PROJECT REPORT

Virtual Eyewear Try-On System

---

1. INTRODUCTION

With the rise of e-commerce in the fashion and eyewear industry, customers often struggle to select suitable eyewear without physically trying it on. This project aims to solve that problem by building a Virtual Eyewear Try-On System using a webcam to capture the user's face and overlay virtual glasses in real-time or on captured images.

The system provides a user-friendly interface with login & signup options, allowing users to try different eyewear styles and view their results instantly

2. OBJECTIVES

The primary objectives of this project are:

1. To provide a virtual try-on experience for eyewear using a web-based platform.
2. To allow users to securely log in and sign up before accessing the try-on feature.
3. To capture the user's face via webcam and overlay different eyewear frames on the captured image.
4. To display multiple eyewear options in a stylish and user-friendly manner.
5. To enhance user engagement by providing a visually appealing and interactive interface.

3. TECHNOLOGIES USED

The following technologies and tools were used in this project:

1. Flask (Python Framework) – To handle backend logic, routing, and image processing.
2. HTML5, CSS3, and JavaScript – For creating a responsive and interactive user interface.
3. Webcam Integration (via JavaScript & getUserMedia API) – To capture real-time user images.
4. Werkzeug Security – For password hashing and authentication.
5. Bootstrap & Custom CSS Animations – For a modern, stylish, and animated user interface.

6. Python Imaging & Model Integration – To generate try-on images with eyewear overlays.
7. Local Storage (Temporary) – For storing user credentials (in-memory dictionary).
8. Jinja2 (Flask Templating Engine) – To dynamically render HTML pages with images.

4. SYSTEM ARCHITECTURE

The system architecture of the Virtual Eyewear Try-On project follows a client-server model:

1. Client Side (Frontend):
    - Built using HTML, CSS, and JavaScript.
    - Provides an interactive UI for user login/signup, image capturing, and try-on results.
    - Uses the Webcam API (getUserMedia) to capture real-time photos.
2. Server Side (Backend):
    - Implemented using Flask (Python).
    - Handles user authentication, image upload, and processing logic.
    - Integrates with the ML model (generate_tryon_images) to generate try-on results.
3. Temporary User Authentication:
    - Uses a Python dictionary to store usernames, emails, and hashed passwords.
4. File Handling:
    - Uploaded images are saved in the static/uploads folder.
    - Processed images (try-on results) are stored in the static/output folder.

5. IMPLEMENTATION DETAILS

The implementation of the project is divided into several modules:

5.1 User Authentication Module

- Signup:

    - Users can create an account by providing a username, email, and password.
    - Passwords are securely hashed using generate_password_hash() before storing.
- Login:
    - Users can log in by entering valid credentials.
    - The system verifies passwords using check_password_hash().
- Session Management:
    - Flask sessions are used to maintain login state.
    - Users cannot access try-on features without logging in.

5.2 Image Capture and Upload Module

- Uses the Webcam API (navigator.mediaDevices.getUserMedia) to access the webcam.
- Captured images are sent to the backend via POST request and stored securely.

5.3 Image Processing Module

- The generate_tryon_images() function takes the uploaded photo and applies virtual eyewear using the trained ML model.
- The processed images are saved in the static/output folder with unique filenames.

5.4 Result Display Module

- Users can view the generated try-on images in a gallery format.
- A "Try Again" button redirects them back to the capture page

6.RESULTS AND DISCUSSION

The Virtual Eyewear Try-On System was successfully implemented and tested. The following observations were made:

6.1 Functional Results

- **User Authentication**:
    - Sign-up and login worked smoothly with proper session handling.
    - Unauthorized users were restricted from accessing try-on features.

- **Virtual Try-On**:
    - Users could capture their face using the webcam and view realistic eyewear try-on results.
    - Different eyewear frames were displayed side by side for comparison.

6.2 User Interface

- The UI was designed with modern neon effects, smooth animations, and responsive layouts.
- The try-on gallery provided an interactive experience, allowing users to hover and enlarge eyewear previews.

6.3 Performance & Accuracy

- The system processed images quickly, with try-on images generated within seconds.
- Eyewear alignment on faces was reasonably accurate (depending on the quality of the input image).

6.4 Limitations

- The system works best with frontal face images; angled faces may cause slight misalignment.
- The project currently supports only three eyewear models