

System Verification and Validation Plan for HairEsthetics

Team 18
Charlotte Cheng
Marlon Liu
Senni Tan
Qiushi Xu
Hongwei Niu
Bill Song

April 3, 2023

1 Revision History

Date	Version	Notes
Oct 25	1.0	Initial draft
Apr 3	2.0	Revision 1

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	iv
3	General Information	1
3.1	Summary	1
3.2	Objectives	1
3.3	Relevant Documentation	1
4	Plan	1
4.1	Verification and Validation Team	1
4.2	SRS Verification Plan	2
4.3	Design Verification Plan	2
4.4	Verification and Validation Plan Verification Plan	3
4.5	Implementation Verification Plan	3
4.6	Automated Testing and Verification Tools	3
4.7	Software Validation Plan	4
5	System Test Description	4
5.1	Tests for Functional Requirements	4
5.1.1	Hair Style System	4
5.1.2	Hair Color System	6
5.1.3	Hair Salon Recommendation System	7
5.1.4	Authentication System (Removed from SRS)	8
5.2	Tests for Nonfunctional Requirements	10
5.2.1	Look and Feel Requirements	10
5.2.2	Usability and Humanity Requirements	11
5.2.3	Performance Requirements	14
5.2.4	Operational and Environmental Requirements	20
5.2.5	Maintainability and Support Requirements	21
5.2.6	Security Requirements	22
5.2.7	Cultural and Political Requirements	22
5.2.8	Legal Requirements	23
6	Unit Test Plan	23
6.1	Unit Testing Scope	23

7	Traceability Matrix	24
8	Appendix	30
8.1	Symbolic Parameters	30
8.2	Usability Research Plan	30
8.3	Usability Survey Questions	30
8.4	Reflection	31
8.4.1	Senni	31
8.4.2	Bill	31
8.4.3	Jack	32
8.4.4	Charlotte	32
8.4.5	Marlon	33
8.4.6	Hongwei	33

List of Tables

1	The verification tools used in each module/section	3
2	Traceability Matrix Showing the Connections Between Functional Requirements and functional requirements tests	25
3	Traceability Matrix Showing the Connections Between Functional Requirements and functional requirements tests	25
4	Traceability Matrix Showing the Connections Between Functional Requirements and functional requirements tests	26
5	Traceability Matrix Showing the Connections Between non-functional Requirements and non-functional requirements tests	27
6	Traceability Matrix Showing the Connections Between non-functional Requirements and non-functional requirements tests	28
7	Traceability Matrix Showing the Connections Between non-functional Requirements and non-functional requirements tests	29

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
SRS	Software Requirements Specification
DP	Development Plan
HA	Hazards Analysis
MG	Module Guide
GUI	Graphical User Interface
PoC	Proof of Concept

This document will provide the testing plan for HairEsthetics, including verification and validation Plan of SRS, design, and implementation, as well as system tests ranging from functional requirements tests to unit tests. It will also act as an outline of testing methods and outline the tools that will be utilized.

3 General Information

3.1 Summary

The software application HairEsthetics will be tested. Its functionality includes 3D hair style and color simulation, hair salon recommendation and authentication System.

3.2 Objectives

The objective for testing this project is to build confidence in the software correctness and demonstrate safety of data, efficient execution, and adequate usability.

3.3 Relevant Documentation

Relevant documentation includes *SRS*, *HA*.

1. SRS - </docs/SRS/SRS.pdf>
2. Hazard Analysis - </docs/HazardAnalysis/HazardAnalysis.pdf>

4 Plan

This section will provide the verification and validation plan of SRS, design, and implementation as well as tools that will be used for Verification and Automated Testing.

4.1 Verification and Validation Team

Everyone on the team shares a role in the verification and validation process.

- Bill Song and Marlon Liu - Responsible for backend API testing and system integration testing
- Charlotte Cheng and Senni Tan - Responsible for frontend interface test
- Qiushi Xu and Hongwei Niu - Responsible for AR related testing.

4.2 SRS Verification Plan

Our SRS would be verified through getting ad hoc feedback from our classmates and TAs. Since the team that reviewed The plan and TA had knowledge on our SRS, our plan is to ask them to explore our application with improvisation and ensure that they cover every possible functionality covered in our SRS. They can use their own ways to try to explore the extent of each features provided in our application and find possible behavioral bugs.

4.3 Design Verification Plan

Design will be verified by teammates, classmates and stakeholders. A checklist will be created to ensure the success of design in terms of the ability to meet the requirements and potential to satisfy stretch goals. Design is verified if the actual design output is the same as expected design output which satisfies the specifications of the product. The following checklist is composed of questions with regard to the design of the project.

- Did you design the product right?
- Did you design the right product?
- Can users accomplish their goals and tasks with the application?
- Is the design effective?
- Does the design provide good utility?
- Does the design's actual output match the expected output?

4.4 Verification and Validation Plan Verification Plan

The verification and validation plan will be verified by a group of our classmates and TAs where they will be asked to perform a walk-through of the document to ensure that our test plan is complete and effective. Any feedback will be accepted and changes will be made to address those reviews.

4.5 Implementation Verification Plan

The functionality and performance of our code implementation will be mostly verified through our system testing. It will ensure our application performs all the specified tasks as expected. Additionally, a part of the NFRs are also validated by the system tests. Our unit test are designed to take care of each individual module by assessing their quality using stubs or drivers. And it also evaluates the details of each module by providing test case for each function.

Our implementation would also require static verification. The method is decided to have classmates and TA perform code inspection to our source code. This can help reveal any code styling issue and potential logical errors in our implementation.

4.6 Automated Testing and Verification Tools

The following tools will be used for automated testing:

Module	Automated Testing and Verification Tools
Backend API	Postman
Python code	Pytest, Pycoverage, flake8
JavaScript code	Jest, Mocha

Table 1: The verification tools used in each module/section

4.7 Software Validation Plan

No external data will be used for validation, therefore no plans are formulated for it.

Review sessions will be conducted with the stakeholders to check that the requirements document fully capture the right requirements. For this purpose, a task-based walkthrough will serve as the validation. Users of the system were already identified in SRS along with a set of real-world tasks that they complete using the system. These tasks must mirror what users actually do in their environment but not how they do it. These tasks must be very specific and they must be complete. This information was gathered by observing the actual target users using the application in their environment, team 18 came up with a list of target users and realistic tasks. The combination of a user and their corresponding task descriptions is called a scenario and the goal is to tell a complete story. With a manageable list of scenarios in place from SRS, a task-based walkthrough of the system can be performed by teammates and stakeholders. A scenario will be selected and participants will place themselves in the mind of their target user and stay on track with what that user is trying to accomplish during the scenario.

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Hair Style System

1. HS-T1

Type: Functional, Dynamic, Manual

Initial State: Hair Style Page, with granted webcam permission

Input: Live video footage from webcam

Output: The chosen hair model is placed on top of the tester's head

How the test will be performed: Open the hair style page on desktop or mobile phone. The tester will click on the hair style page, and select different hair styles. And check if the hair models are rendered correctly by rotating their head.

2. HS-T2

Type: Functional, Dynamic, Manual

Initial State: Hair Style Page

Input: User click deny webcam permission

Output: The live video will be disabled and a error message will be displayed.

How the test will be performed: Open the hair style page on desktop or mobile phone. The tester will click on the hair style page, and select 'deny' when the browser prompts the permission dialogue. After clicking 'deny', the test will not see themselves on the page, and the video container will be shown in black.

3. HS-T3

Type: Functional, Dynamic, Manual

Initial State: Home page

Input: Click hair style on the menu

Output: Switch to the hair style page

How test will be performed: When at the home screen, the tester clicks on the button that directs to virtual simulation, see if it switches to the virtual simulation screen

4. HS-T4

Type: Functional, Dynamic, Manual

Initial State: Hair Style Screen

Input: A hairstyle is selected in the color selection bar at the bottom of the screen

Output: The application fits the hairstyle onto the user's head, at an appropriate position and scale.

How test will be performed: The tester should try to select a hairstyle, and check if the system fits the hairstyle onto the user's head at an appropriate position and scale.

5.1.2 Hair Color System

1. HC-T1

Type: Functional, Dynamic, Manual

Initial State: Home page

Input: Select hair color on the menu

Output: Switch to hair color page

How test will be performed: The tester will select the hair color tab and checks if the hair color page is rendered correctly

2. HC-T2

Type: Functional, Dynamic, Manual

Initial State: Hair Color Page

Input: User click deny webcam permission

Output: The live video will be disabled and an error message will be shown.

How the test will be performed: Open the hair style page on desktop or mobile phone. The tester will click on the hair style page, and select 'deny' when the browser prompts the permission dialogue. After clicking 'deny', the test will not see themselves on the page, and the video container will be shown in black.

3. HC-T3

Type: Functional, Dynamic, Manual

Initial State: Hair Color Screen

Input: An image is uploaded and select a color from the palette

Output: The page will display the transformed image with the correct color

How test will be performed: The tester should upload an image and select a color, then check if the system outputs the transformed image with the selected color

4. HC-T4

Type: Functional, Dynamic, Manual

Initial State: Hair Color Screen

Input: Click upload image button

Output: The page will open the select image dialogue and allows the user to upload images from local storage.

How test will be performed: The tester will click the upload image button and check if they can upload an image in the specific format

5.1.3 Hair Salon Recommendation System

1. HSR-T1

Type: Functional, Dynamic, Manual

Initial State: Home screen

Input: Click operation on recommendation tab

Output: Switch to the recommendation page with a map

How test will be performed: The tester clicks on the recommendation tab and see if the screen changes to the recommendation page with a map

2. HSR-T2

Type: Functional, Dynamic, Manual

Initial State: Recommendation page

Input: The user entered a valid address in the search box

Output: The map zooms to the entered location and displays up to 5 location markers (recommended hair salon) near the entered location

How test will be performed: Enter a valid address and see if the map zooms to that location correctly and markers being displayed around the location (within 10 kilometer radius)

3. HSR-T3

Type: Functional, Dynamic, Manual

Initial State: Recommendation page

Input: User entered an invalid or unknown address

Output: The page displays an address not found message

How test will be performed: Enter an invalid address and check if the error message shows up.

4. HSR-T4
Type: Functional, Dynamic, Manual
Initial State: Recommendation page
Input: The user clicked on the "use my current location" button
Output: The map zooms to the user's current location and displays up to 5 location markers (recommended hair salon) near the location (within a 10-kilometer radius)
How test will be performed: Click on the 'use my current location' button and see if the map zooms to the current location correctly and markers being displayed around the current location.
5. HSR-T5
Type: Functional, Dynamic, Manual
Initial State: Recommendation page with markers
Input: User clicked on one of the marker
Output: The details of that hair salon will be shown. Details include address, postal code, overall ratings, and link to their web page if possible.
How test will be performed: Click on one of the marker and see if the details of the hair salon will be displayed.
6. HSR-T6
Type: Functional, Dynamic, Manual
Initial State: Hair salon details tab opened
Input: User clicked on the website link
Output: The app will open user's default browser and redirect to the clicked link (hair salon website)
How test will be performed: Click on the link and see if a browser is being opened redirected to the selected hair salon website

5.1.4 ~~Authentication System (Removed from SRS)~~

1. ~~AS-T1~~
~~Type: Functional, Dynamic, Manual~~
~~Initial State: Home screen~~

~~Input: Correct email and password and click register~~

~~Output: Successful registration~~

~~How test will be performed: Create a test user using proper email format (xxx@xxx) and password. Click register and check if successful registration~~

2. AS-T2

~~Type: Functional, Dynamic, Manual~~

~~Initial State: Home screen~~

~~Input: Incorrect email and password and click register~~

~~Output: Failed registration with incorrect email format message~~

~~How test will be performed: Create a test user using improper email format (without @ sign) and password. Click register and check if the registration failed.~~

3. AS-T3

~~Type: Functional, Dynamic, Manual~~

~~Initial State: Home screen~~

~~Input: Email and password~~

~~Output: Successful login~~

~~How test will be performed: Create a test user. Enter correct credentials and check if the internal state of the system is logged in.~~

4. AS-T4

~~Type: Functional, Dynamic, Manual~~

~~Initial State: Home screen~~

~~Input: Incorrect email and password~~

~~Output: Failed login~~

~~How test will be performed: Create a test user. Enter incorrect credentials and check if the internal state of the system is not logged in.~~

5. AS-T5

~~Type: Functional, Dynamic, Manual~~

~~Initial State: User personal page (logged in)~~

~~Input: Open gallery~~

~~Output: Display user's saved images~~

~~How test will be performed: Create a test user with sample images.~~

~~Then login as the test user and see if the sample images show up in the gallery.~~

5.2 Tests for Nonfunctional Requirements

5.2.1 Look and Feel Requirements

Appearance Requirements

1. AP-T1

Type: Structural, Dynamic, Manual

Initial State: Application installed onto a cell phone and running

Input/Condition: App opened

Output/Result: Interfaces of the application

How test will be performed: Tests will be done manually by developers running the prototype. Tests will be enhanced by actual user feedback via a survey asking them "Is the text on the application outstanding and easily recognizable for you with this font size? Rate from 1-5(bad to good)" The appearance should achieve an average mark above 4.

2. AP-T2

Type: Structural, Dynamic, Manual

Initial State: Application installed onto a cell phone and running

Input: App opened

Output: Logo, icon, buttons, and other functional entity units that the user can interact with

How test will be performed: Tests will be done manually by developers running the prototype. Tests will be enhanced by actual user feedback via a survey asking them "Is the elements on the interface well-organized enough for you to use the application easily? Rate from 1-5(bad to good)" The average mark should be greater than 4.

Style Requirements

1. IA-T1

Type: Structural, Dynamic, Manual.

Initial State: ~~Application installed onto a computer~~ **Application running on a computer website** ~~computer and running~~

Input/Condition: ~~App~~ **website** opened

Output/Result: Any visual elements in the interface

How test will be performed: The test will be done manually by developers running the prototype. Tests will be enhanced by actual user feedback via a survey asking them "In what scale, would you say the style of the interface matches the modern aesthetic? (1-5) (poorly - very)" The average mark should be greater than 4

2. IA-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~ **Application running on a computer website**

Input: ~~App~~ **website** opened

Output: The entire interface presented to the user

How test will be performed: The test will be done manually by developers running the prototype. Tests will be enhanced by actual user feedback via a survey asking them "In what scale, do you think the interface is simple and clear? (1 - 5) from very disagree to very agree" The average mark should be greater than 4

5.2.2 Usability and Humanity Requirements

Ease of Use Requirements

1. EU-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: The entire interface and any interactable elements on the interface

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "In what scale, do you think the interactable components in the interface is easy to be recognized? (1 - 5) from very disagree to very agree" The average mark should be greater than 4

2. EU-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input: User inputs

Output: The entire interface and any interactable elements on the interface

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "During the use of this application, how many times did you misclicking an interactable element?" All the responded times that the user misclick an interactable element should be less than 5

Learning Requirements

1. LR-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application is not yet installed~~ Application is not running on the website

Input/Condition: User operations

Output/Result: ~~installed~~ **run** application

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "Did you finish the installation within 3 steps: open App Store, search for "Hairesthetics", click on install?" All the response should be "yes"

2. LR-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~ **Application running on a computer website**

Input: User inputs

Output: Instructions on the interface when user first open the application

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "In what scale do you find the instruction easy to follow? (1 - 5) from very disagree to very agree" The average mark should be greater than 4

Understandability and Politeness Requirements

1. UP-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~ **Application running on a computer website**

Input/Condition: User inputs

Output/Result: Icons on the interface

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "In what scale do you find the icon is appropriate and with no offense? (1 - 5) from very disagree to very agree" The average mark should be greater than 4

Accessibility Requirements

1. AS-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Text on the interface

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "In what scale do you find the text has appropriate font size and color? (1 - 5) from very disagree to very agree" The average mark should be greater than 4

5.2.3 Performance Requirements

Speed and Latency Requirements

1. SL-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Hairstyle suggestion is given on the screen

How test will be performed: Test will be done manually by developers running prototype. The developer will test the hairstyle suggestion functionality 100 times and measure the time from giving input to the hairstyle suggestion output. Number of times when the response time is greater than 1 second will be recorded. The recorded number of times should be less than or equal to 10

2. SL-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Hairstyle changing on the screen

How test will be performed: Test will be done manually by developers running prototype. The developer will test the hairstyle changing functionality 100 time and measure the time from giving input to the hairstyle changing output. Number of times when the response time is greater than 1 second will be recorded. The recorded number of times should be less than or equal to 10

3. SL-T3

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Responses to the user inputs

How test will be performed: Test will be done manually by developers running prototype. The developer will randomly clicking any interactable buttons 100 time and measure the time from clicking the button to the response output. Number of times when the response time is greater than 0.5 second will be recorded. The recorded number of times should be less than or equal to 10

Precision and Accuracy Requirements

1. PA-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Hairstyle on the user's head on the screen

How test will be performed: Test will be done manually by developers running prototype. The developer will test the hairstyle functionality 100 time and take screenshots, then measure the length from the hair image to the head's position. All measurements are recorded and the average error in length should be within 1 millimeter.

2. PA-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Moving hair image on the user's head on the screen

How test will be performed: Test will be done manually by developers running prototype. The developer will test the hairstyle functionality 100 time with moving developer's head and record the screen. Each scene in the recording video will be made into screenshots, then developers measure the length from the hair image to the head's position. All measurements are recorded and the average error in length should be within 2 millimeter.

Reliability and Availability Requirements

1. RA-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via

a survey asking them "How many times have you faced a crash when using the application?" The total number will be add up and divided by the number of users to get the crash percentage. And the crash percentage should be less than 2%

2. RA-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "How many times have you get an error message or any unexpected wrong behavior of the application when using the application?" The total number will be add up and divided by the number of users to get the expected error message percentage, and the percentage should be greater than 98%

Robustness Requirements

1. RB-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. The developer will try to inject irrational inputs to break the application and record the number of times that the application has an systematic error message(not error message from error exception handling) or any unexpected wrong behavior. The number of times that the application crashes is divided by the total number

of injected irrational inputs to get a percentage and this percentage should be less than 3%

2. RB-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. The developer will try to inject irrational inputs to break the application and record the number of times that the application crashes.

Capacity Requirements

1. CC-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. At 10 different time points, the database developer will check if the number of user's information in database matches the actual number of users. Any mismatch will be recorded.

2. CC-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. Developer will check at any 100 different time points, the database stores at least 10 different hairstyles. Any mismatch will be recorded.

Scalability and Extensibility Requirements

1. SE-T1

Type: Structural, Dynamic, Manual

Initial State: Application installed onto a cell phone and running

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Any possible future developing features will be kept in the section of Watch Room in the SRS document.

Longevity Requirements

1. LG-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: The developer will perform the tests written in this document every month to ensure the application is functioning and meeting requirements.

5.2.4 Operational and Environmental Requirements

Productization Requirements

1. PD-T1

Type: Structural, Dynamic, Manual

Initial State: Application is not yet installed

Input/Condition: User operations

Output/Result: Search result of "Hairesthetics" application is shown in App Store

How test will be performed: Test will be done manually by user searching "Hairesthetics" in App Store. Tests will be enhanced by actual user feedback via a survey asking them "Can you find Hairesthetics application on App Store?"

Release Requirements

1. RE-T1

Type: Structural, Dynamic, Manual

Initial State: Application installed onto a cell phone and running

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Developers will run tests in this document to ensure that data can be used between different builds.

2. RE-T2

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~ Application running on a computer website

Input/Condition: Version of Application is updated. Application is updated and running

Output/Result: Update Log windows pop up

How test will be performed: After application is updated, users will be asked in a survey "Have you seen the pop-up window for the update log?"

5.2.5 Maintainability and Support Requirements

Maintenance Requirements

1. MS-T1 and MS-T2

Type: Structural, Dynamic, Manual

Initial State: Application is published and able to be ~~downloaded~~ **accessed** from ~~app-store~~ **website**.

Input/Condition: Developer input

Output/Result: Test results

How test will be performed: Developers collect feed backs from user and fix any bugs reported. Once a month, the developing team will also run the existing tests.

2. MS-T3

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~ **Application running on a computer website**

Input/Condition: System failure occurs to the app

Output/Result: whether the system recovers within time

How test will be performed: Developers would try to crash the app and record the time for recovering 10 times. Then the team will calculate the average recovering time to see whether it is within 1 hour.

Supportability Requirements and Adaptability Requirements

1. SP-T1 and AD-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "Is your iPhone with iOS 14 and later? If yes, is the application running on your phone?"

5.2.6 Security Requirements

Access Requirements

1. AC-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "Are you able to access images previously stored on the application?"

5.2.7 Cultural and Political Requirements

Cultural Requirements

1. CR-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Response from user inputs

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "Do you think that any terms in the application is inappropriate or offensive toward your or others culture?"

5.2.8 Legal Requirements

Compliance Requirements

1. CP-T1

Type: Structural, Dynamic, Manual

Initial State: ~~Application installed onto a cell phone~~Application running on a computer website

Input/Condition: User inputs

Output/Result: Pop up windows for the legal statements

How test will be performed: Test will be done manually by developers running prototype. Tests will be enhanced by actual user feedback via a survey asking them "Have you seen the pop up windows of the legal statements when you start the application for the first time?"

6 Unit Test Plan

The pytest framework will be used as unit testing too for this project. In unit testing, each "unit" of our code is tested as fully as possible, as an isolated chunk of code. A "unit" stands for a function or a method in our system.

6.1 Unit Testing Scope

Unit tests are narrow in scope, and allow us to cover all cases, ensuring that every single part works correctly. Unit testing typically have to test every function, method, object or component in our design. In our application, unit testing should mainly focus on testing the accuracy of the AR output on face

capture through camera, which will compare the similarity of expected look with the actual look. Additionally, other compatible components need to be tested as well. In order to make single unit test pass and achieve a high coverage rate, we will use test coverage metrics to measure and monitor the testing activity

Modules outside of the scope:

- It interact with the database
- It uses interface through the network
- It communicates with the file system
- It cannot be ran automatically using the unit testing tool with other unit tests at the same time
- It needs mandatory operations (like modify makefile, configfile, environment variables) to run it

More specifically, most of the non-functional requirements cannot be verified by unit testing. Non-functional tests are mostly categorized as a performance test and cannot be tested quantitatively.

7 Traceability Matrix

	FR1	FR2	FR3	FR6	FR7	HM7	HM8	HM9	HM11	HM12
HS-T1	X	X	X	X	X				X	X
HS-T2	X									X
HS-T3										X
HS-T4	X	X	X	X	X	X	X	X	X	X

Table 2: Traceability Matrix Showing the Connections Between Functional Requirements and functional requirements tests

	FR1	FR2	FR3	FR6	FR7	HM1	HM2	HM3	HM4	HM5	HM6
HC-T1											
HC-T2	X										X
HC-T3	X	X	X	X	X	X	X	X	X	X	X
HC-T4	X	X	X	X	X	X	X	X	X	X	X

Table 3: Traceability Matrix Showing the Connections Between Functional Requirements and functional requirements tests

	HR1	HR2	HR3	HR4
HSR-T1				
HSR-T2	X	X	X	X
HSR-T3	X	X	X	
HSR-T4	X	X	X	X
HSR-T5	X		X	
HSR-T6	X			X

Table 4: Traceability Matrix Showing the Connections Between Functional Requirements and functional requirements tests

	APR1	APR2	IAR1	IAR2	EUR1	EUR2	LR1	LR2	UPR1	ASR1	SLR1	SLR2	SLR3
AP-T1	X												
AP-T2		X											
IA-T1			X										
IA-T2				X									
EU-T1					X								
EU-T2						X							
LR-T1							X						
LR-T2								X					
AS-T1										X			
SL-T1											X		
SL-T2												X	
SL-T3													X

Table 5: Traceability Matrix Showing the Connections Between non-functional Requirements and non-functional requirements tests

	PAR1	PAR2	RAR1	RAR2	RBR1	RBR2	CCR1	CCR2	SER1	LGR1	PDR1	RER1	RER2
PA-T1	X												
PA-T2		X											
RA-T1			X										
RA-T2				X									
RB-T1					X								
RB-T2						X							
CC-T1							X						
CC-T2								X					
SE-T1									X				
LG-T1										X			
PD-T1											X		
RE-T1												X	
RE-T2													X

Table 6: Traceability Matrix Showing the Connections Between non-functional Requirements and non-functional requirements tests

	MSR1	MSR2	MSR3	SPR1	ADR1	ACR1	ACR2	ACR3	ACR4	CR1	CPR1
MS-T1	X	X									
MS-T2			X								
SP-T1				X	X						
AC-T1						X	X	X	X		
CR-T1										X	
CP-T1											X

Table 7: Traceability Matrix Showing the Connections Between non-functional Requirements and non-functional requirements tests

8 Appendix

8.1 Symbolic Parameters

The definition of the test cases will call for `SYMBOLIC_CONSTANTS`. Their values are defined in this section for easy maintenance.

8.2 Usability Research Plan

For our project, the interface and user experience would be very important. In order to conduct the best user friendly application, we decided to make a user research by two methods. First of all, we are planning to prepare a usability survey and make small interviews with users interested in such apps. By gathering suggestions from public we can make small changes to make our product easier to use. Besides, we are also planning to ask guests to test use our application and gain useful feedbacks once it is implemented. By making small changes accordingly before submitting may improve the usability experience significantly.

8.3 Usability Survey Questions

- Would you prefer to have a tutorial of our application by text or video?
- Would you prefer to have all the features hidden or present when virtually presenting the hairstyle?
- Would you prefer to have cartoon style interface or modern style interface?

8.4 Reflection

8.4.1 Senni

More knowledge on dynamic testing will be acquired for the team to successfully complete the verification and validation of the project. In the system test for nonfunctional requirement, most of the tests are plan to be done by the user and gathering feedback from the user survey, a more generic approach or a more generic measure tool or methods are required to give a more thorough test.

To acquire the knowledge of better testing nonfunctional requirements, there are two approaches: 1. do research on any measurement tools for dynamic testing 2. interview with experienced senior developer in the industry or project manager in the industry to learn more knowledge about testing nonfunctional requirements with dynamic testing. After discussion, we decided to take the second approach because measurement tools for dynamically testing nonfunctional requirement is very rare in the current industry, and all members had connections to senior developers or managers in the software industry, and thus it will be a good way to interview with the seniors and learn more about testing for nonfunctional requirement in dynamic testing.

8.4.2 Bill

I am responsible for conducting verification and validation of the functional testing in our project. On the back-end side of our application, the team chose to use a testing framework called Pytest. However, I am not familiar with this tool, I need to acquire more knowledge and information regarding to Pytest's testing structure and syntax. For example, ways of unit testing, integration testing, and code coverage analysis using Pytest. In addition, more of the testings are related to image processing and transformation, I need to learn how to conduct these test efficiently.

The first action I would take is to talk to professionals and friends who have extensive experiences in testing image processing and transformation. I want to formulate some ideas and strategies, then try to apply them using Pytest. This method would be time-efficient, since they gave me a general direction on testing based on their professional experiences. The second approach is looking at information available online, some resources provide

useful information and solutions to specific domains I might encounter.

8.4.3 Jack

In this document, I am mainly responsible for provide tests method for some of the non-functional requirements and also conclude all the tests method into a tractability matrix. For non-functional requirements, the tests mostly reply on manual and user input. Our team needs to conduct physical tests to ensure that the application meets all the requirements. Besides, I have also added a matrix to trace which test can ensure which requirements.

There are a few knowledge I need to learn before the actual testing begin. First of all, we need to come up with a plan to conduct these tests effectively. For example, finding suitable people to test our app after it is being developed. Secondly, I will be coffee talking to other project managers and learn about their non-functional requirements testing procedures. Last but not least, I will also go through some testing reports on the open source projects on GitHub to gain more knowledge about testing methods.

8.4.4 Charlotte

For this VnV plan, I learnt that in software testing, verification and validation (VV) is the process of checking that a software system meets specifications and requirements so that it fulfills its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle. In simple terms, software verification is: "Assuming we should build X, does our software achieve its goals without any bugs or gaps?" On the other hand, software validation is: "Was X what we should have built? Does X meet the high-level requirements?".

Verification and validation are not the same things, although they are often confused. Verification is asking the question: Are we building the product right? Validation is asking the question: Are we building the right product? "Building the product right" checks that the specifications are correctly implemented by the system while "building the right product" refers back to the user's needs. In some contexts, it is required to have written requirements

for both as well as formal procedures or protocols for determining compliance.

8.4.5 Marlon

During the development of the VnV plan, I was responsible for coming up with the verification plan. One message I learnt is the importance of getting reviews from people other than the development group, which also reflects the needs for roles like software engineer in testing in the industry. During the software development life cycle, the developers usually strictly follow the development plan and ideas when contributing to a grand software. Thus, it's hard for them to jump out of the box and realize their mistakes through their development process. Additionally, all the documents are required to be inspected as well in case any ignorance causing problems forming in the early stages.

The process also rings a bell for me to start learning testing frameworks required when creating automation for my assigned parts in the project. I will be accompany with Bill to test the backend API and algorithm. One important tool used for testing API is Postman. I will need to learn how to integrate it into part of the pytest framework. The best way to learn how to set up the testing automation properly is to consult someone with QA experiences, such as colleagues from our group during my internship. It will also help to start some research online since I already have some experiences working with both postman and pytest.

8.4.6 Hongwei

Based on development of this document, I learned Verification process (static testing) includes checking documents, design, code, and program, whereas Validation process (dynamic testing) includes testing and validation of the actual product. For example, Dynamic testing contains the steps of Test case design, Test environment step-up, Test case execution, Test analysis and evaluation, Bug Reporting. More specifically, in the hierarchy of Dynamic testing - Black-box testing - Functional testing, there are Unit testing, Integration testing, System testing, User acceptance testing. In unit testing, we will be using pytest to test each component in our design.

Two possible approaches to acquiring related knowledge and get hands-on

experience with the tool: First, do research on pytest.org and other open-source community, run and understand examples, try to integrate our design into pytest. Secondly, talk to friends and former colleagues who has related experience. After analysing pros and cons of both methods, we choose to use the second one, since there is a limitation in online resources and the teammate (Marlon) who work with me has industrial-level experience in pytest.

Appendix — References

- Team 18, Software Requirements Specification for Hairesthetics: A 3D Hairstyle Simulation iOS App, 2022. [Online]. Available: <https://github.com/marlon4dashen>
- Team 18, Module Guide for Hairesthetics: A 3D Hairstyle Simulation iOS App, 2022. [Online]. Available: <https://github.com/marlon4dashen/Hairesthetics/tree/r>
- Team 18, Development Plan for Hairesthetics: A 3D Hairstyle Simulation iOS App, 2022. [Online]. Available: <https://github.com/marlon4dashen/Hairesthetics/t>
- Team 18, Hazard Analysis for Hairesthetics: A 3D Hairstyle Simulation iOS App, 2022. [Online]. Available: <https://github.com/marlon4dashen/Hairesthetics/tree/r>