University of Nottingham
UK | CHINA | MALAYSIA

UNIVERSITY OF NOTTINGHAM NINGBO CHINA

SCHOOL OF COMPUTER SCIENCE

COMP2043.GRP Summary of quality assurance

[P2024-16] Online Virtual Dressing Room with Advanced Try-On and Clothing Retrieval Features

March, 2025

TEAM2024.07

SUPERVISED BY Qian Zhang

Zhihao Cao (scyzc10), Zixin Ding (ssyzd4), Peini She (scyps2), Zihan Zhou (scyzz15), Jinghao Liu (scyjl16)

20515084, 20513868, 20516641, 20515086, 20413235

Contents

1 Overview

The objective of the project is to design a comprehensive visual dressing room application online. The application allows users to try on clothes virtually with enhanced functionalities, such as clothing retrieval based on descriptions, efficient browsing and personalized recommendations. In addition, we have already set up a try-on clothing dataset, connecting every clothing item with their text description, to improve the accuracy of search and recommendation based on users.

2 Test Objectives

This document is designed to present the comprehensive functional testing evaluation of the virtual fitting room project, encompassing the assessment of the web application's intuitive user interface, corresponding textual descriptions, and the developed garment dataset. Functionality evaluations include the natural language retrieval system for garments and the garment recommendation system. Aligned with both functional and non-functional requirements, our objective is to ensure a seamless user experience that meets established quality standards.

3 Testing scope

The testing process comprehensively covers both functional and non-functional requirements of the online virtual dressing room. The primary goal was to ensure the correctness, usability, and robustness of all critical modules before delivery. Our test scope includes back-end logic, front-end user interface, database operations, image processing, and recommendation intelligence.

Functional Scope

- **Frontend React Components**: All major pages (Homepage, Login, Register, FullCloset, TryOn, Upload, ClothesDetail, History) were tested for UI rendering, user interaction, and navigation flows.
- **Backend APIs**: RESTful endpoints for clothing retrieval, history logging, try-on generation, and email delivery were validated using unit tests and manual calls.
- **Recommendation & Search Algorithms**: The similarity matrix and natural language search logic were tested for correctness, top-N ranking accuracy, and response consistency.
- **Image Upload and Try-On**: The system was tested for handling image input

(file/webcam), correct base64 conversion, server-side processing, and rendering of the result.

- **Email Dispatch**: The try-on image can be sent to a user-specified email address. Email structure and delivery success were verified.
- **Database Operations**: Initial schema creation, record insertion, and full backup-recovery procedures were tested using SQL scripts.

Non-Functional Scope

- **Performance**: We measured system response time for key actions such as image upload, try-on generation, and clothing retrieval. All actions completed within acceptable time (¡ 2 seconds).
- **Cross-device Compatibility**: The interface was tested on various screen sizes and resolutions to ensure responsive layout and consistent behavior.
- **Stability**: The system maintained consistent behavior across multiple sessions and concurrent requests.
- **Testing Environment**: We configured and validated the React testing environment with `jest-dom` and `App.test.js` to ensure testability and maintainability.

Some advanced features are reserved for future update and are not included in the current testing scope.

4 Test strategies

To ensure the quality of the software before delivery, we mainly use two kinds of test strategy: alpha testing and In-house testing.

4.1 Alpha-testing

Tables 1 to 4 present the alpha testing outcomes for the system's key components: UI, back-end, front-end, and database. Test cases assessed functionalities such as account operations, image capture, recommendation, API behavior, data handling, and virtual try-on rendering. A total of 57 test cases passed, demonstrating a high level of functional correctness.

UI tests confirmed user operations and feedback; back-end tests validated API logic and data flow; front-end tests ensured component interaction and visual accuracy; database tests confirmed schema integrity and data recovery. The results reflect strong alignment with system requirements and indicate readiness for broader beta testing under real-user scenarios.

ID	Scenario	Test Step	Test Data	Expected Result	Actual Result	P/F
TC001	User Create	Enter email and password	Click create button	Redirect to homepage	Success, token issued	Pass
TC002	User login	Enter email and password	Valid infos	Redirect to homepage	Success, token issued	Pass
TC003	Invalid login	Enter wrong password	Valid email + wrong pw	Error message shown	Error displayed	Pass
TC004	Image capture	PC Camera capture	Camera capture image	capture successfully	capture successfully	Pass
TC005	Try-on preview	Upload user image and apply clothing	user.jpg, cloth_id=45	Image aligned and rendered	Misaligned at neckline	Pass
TC006	Image search	Input query and click search	“red dress”	Matching items returned within 1s	Correct results shown	Pass
TC007	Popular recommend	Access /recommend/popular	None	Return top 5 items	Correct items returned	Pass
TC008	Upload image	Upload jpg under 2MB	white_shirt.jpg	Saved and preview shown	Upload successful	Pass
TC009	Empty search	Submit empty input	-	Prompt user to enter text	Alert shown correctly	Pass
TC010	View clothes	Click ”View all clothes” button	Click operation	Show all clothes	clothes shown correctly	Pass
TC011	Return function	Click ”Back to the dressing room” button	Click operation	Return to dressing room	Return correctly	Pass
TC012	Recommend function personalized	Click ”Explore recommendation” button	Click operation	Show 5 clothes of recommendation	Show correctly	Pass
TC013	Add to closet	Click ”Add” on clothing item	cloth_id=102	Item saved in closet	Item visible in UI	Pass
TC014	View history	Click ”View my history”	single click operation	correct cloth-Id	correct cloth-Id	Pass

Table 1: Functional Test Cases of UI

ID	Scenario	Test Step	Test Data	Expected Result	Actual Result	P/F
TC001	Send try-on image via email	POST to send email and image format	correct email and image format	Email sent successfully with image attachment	As expected	Pass
TC002	Validate registration form input	Submit short username/password to form	username=ab, password=123	Validation error raised	As expected	Pass
TC003	Add and retrieve browsing history	POST to /add-history and GET /get-history for same user	user_id=1, clothing_id=10	History added and returned successfully	As expected	Pass
TC004	Process virtual try-on image	POST to /process_image with user_id and cloth_url	userId = 1, cloth-url is special path	Image processed and result path returned	As expected	Pass
TC005	Search algotithm	return items of clothes	change topn = 4	show 4 recommend clothes	correct cloth items	Pass
TC006	Image Captioning	Run the generate caption file	Image set of 300 clothes	Caption JSON generated for each image	All JSON stored successfully	Pass
TC007	Caption Insertion	Run the insert cloth data file	JSON + image files	Data inserted into Clothing table	DB contains all entries	Pass
TC008	Search Algorithm	Call `/recommend/topn`	topn=4	Return 4 similar clothing items	Correct items returned	Pass
TC009	Similarity Matrix	Run file of precomputing matrix similarity	900 images(tops, bottoms, dresses)	Save .npy matrix files	File shape: (900, 900)	Pass
TC010	Caption Model Switching	Switch to DeepSeek in the deepseek generate caption file	-	Captions generated using DeepSeek model	JSON format consistent with spec	Pass
TC011	Get Clothing API	Call `/api/clothes/` from `database_api.py`	GET request	All clothing returned as JSON	Data in correct format	Pass
TC012	Single Clothing Query	Call `/api/clothes/23`	Valid cid=23	Return cloth detail info	Matching item returned	Pass
TC013	Auth Decorator	Access API with login_required	g.user = None	Redirect to login page	Redirect correctly	Pass
TC014	Serve clothing image	Call serve clothing image() fuction with valid path	000001_top.jpg	Returns image content	Image served correctly	Pass
TC015	Generate clothing title	Call generate_title() with valid caption dict	caption JSON (top)	Returns readable title string	Title matches expected pattern	Pass

Table 2: Functional Test Cases of Back-end

ID	Scenario	Test Step	Test Data	Expected Result	Actual Result	P/F
TC001	App Routing	Load base route /	Access Homepage	Homepage component rendered	Welcome message shown	Pass
TC002	Login Functionality	Submit valid credentials	Username + Password	Redirect to Try-On Page	Successful login popup	Pass
TC003	Register Functionality	Submit matching passwords	New account data	Redirect to login page	Success message shown	Pass
TC004	Clothing Detail View	Navigate to /detail/:id	Valid clothing ID	Item info and image shown	Details rendered correctly	Pass
TC005	Add to Closet	Click add in detail page	user_id + clothing_id	Add item to closet	Alert shown + success	Pass
TC006	Full Closet Display	Open full closet with user ID	user_id=1	Show all user items	Images loaded by category	Pass
TC007	Search in Closet	Enter text and press Enter	Query = "red dress"	Show matching items only	Filtered results shown	Pass
TC008	History Tracking	Visit item detail	user_id, item_id	Save to browsing history	Record in backend + render	Pass
TC009	Popular Sidebar	Visit full closet	No input	Show top 5 popular items	Items displayed in sidebar	Pass
TC010	Performance Reporting	App startup with reportWebVitals	On load event	Console logs performance metrics	CLS, LCP etc. logged	Pass
TC011	Image Upload (Base64)	Upload image via file or webcam	Local .jpg or webcam stream	Preview appears + image stored	Image displayed correctly	Pass
TC012	Virtual Try-On	Click "Generate" with uploaded image	Uploaded base64 + cloth_url	Server returns try-on result	Rendered result shown in UI	Pass
TC013	Send Try-On to Email	Enter email and click send	example-test.com	Image sent to mailbox	Success message displayed	Pass
TC014	Jest Setup Validation	Use testing-library-jest-dom in App.test.js	Load test.js	Matchers available	No error in test environment	Pass

Table 3: Functional Test Cases of Front-end

ID	Scenario	Test Step	Test Data	Expected Result	Actual Result	P/F
TC001	Database Initialization	Run OVDR.sql	SQL schema file	All tables created successfully	Tables created with correct schema	Pass
TC002	Database Backup Recovery	Import backup.sql	Dumped SQL data	All previous data restored	Clothing, user data intact	Pass

Table 4: Functional Test Cases of Database

All alpha test demonstrate functional correctness across UI, backend, frontend, and database layers, ensuring system stability, accurate interactions, and reliable data flow in controlled environments.

4.2 In-house testing

In-house testing phase was conducted after internal alpha testing, with the goal of validating the system usability. Additionally, this phase focused on accessibility and performance under realistic usage conditions.

Testing Objectives

The primary objective of In-house testing was to identify user experience issues that were not discovered during alpha testing. It aimed to evaluate whether the application met user expectations in terms of interaction flow, response time, and overall satisfaction.

Participants

We invited a group of 10 non-technical users, including university students and casual users, to participate in In-house testing. These users had no prior involvement with the development process.

Testing Process

Each participant was asked to perform a set of normal operations within the online virtual dressing room, including:

- Registering and logging into the system
- Uploading a personal image
- Searching for clothing using descriptive keywords
- Trying on clothing virtually
- Viewing and interacting with recommended items

Throughout the testing process, users operated the software entirely on their own, without any guidance or intervention from the developer, and then summary the users experience.

User Feedback Summary

The majority of users found the interface intuitive and aesthetically pleasing. The search functionality was reported as accurate in most cases. However, the following issues were observed:

- Slight delays in virtual try-on rendering on slower networks
- Occasional image misalignment in head replacement
- Lack of filter options such as brand or size

- Difficulty locating the "closet" feature without prior instructions

User satisfaction distribution in In-house testing:

As shown in Figure 1, 70% of users reported being either satisfied or very satisfied with the system. The feedback confirms the application's usability and readiness for public release. Identified issues and suggestions will be addressed in future iterations to further enhance user experience.

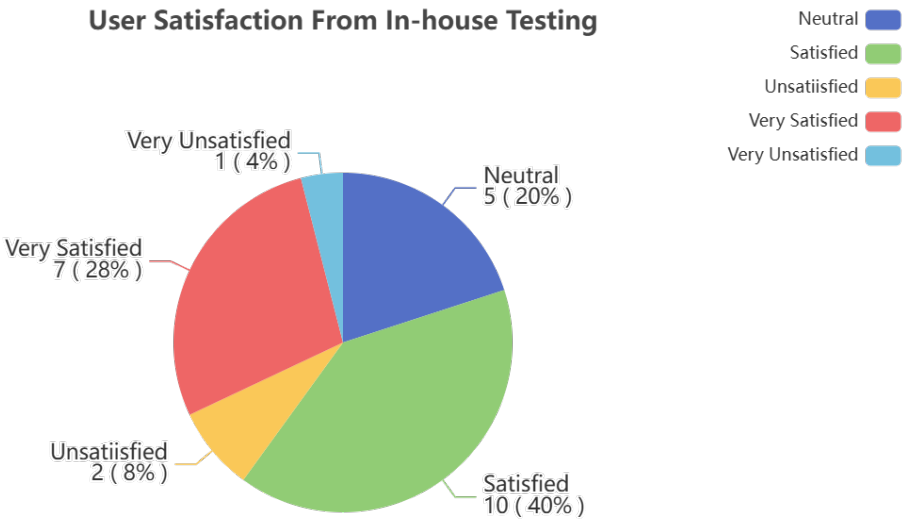

Figure 1: User satisfaction distribution based on In-house testing feedback

- Very Satisfied – 30%
- Satisfied – 40%
- Neutral – 20%
- Unsatisfied – 7%
- Very Unsatisfied – 3%

5 Future improvement

Future improvements will focus on enhancing user personalization and optimize algorithm to reduce the time costs of image generation. Additional improvements in image alignment accuracy and try-on realism are also scheduled. Furthermore, the feedback of users gathered during In-house testing will be prioritized in future development cycles.

6 Summary

This report outlines the complete quality assurance process of the online virtual dressing room application. Through a large number of alpha and In-house tests, validating the functionality, usability, and stability of the application. The testing covered major modules, ensuring the application meets expected quality standards and is ready for future delivery.