Hairdressing Project

Software Requirements Specification

1.0

21/08/2020

Diego C., Gerardo G.

Development Team

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 21/08/2020 | Initial draft | Diego C. |  |
| 27/08/2020 | Added diagrams and other sections. Second review | Diego C., Gerardo G. |  |
| 28/08/2020 | Added definitions and references | Diego C. | Removed some sections deemed irrelevant |
|  |  |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Diego C. | Developer |  |
|  | Gerardo G. | Developer |  |
|  |  |  |  |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 References 3

1.5 Overview 3

2. General Description 3

2.1 User Characteristics 4

2.2 General Constraints 4

2.3 Assumptions and Dependencies 4

3. Specific Requirements 5

3.1 External Interface Requirements 5

3.1.1 User Interfaces 5

3.1.2 Hardware Interfaces 5

3.1.3 Software Interfaces 5

3.1.4 Communications Interfaces 5

3.2 Functional Requirements 5

3.2.1 User authentication / authorisation 5

3.2.2 History of pictures and selected face shapes / skin tones / hair styles 5

3.2.3 Pictures of pre-set models showcasing each feature (hair colour, hair style, etc.) 6

3.2.4 Diagrams or galleries presenting the options available for each feature 6

3.2.5 Colour picker 6

3.2.6 Image preview based on users’ input 7

3.2.7 Image upload 7

3.2.8. Additional hair features as reference 8

3.3 Use Cases 9

3.3.1 Sign up 9

3.3.2 Sign in 10

3.3.3 Upload pictures 11

3.3.4 Select a feature (e.g. hair colour, hair style, etc.) 12

3.3.4 Comparing results to original picture 13

3.4 Classes / Objects 14

3.4.1 Picture 14

3.4.1.1 Attributes 14

3.4.1.2 Methods 14

3.4.2 History 14

3.4.2.1 Attributes 14

3.4.2.2 Methods 14

3.4.3 Image service 14

3.4.3.1 Attributes 14

3.4.3.2 Methods 15

3.5 Non-Functional Requirements 15

3.5.1 Usability 15

3.5.2 Interactivity 15

3.5.3 Responsiveness 15

3.5.4 Extendibility 15

3.5.5 Maintainability 15

3.5.6 Security 15

3.5.7 Performance 16

3.6 Inverse Requirements 16

3.7 Design Constraints 16

3.8 Logical Database Requirements 17

3.9 Other Requirements 17

4. Analysis Models 17

4.1 Sequence Diagrams 17

4.3 Data Flow Diagrams (DFD) 18

4.2 State-Transition Diagrams (STD) 18

5. Change Management Process 19

A. Appendices 19

A.1 Appendix 1 19

A.2 Appendix 2 19

# 1. Introduction

This document describes general and specific requirements that should be implemented in this project. It includes details that should be relevant both to developers and non-technical parties.

It initially outlines the purpose and scope along with dependencies (references) of the project, along with constraints and assumptions. General and specific requirements are then listed for both the API and the app, followed by functional and non-functional requirements.

Any information that is omitted or not clear for the reader should be notified to the developers, so that they can rectify the issue as soon as possible and contact other stakeholders to confirm changes.

## 1.1 Purpose

Based on requirements that have been gathered and agreed upon with the clients, the purpose of this document is to provide a general set of guides that should aid the development process of this project, including scope, functional and non-functional requirements and diagrams.

This document is intended for software developers and/or engineers who are interested in the overall structure of this project. It is assumed that they have some degree of familiarity with app development.

It is also intended for clients to validate the development process according to agreements with the development team.

## 1.2 Scope

The following software and services will be used for the development environment of the app:

* Operating System: Windows 10 (development), Ubuntu 18.04 LTS (production)
* IDEs: Visual Studio, Visual Studio Code, Android Studio
* DBMS: MariaDB, Amazon RDS, Amazon DynamoDB
* Programming languages: C#, Dart, Python, SQL
* Frameworks: Flutter, ASP.NET Core, FastAPI
* Storage: Amazon S3
* Source control: GitHub
* Communication: Slack
* Collaboration: Asana

For production environment, a Linux OS was chosen because it provides more flexibility when using different open source software and this makes it cheaper to maintain.

Visual Studio Code is the chosen IDE as it provides lots of useful extensions to speed up the development process. It also allows to connect remotely to a virtual host.

MariaDB was chosen instead of MySQL as MariaDB does not have licensing issues.

FastAPI was chosen as a Python REST framework because for this project we are not serving webpages, so we only need a simple REST API with good performance benchmark, and also it is used to run Python scripts and libraries to process images.

Amazon S3 service will be used to store all images.

At the very least, the app to be developed WILL provide:

* User accounts (sign in / sign up)
* Picture upload (from the user’s gallery or from a new picture taken)
* History of pictures uploaded and their respective changes
* Ability to change hair styles, face shapes and hair colours of pictures uploaded
* Saving pictures locally
* Comparison between new pictures generated from changes selected and the original picture

The app WILL NOT provide:

* 3D imagery (rotating heads)

For more details about the scope of this project, see the Scope Document as referenced in section 1.4 (References).

## 1.3 Definitions, Acronyms, and Abbreviations

API: Application Programming Interface – A layer that serves as a communication channel between data and services that use it

REST: Representational State Transfer – Architectural style most used in the context of building APIs through actions (verbs) and resources (nouns).

IDE: Integrated Development Environment – A comprehensive software that encompasses text editing capabilities, debugging tools, integration with version control systems, among other features.

DBMS: Database Management System – Software used by developers and database administrators to access, create and modify databases.

ML: Machine Learning – A field of AI (Artificial Intelligence) that involves using algorithms in order to train computers to predict certain results based on user input.

Agile – A set of principles that guide certain software development methodologies which aims to break down the development process into iterations that more accurately satisfy the needs of the client(s) over time.

User authentication – A process that verifies whether a user account is valid (or whether they are logged in).

User authorisation – A process that verifies whether a user can access certain sections of the app.

## 1.4 References

Documents that are missing or not referenced in this section can be accessed by contacting the development team.

Contacts:

* Diego C.: [20026893@tafe.wa.edu.au](mailto:20026893@tafe.wa.edu.au)
* Gerardo G.: [20026902@tafe.wa.edu.au](mailto:20026902@tafe.wa.edu.au)

Referenced documents:

* Scope document: <https://github.com/HairdressingProject/styleme/blob/master/Documentation/ScopeDocument.docx>
* Implementation plan: <https://github.com/HairdressingProject/styleme/blob/master/Documentation/SoftwareDevImplementPlan.xlsx>
* Project plan: <https://github.com/HairdressingProject/styleme/blob/master/Documentation/Project%20Plan.docx>
* Requirements prioritisation matrix:

## 1.5 Overview

This Software Requirements Specification is organized into five different sections as in the table of contents.

The **General Description** section gives an overview of the requirements which will be described in greater detail under the **Specific Requirements** section.

Subsections of the overall description are product perspective, product functions, user characteristics, general constraints and assumptions and dependencies.

The purpose of the specific requirements section is to detail all the requirements necessary for development. This section provides a framework for designers to create the product in accordance with requirements.

# 2. General Description

Factors that influence the requirements outlined in this document, the software choices made and how the app is expected to behave include:

* Degree of familiarity of the target audience with the main feature of the app
* Optimal response times expected of mobile apps
* Due to graphical features presented, the layout should be well designed
* Factors that influence the cost the most (such as storing images and hosting the backend infrastructure of the app) should be minimal
* The agile nature of this project benefits from technologies that provide the best prototyping capabilities

## 2.1 User Characteristics

As the main target audience of this app includes hairdressing lecturers and students, it is important to provide as many customisation options as possible for changes made to pictures uploaded, with a wide variety of hair styles and colours available for users to choose.

Also, it should provide a clear step-by-step guide to select such customisation options, so that it will be able to provide the best results based on user inputs.

Machine Learning libraries used in the app should provide accurate results which reflect the standards expected by the clients.

## 2.2 General Constraints

Because of the developers’ lack of familiarity with 3D imagery solutions, there would not be enough time to implement such feature to enable users to preview their changes through all angles.

Also, amongst the main choices of mobile frameworks and tools to develop the app – which are Flutter, React Native and Xamarin - it was found that Flutter works best for this project and for the developers. The main reasons for this choice are:

* Flutter provides by far the easiest development environment to set up for mobile apps among the three
* Reliance on third party packages is minimal with Flutter
* Unlike Xamarin, there is no architectural complexity overhead when developing cross-platform apps with Flutter
* Unlike React Native, first party packages provided by Google for Flutter are, for the most part, well documented, stable and fulfill most needs of the developers to implement the features of the app out of the box

## 2.3 Assumptions and Dependencies

It is assumed that there will be no significant interruptions for the schedule of the project plan, since no additional staff can be hired to complete the work.

Additionally, developers are expected to have access to all the software needed to work on their tasks at TAFE but are encouraged to bring their laptops in case the software that they need are not available on the classroom computers.

It is also assumed that there will no big changes in the project scope over the development lifecycle considering the limited timeframe available and the complexity of certain parts of the app.

Face Shape recognition dependencies:

* <https://github.com/VinitaSilaparasetty/face_shape_analyzer>
* <https://github.com/rajendra7406-zz/FaceShape>
* <https://github.com/hussein073/Hair_Style_Recommendation>

# 3. Specific Requirements

# 3.1 External Interface Requirements

### 3.1.1 User Interfaces

As described on section 3.5.1 and presented to the clients, the user interfaces need to be user friendly, accessible and self-explanatory.

### 3.1.2 Hardware Interfaces

Since the code/dependencies of this project are going to be hosted in the cloud, there will not be any need for the developers to manage the hardware themselves in a production environment.

### 3.1.3 Software Interfaces

The software required for development and deployment of this project is described on section 1.2.

### 3.1.4 Communications Interfaces

E-mail will be the main channel of communication between the clients and the development team. Slack is going to be used as the communication channel among developers.

## 3.2 Functional Requirements

### 3.2.1 User authentication / authorisation

3.2.1.1 Introduction

This feature enables users to sign in or sign up to the app

3.2.1.2 Inputs

User email or username and password are expected as inputs in the sign in and sign up forms

3.2.1.3 Processing

Requests sent by users are validated in the back end and if their details are correct, access to the app is granted.

3.2.1.4 Outputs

If inputs are correct, a secure cookie or token are sent back to users.

3.2.1.5 Error Handling

If any of the inputs is incorrect, an error message is displayed informing users which field is incorrect. If a user with the specified email or username already exists, a different error message is shown alerting users that they should choose another email/username.

### 3.2.2 History of pictures and selected face shapes / skin tones / hair styles

3.2.2.1 Introduction

Users will be able to visualise history of changes to pictures that they have submitted.

3.2.2.2 Inputs

Pictures uploaded by users

3.2.2.3 Processing

Changes to pictures are stored in a table in a database.

3.2.2.4 Outputs

Pictures uploaded and their corresponding history of changes

3.2.2.5 Error Handling

If any of the pictures is corrupted or was not properly uploaded (e.g. due to network errors), a message is shown stating the picture could not be loaded.

### 3.2.3 Pictures of pre-set models showcasing each feature (hair colour, hair style, etc.)

3.2.3.1 Introduction

Before users upload their picture, a random sample image is displayed as example

3.2.3.2 Inputs

No inputs are required for this feature. The sample images will be pre-processed and automatically displayed before users submit their own pictures.

3.2.3.3 Processing

After users upload their picture, the original sample will be replaced.

3.2.3.4 Outputs

Sample images before users upload their pictures.

3.2.3.5 Error Handling

If sample images cannot be properly processed by the server, an error message will be displayed.

### 3.2.4 Diagrams or galleries presenting the options available for each feature

3.2.4.1 Introduction

After users upload their pictures, the next few steps would involve choosing features such as hair colour or hair style. For each one of these features, a gallery should display the options available.

3.2.4.2 Inputs

Options selected by users for features such as hair colour or hair style.

3.2.4.3 Processing

After users confirm their choices, their images will be processed by the server with the selected features.

3.2.4.4 Outputs

A new image containing the selected features.

3.2.4.5 Error Handling

If sample images cannot be properly processed by the server, an error message will be displayed.

### 3.2.5 Colour picker

3.2.5.1 Introduction

Users should be able to select hair colour through a colour picker, which should contain a colour swatch with shades as specified by the clients and a slider with 10 levels to choose the lightness of such colours.

3.2.5.2 Inputs

Users’ selection for colour shade and lightness.

3.2.5.3 Processing

Based on the values selected, a colour is processed and displayed (along with its hex/hsla code).

3.2.5.4 Outputs

A preview frame of the colour chosen along with its hex/hsla code.

3.2.5.5 Error Handling

No errors are expected for this feature. If there are no inputs, a random colour is selected.

### 3.2.6 Image preview based on users’ input

3.2.6.1 Introduction

After users upload their pictures and choose their features (see 3.2.4), a new picture with the selected changes should be displayed on the home page.

3.2.6.2 Inputs

Users’ selection for hair colours, hair styles and face shapes.

3.2.6.3 Processing

Images are processed in the backend through Python ML libraries and the results are sent back to users.

3.2.6.4 Outputs

Processed images based on the selected features

3.2.6.5 Error Handling

An error message will be displayed in case the image cannot be processed by the server for any reason or if users’ input cannot be sent to the server e.g. due to network issues.

### 3.2.7 Image upload

3.2.7.1 Introduction

Users should be able to upload their pictures either from their phones’ gallery or take a new picture. They should then crop their pictures to fit a square frame so that they can properly processed by Python ML libraries running in the backend.

3.2.7.2 Inputs

Pictures uploaded by users.

3.2.7.3 Processing

Images uploaded should be recorded in a database (linking them to the respective user) and saved to a S3 bucket.

3.2.7.4 Outputs

Images uploaded by users are cropped and sent back to the app.

3.2.7.5 Error Handling

Validation errors involve image file size (e.g. should not be bigger than 2 MB) and dimensions (e.g. should have equal width and height). If the picture cannot be uploaded at all (e.g. due to network errors), a different error message is displayed.

### 3.2.8. Additional hair features as reference

3.2.8.1 Introduction

As described by the clients, a consultation section for hair porosity, density and texture should be available in the app. It will display pictures as references to compare the different levels of each characteristic.

3.2.8.2 Inputs

No user input is required.

3.2.8.3 Processing

Reference images will be loaded from the S3 bucket used for the project.

3.2.8.4 Outputs

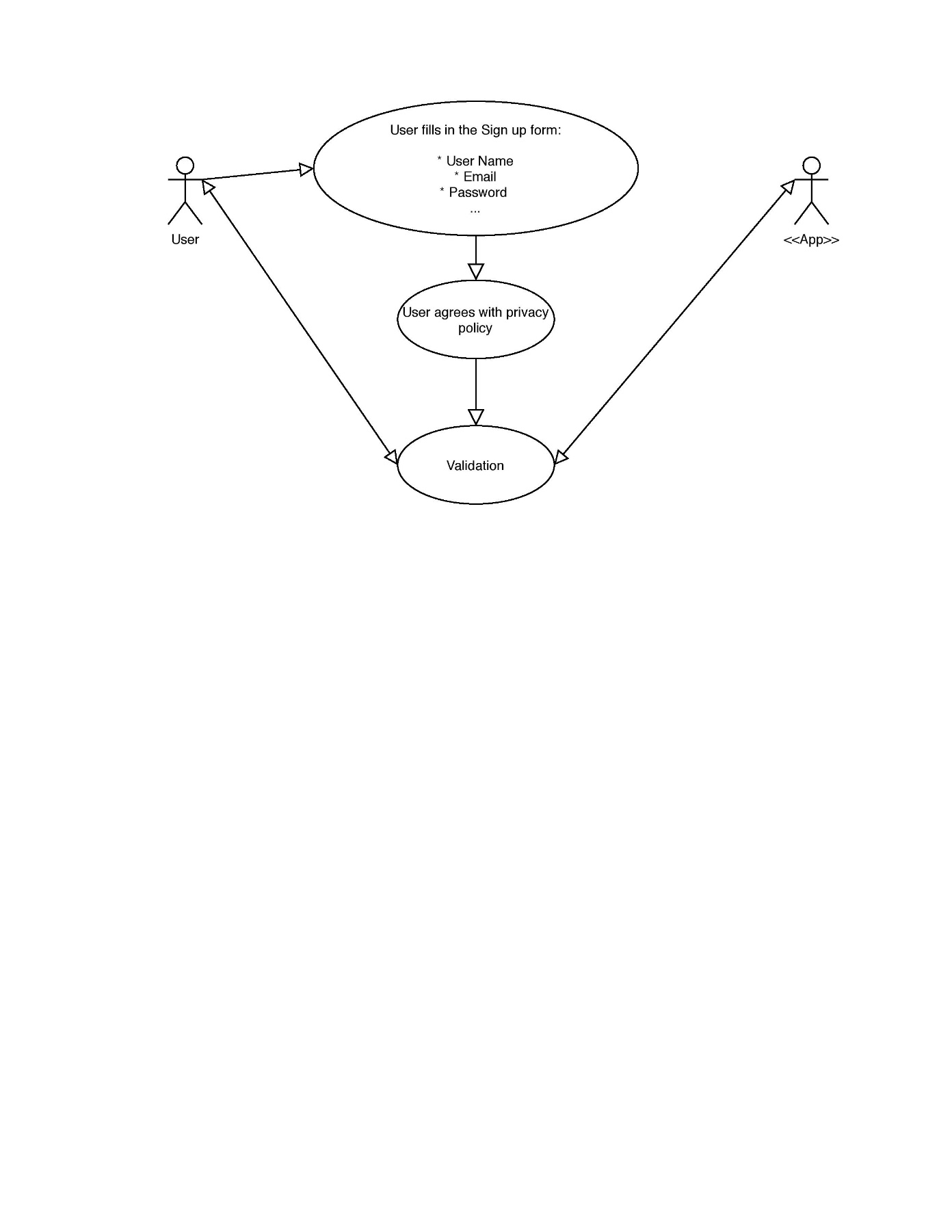
Reference images to be displayed, including degrees of porosity, density and texture.

3.2.8.5 Error Handling

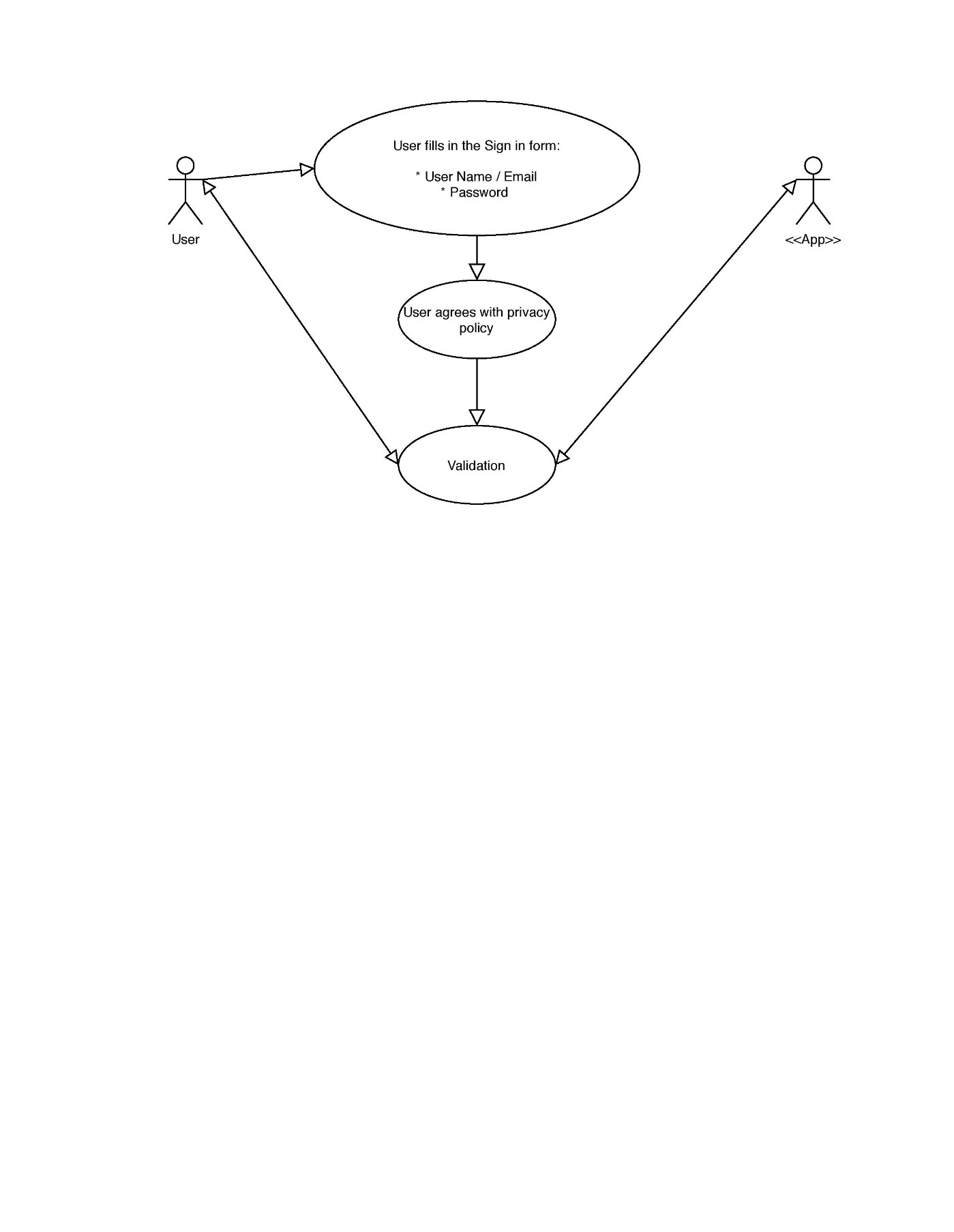
If some of the images cannot be loaded, an error message will be displayed.

## 3.3 Use Cases

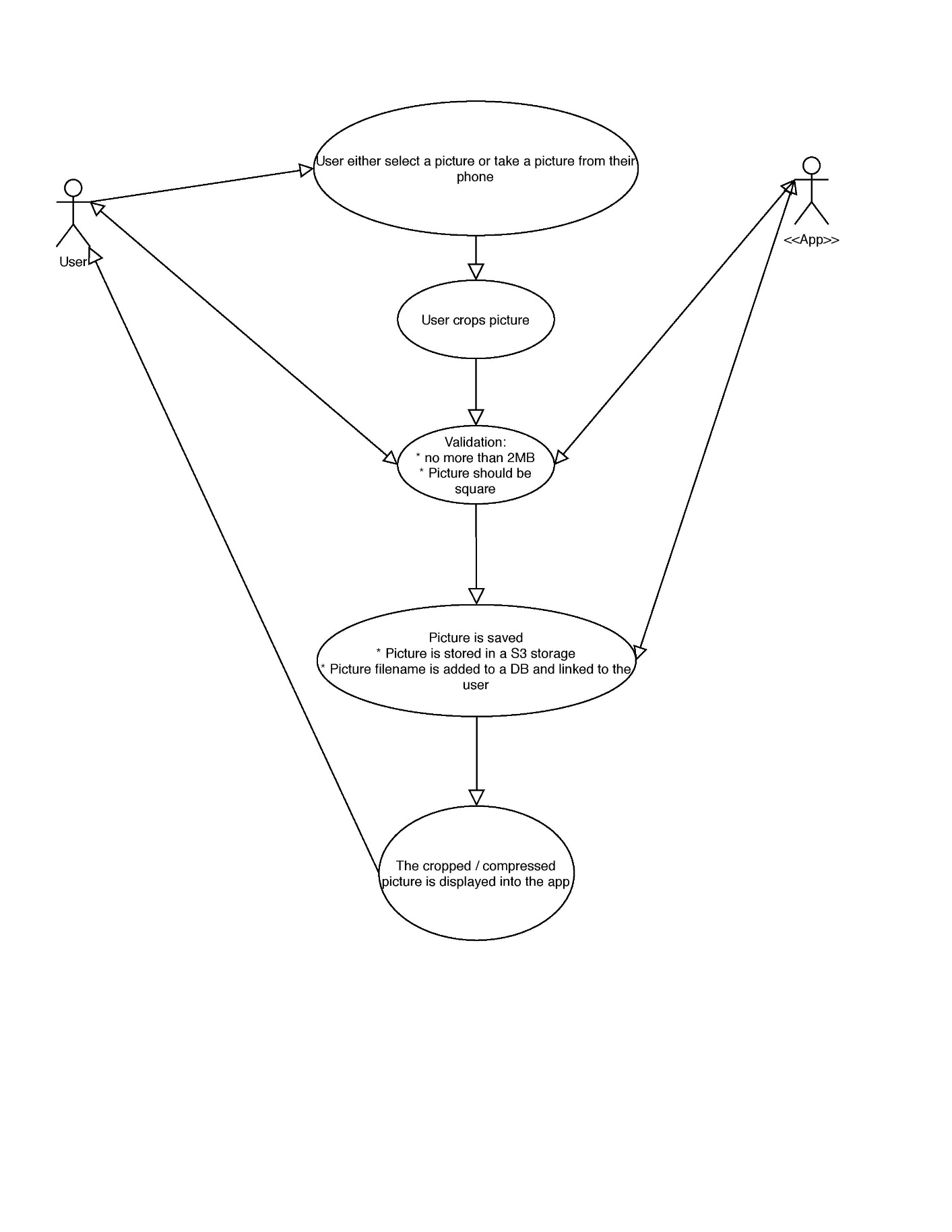
### 3.3.1 Sign up



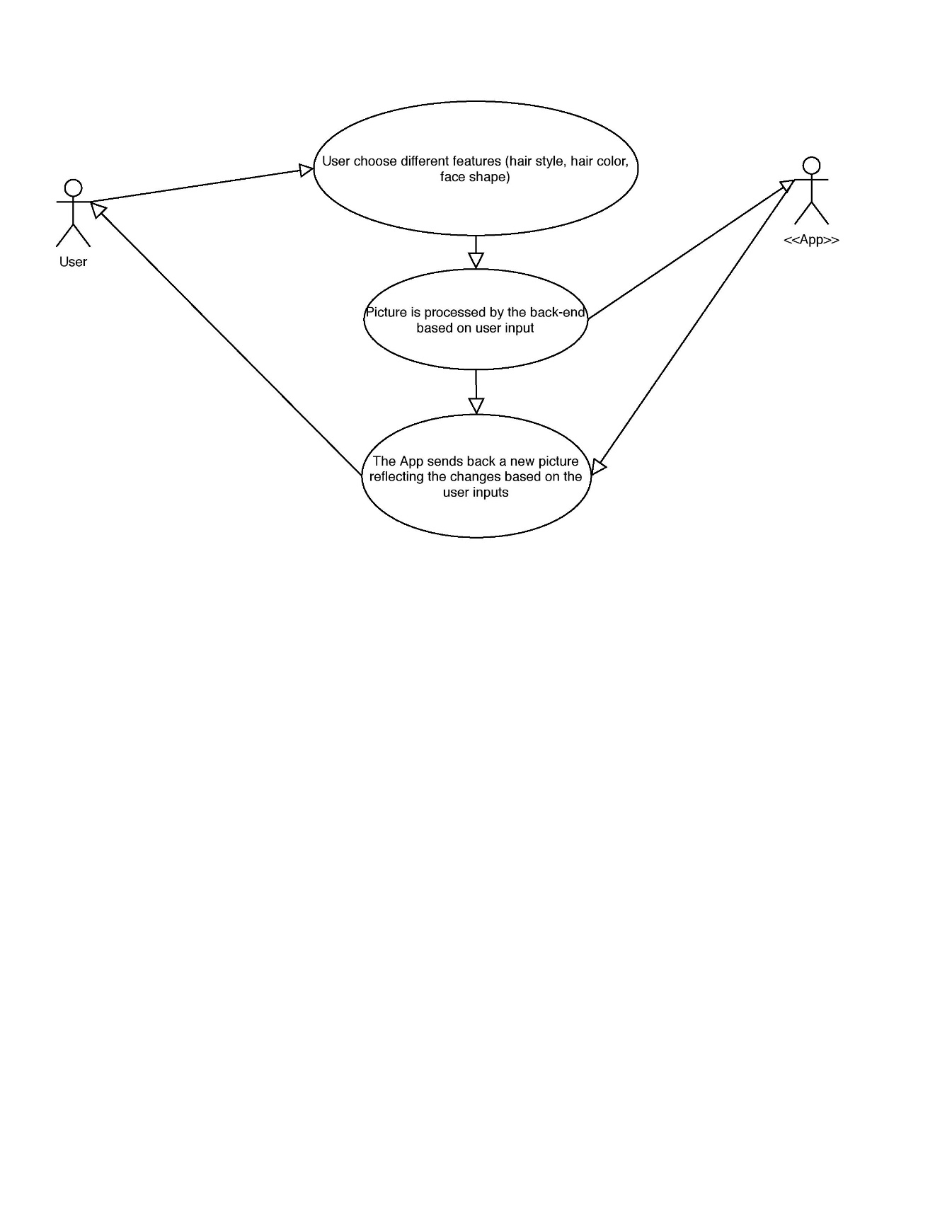
### 3.3.2 Sign in



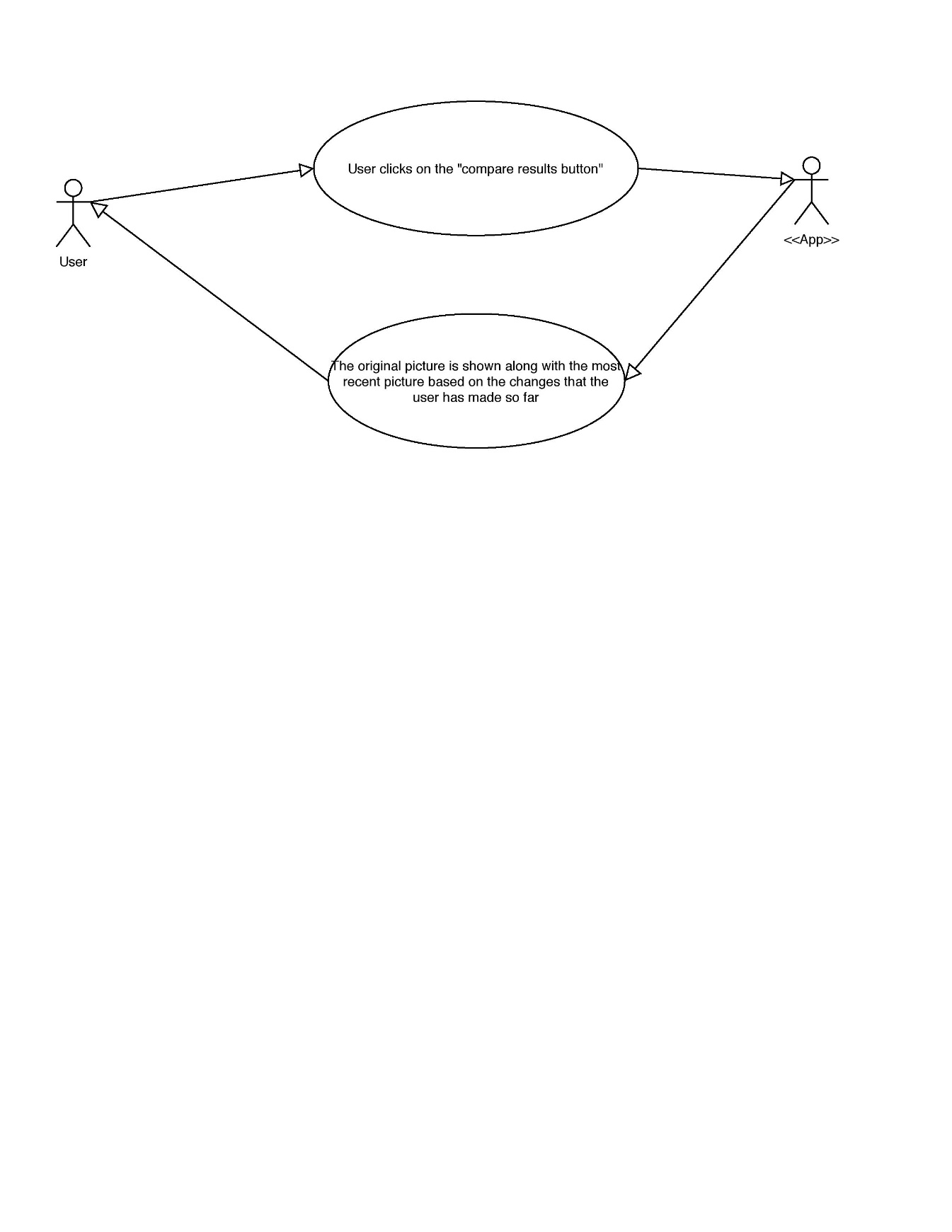
### 3.3.3 Upload pictures



### 3.3.4 Select a feature (e.g. hair colour, hair style, etc.)



### 3.3.4 Comparing results to original picture



## 3.4 Classes / Objects

Refer to the [database diagram](https://dbdiagram.io/d/5e82af534495b02c3b890292) to understand how the classes are modelled. Additional classes to be implemented:

### 3.4.1 Picture

Class that stores image data, such as filename, resolution, size and a link to the corresponding user.

### 3.4.1.1 Attributes

* Filename
* Resolution
* Size
* User ID

### 3.4.1.2 Methods

Getters and setters.

### 3.4.2 History

This class stores the history of changes (linking new pictures to the original ones) made to each picture that has been uploaded by users.

### 3.4.2.1 Attributes

* Picture ID
* Original picture ID
* User ID
* Hair colour
* Hair style
* Face shape

### 3.4.2.2 Methods

Getters and setters.

### 3.4.3 Image service

This class should contain the bulk of the business logic in the backend. It should have methods to process images based on user input, e.g. hair colour or hair style.

### 3.4.3.1 Attributes

* Picture

### 3.4.3.2 Methods

* Detect Face Shape (Picture) => a string with the detected face shape
* Change hair style (Picture, Hair style selection) => New picture with changes

Based on the hair style selected by users, this method should modify the original picture and return a new picture that displays the changes.

* Change hair colour (Picture, Hair colour selection) => New picture with changes

Similar to the previous method, this one should return a picture with a new hair colour.

* Change face shape (Picture, Face shape selection) => New picture with changes

This method should return a picture with a modified face shape based on user input.

## 3.5 Non-Functional Requirements

### 3.5.1 Usability

The app should be quite user-friendly, considering that the target audience mainly comprises non-technical users. For this purpose, it should feature a modern, clean and familiar UI and step-by-step guides that point users to the right direction.

### 3.5.2 Interactivity

Extending the user-friendliness of the app, it should not make users wait for images to be processed before they can make any further actions. In other words, every user input should be processed asynchronously and should not interrupt the flow of the actions in the app.

### 3.5.3 Responsiveness

The app layout should be displayed correctly on most common mobile screen sizes, without any issues with scale or size.

### 3.5.4 Extendibility

In case new features need to be introduced throughout or after the development lifecycle, there should not be any issues with portability or related to the development environment of the app.

### 3.5.5 Maintainability

The code base of the app should be sane and maintainable, following coding standards and guidelines (e.g. naming conventions) along with a descriptive documentation with examples.

### 3.5.6 Security

Since authentication and authorisation are going to be implemented in the app, there should not be any obvious security breaches that could leak users’ details or pictures that they have uploaded.

### 3.5.7 Performance

There should be measures in place to make the app fairly performant, considering that images must be processed by ML libraries and sent back and forth between the app and the backend, so that the time taken to process users’ actions should be minimal.

## 3.6 Inverse Requirements

## 3.7 Design Constraints

Since no one in the development team owns a Mac, the app will NOT be developed for iOS – only for Android. Despite that, the code base will still be cross-platform (since Flutter will be used), so it will still be possible to publish it to iOS in the future.

Also, the overall look and feel of the app should follow North Metropolitan TAFE’s brand guidelines (available [here](https://github.com/HairdressingProject/styleme/blob/master/Documentation/NMT%20Brand%20Style%20Guide%20May%202020.pdf)).

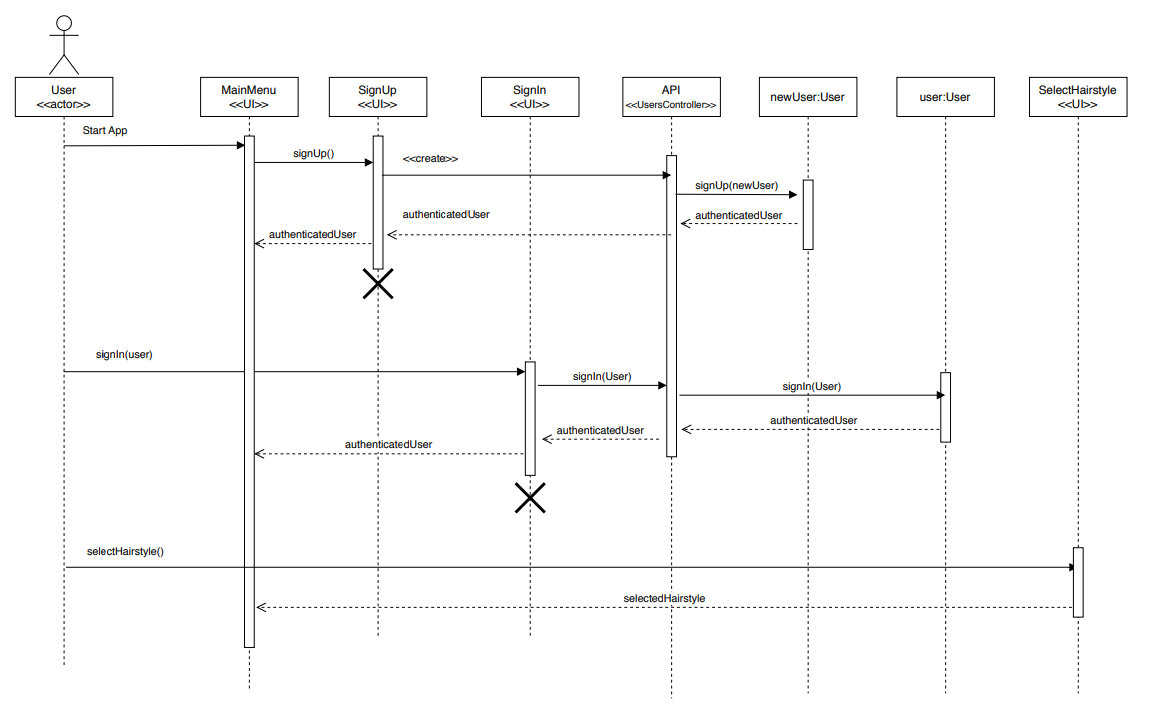
For (lack of) budget reasons, costs should be kept to a minimum to develop and deploy the app. This should not present a big hurdle to the project, since scalability is not a concern for now.

## 3.8 Logical Database Requirements

## 3.9 Other Requirements

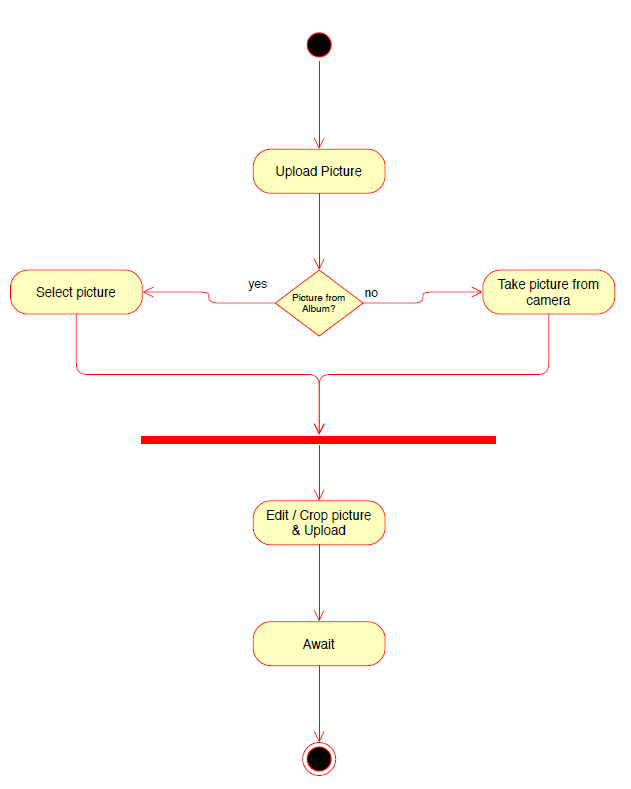
# 4. Analysis Models

## 4.1 Sequence Diagrams



## 4.3 Data Flow Diagrams (DFD)

## 4.2 State-Transition Diagrams (STD)



# 5. Change Management Process

Different changes can occur during the designing and implementation phases, either because the client request to modify a feature, the project scope changes, bugs are found on code, or developers decide to use a different technology.

Any change must be evaluated, first to make sure that changes remain under the project scope, and second to decide how it affects the whole project and how to allocate time to work on it. Developers will discuss in their regular day meetings how changes affect the project and assign tasks to complete them.

For more information, see the change management plan section available in the Project Plan referenced on section 1.4.

# A. Appendices

## A.1 Appendix 1

## A.2 Appendix 2