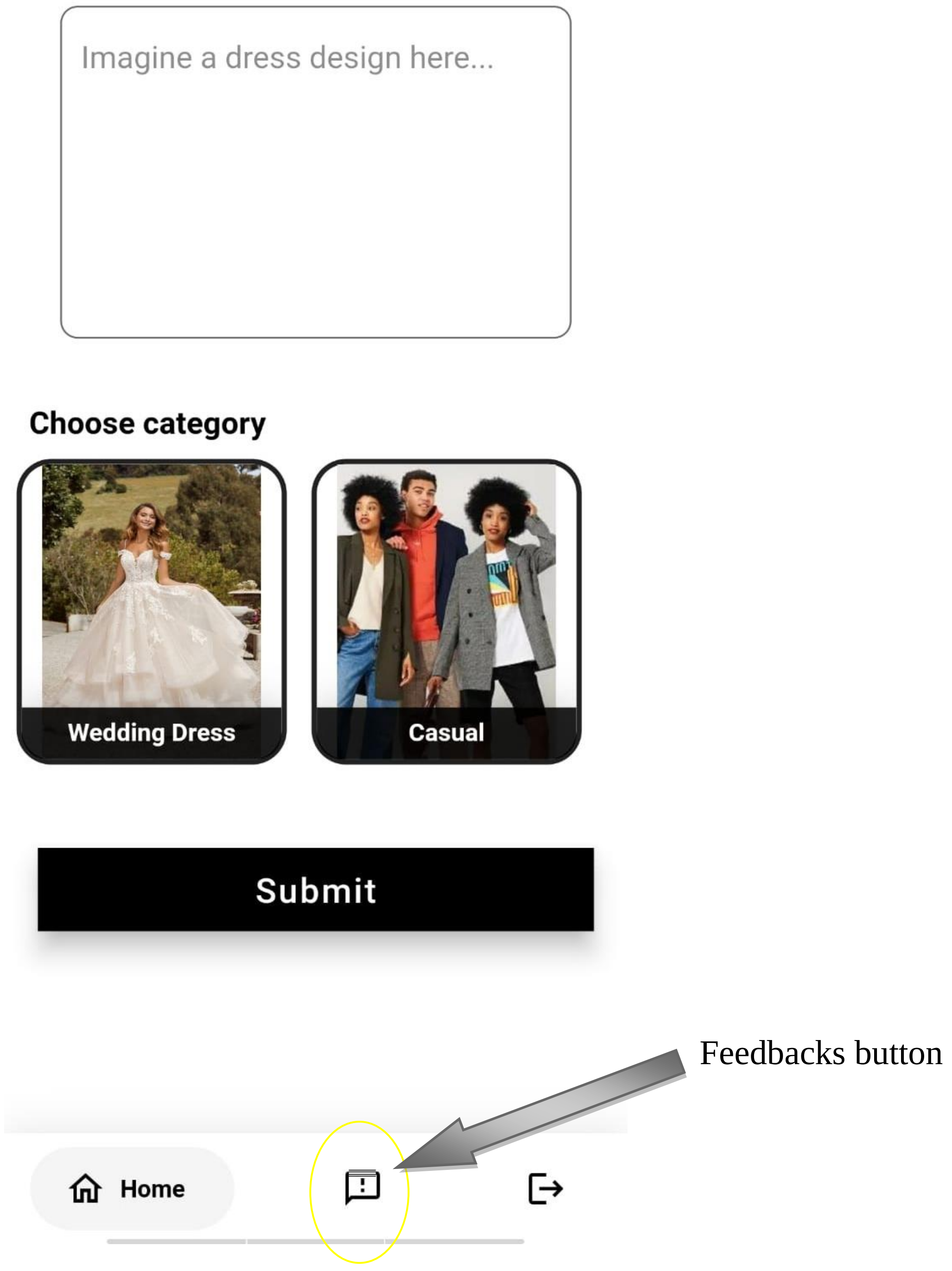
Step 7: Logout

To logout of the application, tap on the "Logout" button in the bottom navigation bar. This will log you out of your current account.

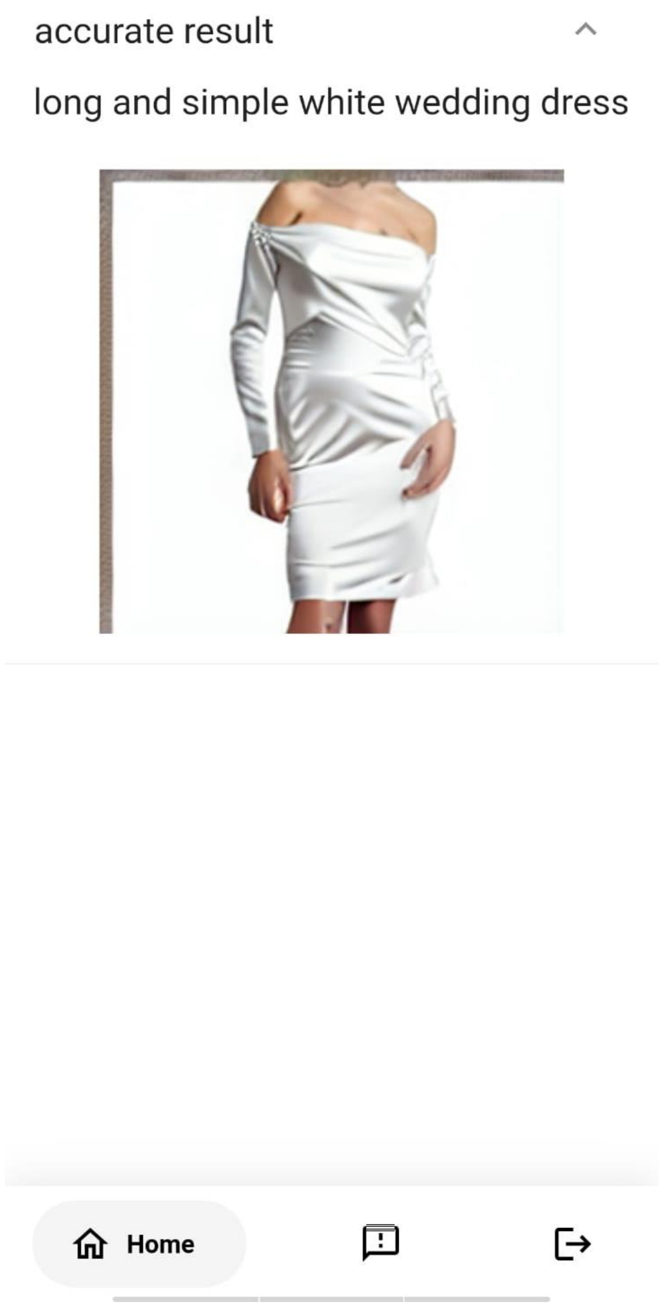
Note: your history depend on which account you sign-in with.


long and simple off the shoulder widding dress

For Administrators:
After signing in with your administrator account, you will be able to access the feedback feature in the bottom navigation bar of the application. To access this feature, simply navigate to the bottom of the screen and locate the "Feedbacks" button. Pressing this button will take you to a new page where you can view all feedback entries submitted by users of the application.



On this page, you will be able to view the text entered by users as well as the generated image associated with each feedback entry. This feature provides a valuable tool for administrators to gain insight into user experiences with the AI art generator and to identify opportunities for improvement.
Please note that this feature is only available to administrators who have signed in with their designated account.


accurate result

long and simple white wedding dress

6- Conclusion and Future Work

6.1 Conclusion

In this project, we have developed a virtual clothing customization system that allows users to generate realistic clothing images based on textual descriptions of desired attributes. Throughout the project, we have discussed the various components, methodologies, and techniques employed to design and implement the system. Now, let us provide a complete summary of the project, including the results obtained.

1- The primary objective of the project was to create a user-friendly and interactive platform where users could effortlessly customize clothing items by simply describing their desired attributes. To achieve this, we designed and implemented a system that combines natural language processing techniques with state-of-the-art image generation models.

2- We began by conducting a thorough analysis of user requirements and designing the system architecture accordingly. The system architecture consists of components such as the user interface, text processing module, image generation module, and database. These components work together seamlessly to provide a smooth and intuitive user experience.

3- The image generation module employs a diffusion model and algorithm, which leverages deep learning and probabilistic modeling to generate visually coherent and contextually relevant clothing images. By utilizing diffusion processes, the model captures both global structure and fine-grained details, resulting in high-quality image synthesis.

4- Throughout the implementation phase, we rigorously tested and optimized the system to ensure its reliability and efficiency. We also integrated the system with a database to store and manage user data and previously generated images.

Regarding the results obtained, the system has demonstrated impressive performance in generating clothing images that align with the provided textual descriptions. The diffusion model and algorithm have proven to be effective in capturing the desired clothing attributes and producing visually appealing results.

In conclusion, the virtual clothing customization system presented in this project showcases the successful integration of natural language processing and image generation techniques. It provides users with a seamless and interactive platform to personalize clothing items based on their preferences. The system's ability to generate realistic and contextually relevant images is a testament to the effectiveness of the diffusion model and algorithm employed.

Overall, this project has been a valuable exploration into the realm of virtual clothing customization and has yielded promising results. It serves as a foundation for future research and development in this exciting domain, with the goal of providing users with a rich and personalized virtual clothing experience.

6.2 Future Work

- The project opens possibilities for further enhancements and extensions, such as incorporating additional clothing categories, refining the user interface, and integrating more advanced image generation models. With continuous development and improvement, the system has the potential to revolutionize the way users interact with virtual clothing customization.

1) train both models on more images.
2) add more categories.
3) enhance the user experience.

References

- Blattmann, A., Rombach, R., Oktay, K. and Ommer, B., 2022. Retrieval-Augmented Diffusion Models. *arXiv preprint arXiv:2204.11824*.
- Feng, Z., Zhang, Z., Yu, X., Fang, Y., Li, L., Chen, X., Lu, Y., Liu, J., Yin, W., Feng, S. and Sun, Y., 2022. ERNIE-ViLG 2.0: Improving Text-to-Image Diffusion Model with Knowledge-Enhanced Mixture-of-Denoising-Experts. *arXiv preprint arXiv:2210.15257*.
- Li, R., Li, W., Yang, Y., Wei, H., Jiang, J. and Bai, Q., 2022. Swinv2-Imagen: Hierarchical Vision Transformer Diffusion Models for Text-to-Image Generation. *arXiv preprint arXiv:2210.09549*.
- Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Kreis, K., Aittala, M., Aila, T., Laine, S., Catanzaro, B. and Karras, T., 2022. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*.
- Yu, J., Xu, Y., Koh, J.Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B.K. and Hutchinson, B., 2022. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*.
- Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. and Chen, M., 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.
- Soloveitchik, M., Diskin, T., Morin, E. and Wiesel, A., 2021. Conditional frechet inception distance. arXiv preprint arXiv:2103.11521.
- Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M. and Aberman, K., 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 22500-22510).
- Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A.H., Chechik, G. and Cohen-Or, D., 2022. An image is worth one word: Personalizing text-to-image generation using textual inversion. arXiv preprint arXiv:2208.01618.