User: This class represents the user model in the application, utilizing SQLAlchemy to define a database schema with attributes like `username`, `email`, `password`, and `date_created`. It includes methods for getting the user's ID and a string representation for easy debugging.
LoginForm: This class defines the form for user login, including fields for username and password, with validation rules for required input and length restrictions.
SignupForm: This class outlines the signup form for new users, featuring fields for username, email, password, confirmation of password, and a checkbox to show the password. It includes validations to ensure proper input format.
ForgotPasswordForm: This form allows users to request a password reset by providing their username and new password. It enforces validation to ensure passwords match and meet length requirements.
VerifyOTPForm: This form captures the OTP (One-Time Password) entered by the user for email verification during the signup process, with validations to ensure the input is exactly six characters long.
generate_otp(): This utility function generates a random six-digit OTP used for user verification during the signup process.
load_user(user_id): This function loads a user from the database based on the user ID passed to it, facilitating user session management in the application.
forgot_password(): This route handles password reset requests by validating user input, updating the password in the database if the username exists, and providing user feedback through flash messages.
home_page(): This route serves the home page of the application, ensuring that only logged-in users can access it, redirecting unauthorized users to the login page.
signup(): This function processes user signup requests, sending an OTP to the user's email for verification and storing temporary user data in the session until the OTP is validated.
verify_otp(): This route verifies the OTP entered by the user against the OTP stored in the session, creating a new user in the database if the OTP is correct and providing appropriate feedback.
home(): This route handles user login, validating user credentials against the database and establishing a session if the login is successful.
logout(): This function logs the user out by clearing the user session and redirecting them to the login page.
login(): Similar to the `home()` function, this route processes user login requests and manages the session based on successful authentication.
about(): This route renders the 'about' page of the application.
contact(): This route renders the 'contact' page of the application.
favorites(): This route retrieves and displays the user's favorite items stored in the session.
add_to_favorites(): This function processes requests to add items to the user's favorites list in the session.
delete_from_favorites(): This route removes an item from the user's favorites list based on the provided link.
fashionrecommender(): This route renders the fashion recommender page for users to input their preferences.
fashionrecommender_cloth(): This function processes the input from the fashion recommender page, retrieves weather data based on user location, and makes a request to an image search API to find clothing options based on the user's preferences, ultimately rendering the results in a dedicated template.

try_on: Renders the main try-on page where users can select which type of accessory they want to try on.

try_on_cloth: Renders the page for trying on clothes, passing any uploaded image data to the template.

`try_on_glasses` : Similar to the above but specifically for trying on glasses.

`try_on_necklace`: Renders the necklace try-on page, again passing the uploaded image.

Glasses Processing :

  - **Input Handling**: It retrieves the uploaded image and glasses file.
  - **Image Decoding**: The base64 encoded image is decoded and converted to a format suitable for processing with OpenCV.
  - **Face and Eye Detection**: Using Haar cascades, it detects the face and eyes in the uploaded image.
  - **Glasses Placement**: The glasses image is resized to fit the detected eye dimensions and is overlaid onto the detected face region.
- **Necklace Processing** :
  - Similar to glasses, it retrieves the uploaded image and necklace file.
  - **Facial Landmark Detection**: Utilizes Dlib's facial landmark detector to find specific points on the face for accurate placement of the necklace.
  - **Resizing and Placement**: The necklace is resized based on the calculated dimensions and placed on the detected neck region.
- **Earrings Processing** :
  - Handles the upload of both left and right earring images.
  - **Face Detection**: Similar face detection is performed.
  - **Earring Resizing**: Earrings are resized based on the dimensions of the detected face and placed appropriately on each ear.
- **Hairstyle Processing** : Employs image segmentation along with SDEdit and the Fast Marching method to apply new hairstyles seamlessly to user images.
- **Clothes Processing** :Utilizes the ACGPN module for warping garments onto the user’s image, integrates object and pose detection for accurate fitting.