

Module Guide for HairEsthetics

Team 18

Charlotte Cheng

Marlon Liu

Senni Tan

Qiushi Xu

Hongwei Niu

Bill Song

April 5, 2023

1 Revision History

Date	Version	Notes
Jan 16	1.0	Rev0 MG
Apr 3	2.0	Revision 1

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
AI	Artificial Intelligence
AR	Augmented Reality
API	Application programming interface
App	Application
DAG	Directed Acyclic Graph
M	Module
macOS	Operating system developed by Apple Inc
MG	Module Guide
MIS	Module Interface Specification
ML	Machine Learning
OS	Operating System
R	Requirement
REST	Representational state transfer
RGB	Red, Green, Blue
SC	Scientific Computing
SRS	Software Requirement Specification
UC	Unlikely Change
UI	User Interface
VnVPlan	Validation and Verification Plan

Contents

1	Revision History	i
2	Reference Material	ii
2.1	Abbreviations and Acronyms	ii
3	Introduction	1
3.1	Overview	1
3.2	Context	1
3.3	Design Principles	1
3.4	Design Pattern	1
3.5	Document Structure	1
4	Anticipated and Unlikely Changes	2
4.1	Anticipated Changes	2
4.2	Unlikely Changes	3
5	Module Hierarchy	3
6	Connection Between Requirements and Design	5
7	Module Decomposition	5
7.1	Hardware Hiding Modules	6
7.1.1	Camera Module	6
7.2	Behaviour-Hiding Module(M2, M3, M4, M5, M6, M7, M10, M11, M12, M13, M14, M15, M16, M17)	6
7.2.1	App Module (M1)	6
7.2.2	Server Module (M2)	7
7.2.3	Hair Color Module (M3)	7
7.2.4	Hair Style Module (M4)	7
7.2.5	Worker Module (M4)	7
7.2.6	Image Worker Module (M5)	8
7.2.7	Salon Recommendation Module (M6)	8
7.2.8	Hair Artist Module (M7)	8
7.2.9	Hair Color View Module (M10)	8
7.2.10	Hair Style View Module (M11)	9
7.2.11	Salon Recommendation View Module (M12)	9
7.2.12	Home View Module (M13)	9
7.2.13	Error View Module (M12)	9
7.2.14	Launch View Module (M14)	9
7.2.15	Footer Module (M14)	10
7.2.16	NavBar Module (M15)	10
7.2.17	AR Canvars Module (M16)	10

7.2.18	Hair Model Module (M17)	10
7.3	Software Decision Module (M8, M9, M18)	11
7.3.1	ML Model Module (M6) Model Utils Module (M8)	11
7.3.2	Utility Module (M7) Image Utils Module (M9)	11
7.3.3	ThreeFiber Helper Module (M18)	11
8	Traceability Matrix	12
9	Use Hierarchy Between Modules	13
9.1	Use Hierarchy Diagram	14
9.2	UML Diagram	14

List of Tables

1	Module Hierarchy	5
2	Trace Between Requirements and Modules	12
3	Trace Between Anticipated Changes and Modules	13

List of Figures

1	Use hierarchy among modules	14
2	UML Diagram Part 1	15
3	UML Diagram Part 2	16

3 Introduction

3.1 Overview

This project is to develop a software application that allows users to virtually simulate hairstyles and hair colors. The goal is to create a user-friendly, functional, and robust application that meets the essential functional requirements and non-functional requirements. The end result will be a software application that helps users easily and effectively experiment with different hairstyles and hair colors, providing an immersive and personalized virtual styling experience.

3.2 Context

This Module Guide(MG) of our project is developed after the Software Requirements Specification(SRS). The purpose of the MG is to explain the structure of modules based on selected design principles and patterns which further helps understand the project's functionalities based on the SRS. It also briefly introduces the services each module provides.

3.3 Design Principles

The design principles which would be used in the MG include information hiding, high cohesion, low coupling, high fan-in, and low fan-out for module decomposition. The information-hiding principle includes identifying and encapsulating the expected changes.

3.4 Design Pattern

Our project follows the MVC (model-view-controller) design pattern. MVC allows us to apply the separation of concern principle and breaks the complex problem into smaller sub-problems that can be solved in different modules.

3.5 Document Structure

- Anticipated and Unlikely Change
- Module Hierarchy: It places modules related to the functional requirements in the behaviour hiding module.
- Connection Between Requirements And Design: It provides the decisions of design that are needed to be made to realize the requirements
- Module Decomposition: It explains each module by following design principles.
- Traceability Matrices: One matrix is between modules and requirements, the other is between modules and anticipated changes.

- Uses Hierarchy: Diagram explains user hierarchy.
- Project Schedule
- Revision History

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

- AC1: This project will only result in the development of a MacOS app **web application**.
~~However, an android application may also be developed to work with, or in place of, the MacOS application in the future.~~
- AC2: ~~he application may be extended to offer compatibility as a web application.~~
- AC3: The application will not initially pursue functionality for users to “cut” their 3D simulated hair.
- AC4: ~~The application may require more data for the facial recognition feature.~~
- AC5: The load capacity of the system is initially set low for the first edition of this application. As demand for the application increases, the load capacity must also increase to support a larger user base.
- AC6: The application may be required to include new features as discovered through use of the system in real-world scenarios.
- AC7: The format of the input and output data of facial recognition, hair modification, hair salon recommendation system.
- AC8: The interface style requirements might change since no metric has been provided for this by stakeholders.
- AC9: The speed and latency requirements might change since no metric has been provided for this by stakeholders.
- AC10: The precision and accuracy requirements might change since no metric has been provided for this by stakeholders.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decisions should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

- ULC1: Since several of the features might require exposing API endpoints, an offline-only version of the application is unlikely to be necessary.
- ULC2: As the main demographic of this application is less computer literate, the ease of use of this application must remain a high priority consideration throughout the design of this system.
- ULC3: The application is designed to ensure that data will only be modified when necessary.
- ULC4: The application is designed to ensure that only admins and supervisors will be able to unlock locked resources.
- ULC5: The application is designed to ensure that no inappropriate or offensive language towards any culture will be used.
- ULC6: The application is designed to ensure compliance with the Data Privacy Act of Canada. All user personal information will not be stored or used without consent.
- ULC7: The application is designed to ensure that users are only able to see data that they have proper authorization for.
- ULC8: The application is designed to help users simulate hairstyles and hair colors virtually.
- ULC9: The primary user of this project is unlikely going to change since the project purpose is determined already, only certain types of users will be interested in this kind of application.
- ULC10: The client of this project is unlikely going to change because this project is a student capstone project and it has limited marketability.
- ULC11: The schedule and budget constraints are not likely going to change, because this project is a student capstone project. It needs to strictly follow SE4G06 course constraints.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: ~~Controller Module~~ App Module
M2: Server Module
M3: Hair Color Module
M4: Worker Module
M5: Image Worker Module
M6: Salon Recommendation Module
M7: Hair Artist Module
M8: Model Utils Module
M9: Image Utils Module
M10: Hair Color View Module
M11: Hair Style View Module
M12: Salon Recommendation View Module
M13: Home View Module
M14: ~~Error View Module~~ Footer Module
M15: ~~Camera Module~~ NavBar Module
M16: ~~Launch View Module~~ AR Canvas Module
M17: HairModel Module
M18: ThreeFiber Helper Module

Level 1	Level 2
Hardware-Hiding Module	M13 M1
	M1
	M2
	M3
Behaviour-Hiding Module	M4
	M5
	M6
	M7
	M10
	M11
	M12
	M13
	M14
	M15
	M16
	M17
Software Decision Module	M6
	M7
	M8
	M9
	M18

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

7 Module Decomposition

Modules are decomposed according to the principle of “information hiding” proposed by David Parnas. The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do

without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *ProgName* means the module will be implemented by the ProgName software.

The software architecture we use is the model-view-controller(MVC) architecture, it provides the principle of separation of concerns for our system as well as convenience for future addition and modification to certain components. So we will follow the MVC architecture while decomposing our modules.

7.1 Hardware Hiding Modules

Secrets: The implementation of the hardware adaptation.

Services: This module would allow the software to change and adapt the changes of hardware environment.

Implemented By: Python

7.1.1 ~~Camera Module~~

Secrets: The implementation of access to camera.

Services: Provide access for the app to the device's camera.

Implemented By: Python

7.2 Behaviour-Hiding Module(M2, M3, M4, M5, M6, M7, M10, M11, M12, M13, M14, M15, M16, M17)

Secrets: The implementation of the behaviors of our application after processing user input

Services: The behavior-hiding modules includes both user-interface modules and backend logic modules. They serve as a communication layer between the hardware-hiding module and the software decision module.

7.2.1 App Module (M1)

Secrets: The implementation of the entire application

Services: The app module initializes the web application and manages the routes in the frontend.

Implemented By: React

Type of Module: Abstract Object

7.2.2 Server Module (M2)

Secrets: The server endpoints and WebSocket implementations

Services: It routes the requests to corresponding endpoints and establishes workers to handle the requests.

Implemented By: Python, Flask, SocketIO

Type of Module: Abstract Object

7.2.3 Hair Color Module (M3)

Secrets: Algorithm for detecting the hair and changing its color on detected faces

Services: Perform hair re-coloring based on hair segmentation results.

Implemented By: Python

Type of Module: Abstract Object

7.2.4 ~~Hair Style Module (M4)~~

~~**Secrets:** Algorithm for adding new hair on detected faces~~

~~**Services:** Obtain position and rotation degree for adding AR hair model based on input user faces.~~

~~**Implemented By:** Python~~

~~**Type of Module:** Abstract Object~~

7.2.5 Worker Module (M4)

Secrets: The details of how input image frames are handled and results are produced.

Services: Handle input frames with individual threads that perform hair color change.

Implemented By: Python, Thread

Type of Module: Abstract Data Type

7.2.6 Image Worker Module (M5)

Secrets: The details of how image files are handled and results are produced.

Services: Handle input image file and produce hair color change.

Implemented By: Python

Type of Module: Abstract Data Type

7.2.7 Salon Recommendation Module (M6)

Secrets: The implementation of nearby salon recommendation.

Services: Highlight the nearby salons on the map according to the input address and provide a listing that can be filtered by rankings and distances.

Implemented By: Python, Google Map API

Type of Module: Abstract Object

7.2.8 Hair Artist Module (M7)

Secrets: Encapsulates the necessary models, library, and functions to perform hair color change.

Services: Serve as the model and algorithm interface for workers to interact with.

Implemented By: Python

Type of Module: Abstract Data Type

7.2.9 Hair Color View Module (M10)

Secrets: Algorithms and data structure for the hair color changing screen.

Services: Display the resulted color of hair when user changes the hair color.

Implemented By: **React**

Type of Module: Abstract Object

7.2.10 Hair Style View Module (M11)

Secrets: Algorithms and data structure for hairstyle changing screen.

Services: Put the selected AR hair model to the detected face when the user presses the buttons accordingly.

Implemented By: React

Type of Module: Abstract Object

7.2.11 Salon Recommendation View Module (M12)

Secrets: Algorithms and data structure for salon recommendation screen.

Services: Display nearby hair salon and display information about a chosen salon.

Implemented By: React

Type of Module: Abstract Object

7.2.12 Home View Module (M13)

Secrets: Algorithms and data structure for displaying the home screen.

Services: Provide basic guidance for the users to start using the app.

Implemented By: React

Type of Module: Abstract Object

7.2.13 ~~Error View Module (M12)~~

~~**Secrets:** Algorithms and data structure for displaying the error screen.~~

~~**Services:** Provide details and guidance to errors happening in the application.~~

~~**Implemented By:** Swift~~

~~**Type of Module:** Abstract Object~~

7.2.14 ~~Launch View Module (M14)~~

~~**Secrets:** Algorithms and data structure for displaying the error screen.~~

~~**Services:** Provide details and status when loading the application.~~

~~**Implemented By:** Swift~~

~~**Type of Module:** Abstract Object~~

7.2.15 Footer Module (M14)

Secrets: Implementation of the footer of the web page.

Services: Serve as a component at the bottom of the web page.

Implemented By: React

Type of Module: Abstract Object

7.2.16 NavBar Module (M15)

Secrets: Implementation of the NavBar at the top of the web page and the navigation to other pages from the NavBar.

Services: Serve as the component at the top of the web page and provide navigation to other pages.

Implemented By: React

Type of Module: Abstract Object

7.2.17 AR Canvars Module (M16)

Secrets: Implementation of face detection for the AR (ThreeFiberHelper Module).

Services: Serve as the component that provides the face detection for the AR module (ThreeFiberHelper Module).

Implemented By: React

Type of Module: Abstract Object

7.2.18 Hair Model Module (M17)

Secrets: Implementation of hair AR models.

Services: Serve as the models that provide different hair styles.

Implemented By: React

Type of Module: Abstract Object

7.3 Software Decision Module (M8, M9, M18)

Secrets: The design decision based on mathematical models, and real-world objects in the form of data structures. It also includes libraries required to support behavioral functionalities.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: Python

7.3.1 ~~ML Model Module (M6)~~ **Model Utils Module (M8)**

Secrets: Data structure for representing a pre-trained model

Services: This module will provide functions to load the pre-trained model for hair segmentation if it's not already loaded. It also provides functions to interact with the model easily.

Implemented By: Python

Type of Module: Abstract Object

7.3.2 ~~Utility Module (M7)~~ **Image Utils Module (M9)**

Secrets: Utility functions for image processing

Services: This module will provide utility services and operations required for performing functional behaviors.

Implemented By: Python

Type of Module: Library

7.3.3 **ThreeFiber Helper Module (M18)**

Secrets: Utility functions for applying AR hairstyles

Services: This module will provide utility services and operations required for applying AR hair style models.

Implemented By: React

Type of Module: Abstract Object

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
FR1	M8, M9 M16, M18
FR2	M7, M16
FR3	M2, M6 M7
FR6	M2 M16
FR7	M2, M6 M16, M17
HM1	M8 10
HM2	M3
HM3	M3
HM4	M4, M6 M5
HM5	M9 M10
HM7	M7
textcolorredM11	
HM8	M4 M16, M18
HM9	M8, M9 M16
HM10	M3, M6
HM11	M3, M6 M17
HM12	M3, M6 M11
HR1	M10 M12
HR2	M5 M6
HR3	M10 M12
HR4	M5 M6

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M10, M11, M12, M13, M14, M15, M16, M17, M18
AC2	Not applicable
AC3	M4, M9 M16
AC4	M6
AC5	Not applicable
AC6	M1 M2
AC7	M2, M4, M5 M3, M6, M16
AC8	M8, M9, M10, M11, M12 M10, M11, M12, M13, M14, M15, M16, M17, M18
AC9	M5, M10 M3, M4, M5
AC10	M4, M9

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 3 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

9.1 Use Hierarchy Diagram

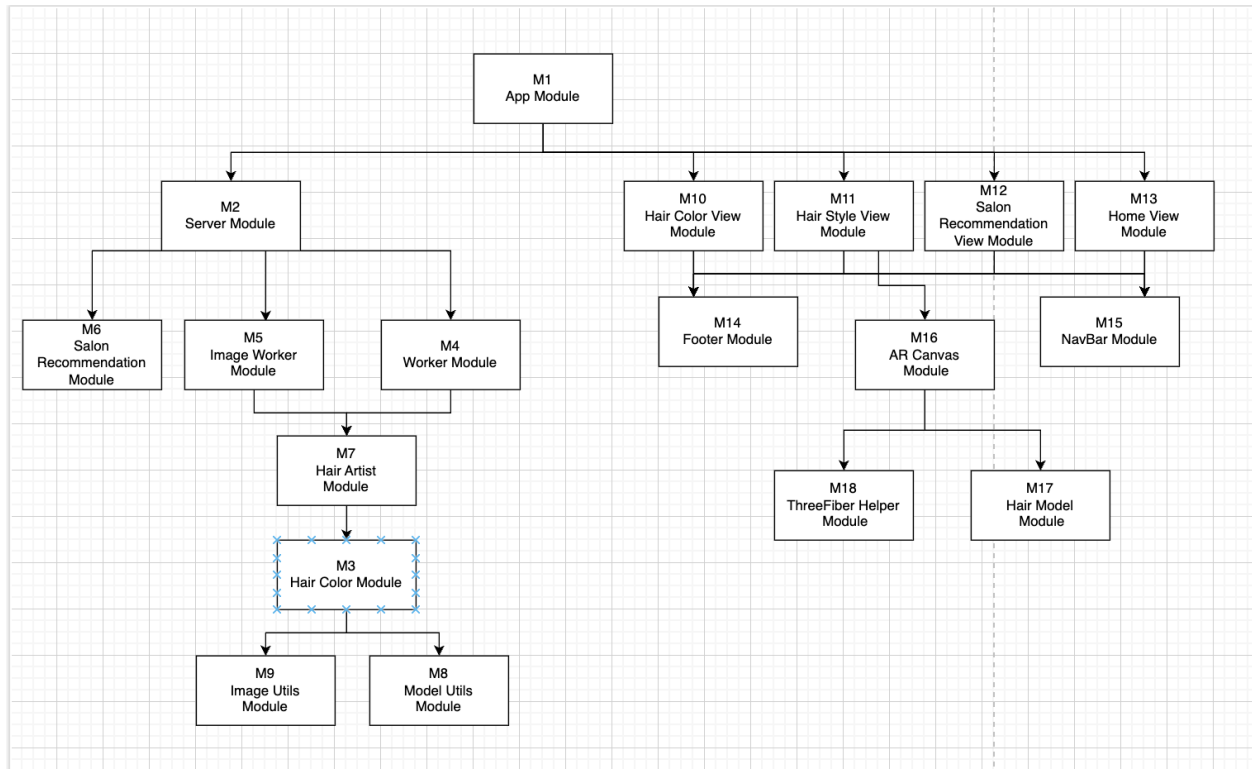


Figure 1: Use hierarchy among modules

9.2 UML Diagram

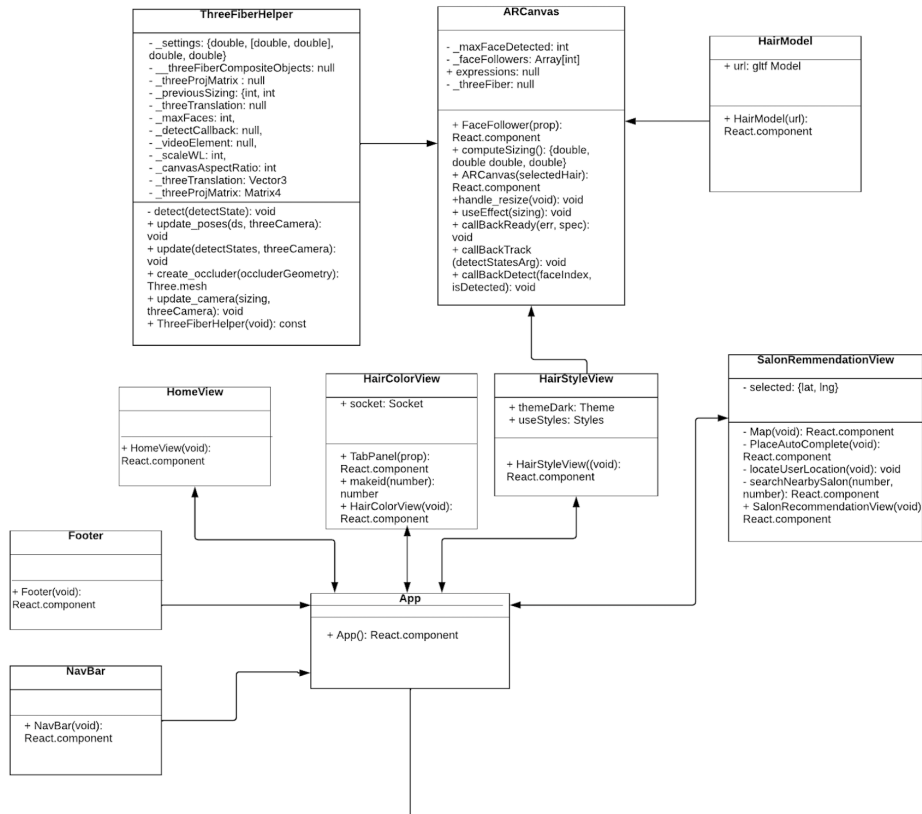


Figure 2: UML Diagram Part 1

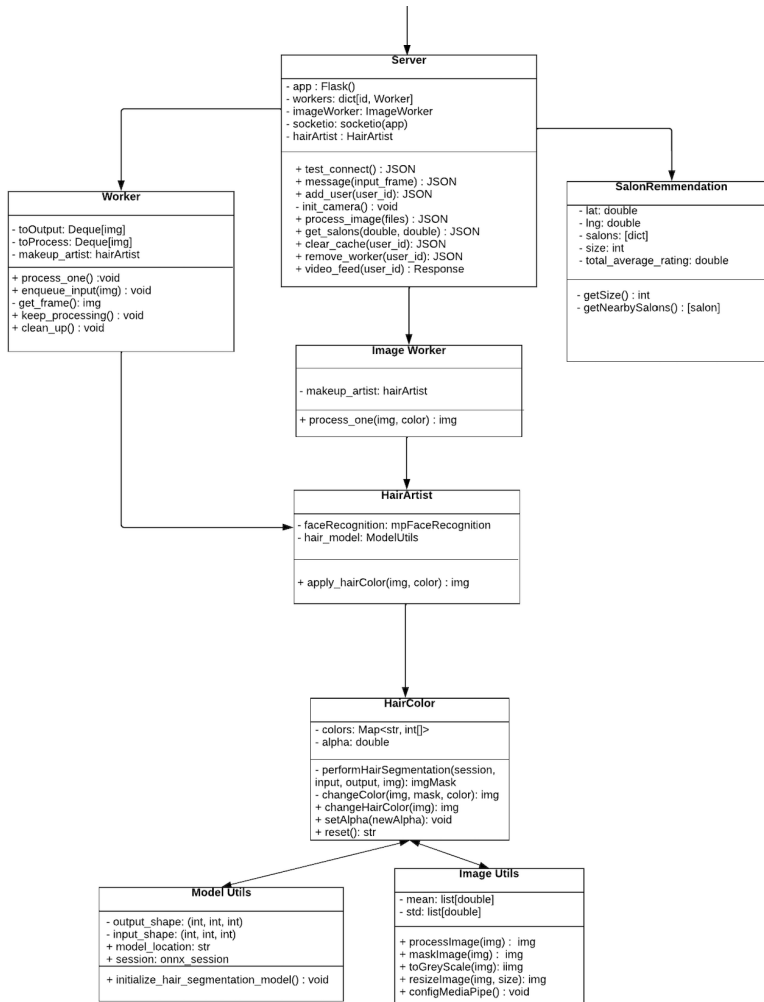


Figure 3: UML Diagram Part 2