eTryOn - Virtual try-ons of garments enabling novel human fashion interactions

Project Title:	**eTryOn - Virtual try-ons of garments enabling novel human fashion interactions**
Contract No:	951908 - eTryOn
Instrument:	Innovation Action
Thematic Priority:	H2020 ICT-55-2020
Start of project:	1 October 2020
Duration:	24 months

Deliverable No: D4.2

Initial eTryOn middleware and APIs

Due date of deliverable:	30 September 2021
Actual submission date:	7 October 2021
Version:	Final
Main Authors:	Anastasios Papazoglou Chalikias Jim Downing Ray Miller

Project funded by the European Community under the H2020 Programme for Research and Innovation.

Deliverable title	Initial eTryOn middleware and APIs
Deliverable number	D4.2
Deliverable version	Final
Contractual date of delivery	30 September 2021
Actual date of delivery	7 October 2021
Deliverable filename	eTryOn_D4.2
Type of deliverable	Report
Dissemination level	PU
Number of pages	144
Workpackage	WP4
Task(s)	T4.2
Partner responsible	Metail
Author(s)	Jim Downing (Metail), Ray Miller (Metail), Anastasios Papazoglou Chalikias (CERTH), Thomas De Wilde (QuantaCorp), Jamie Sutherland (Mallzee)
Editor	Elisavet Chatzilari (CERTH)
Reviewer(s)	Anastasios Papazoglou Chalikias (CERTH) Thomas De Wilde (QuantaCorp)

Abstract	Documentation and API references for initial software components on eTryOn
Keywords	Architecture, integration, API

Copyright

Deliverable history

Version	Date	Reason	Revised by
1.0	2021-09-14	Table of contents	Jim Downing
1.1	2021-09-29	Content	Jim Downing , Ray Miller , Anastasios Papazoglou Chalikias , Thomas De Wilde , Jamie Sutherland
1.2	2021-10-01	Snapshots of techdocs site. Final revision	Jim Downing
2	2021-10-04	Final version	Elisavet Chatzilari

List of abbreviations and Acronyms

Abbreviation	Meaning
KPI	Key Performance Indicator
DevOps	A combination of Software **Dev**elopment and IT **Op**eration**s**. DevOps is a set of practices that combines software development and IT operations. It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.
DSL	Domain Specific Language
API	Application Programming Interface. In this document it exclusively means a network interface.
AR	Augmented Reality - the composition of 3D objects into a scene viewed through a camera and display that adds virtual objects to the scene.
HTTP (S)	HyperText Transfer Protocol (Secure). HTTP is the protocol used to transfer data over the web. It is part of the Internet protocol suite and defines commands and services used for transmitting web data.
NoSQL	Database management systems that do not use the relational model or offer querying in Structured Query Language (SQL). These often offer high throughput and easier updating and reading.
VR	Virtual Reality
ADR	Architecture Decision Record
SDK	Software Development Kit. In this document this refers to a component that can be incorporated into a larger software container and runs in-process with it.
CLI	Command Line Interface. A program interface operated by issuing text commands through a terminal.
AWS	Amazon Web Services. A collection of Infrastructure As A Service products.
JWT	JSON Web Tokens. A compact URL-safe means of representing claims to be transferred between two parties
JSON	JavaScript Object Notation. A standard method of serialising data, particularly when produced by or for consumption by JavaScript programs.
IAM	Identity and Access Management is a framework of policies and technologies that facilitates the management of electronic or digital identities and access to resources.
CI/CD	Continuous Integration / Continuous Deployment. Build and application deployment automation that is configured to run automatically when certain events occur in source control, either when any code is committed, or e.g. when a version is tagged.
CSV	Comma Separated Values. A sort-of specified data encoding.

GCP	Google Cloud Platform. A collection of Infrastructure As A Service products.
UI / Ux	User Interface / User eXperience. Aspects of the application that are the end user interacts with. Ux tends to put more emphasis on the dynamic aspects, and UI more on the purely visible static elements.
Pub/Sub	Publish / Subscribe. A form of messaging architecture in which computer processes can publish messages (small chunks of data) to one of a number of “topics”, and other processes can receive these messages by “subscribing” to the topic.
REST	REpresentational State Transfer. A style of web programming interface that is designed to work in particular sympathy with HTTP and the rest of the web architecture.

Table of Contents

1 Executive summary

The development of architecture in eTryOn is an iterative, ongoing process that aims to promote communication and reduce technical risk on the project. We particularly focus on timely decisions of technology choices and approaches, and on clarifying interfaces between components (especially those between consortium partners).

Most of the contents of this document are continuously updated and maintained through the project technical documentation website at https://etryon.gitlab.io/techdocs/. To make this document self-contained, we have appended snapshots of relevant parts of that site in appendix, but the live site is canonical.

The following sections enumerate each software component in the eTryOn systems, describing the location of running component examples, source code and API documentation. To understand how these fit together, please refer to the architecture diagrams on that site: https://etryon.gitlab.io/techdocs/arch-diags/

2 Authentication and Authorization

We make use of Google Cloud Platform's Firebase Auth service for authentication and authorization in all three use cases, as described in the following ADRs published on our techdocs site, and appended to this document for convenience: -

- ADR-0006 Use an Application Development Platform https://etryon.gitlab.io/techdocs/adr/0006-use-an-app-dev-platform/ (See 7.2 ADR-0006 Use an Application Development Platform)
- ADR-0016 Authentication and Authorization https://etryon.gitlab.io/techdocs/adr/0016-authentication-and-authorization/ (See 7.12 ADR-0016 Authentication and Authorization)
- ADR-0023 Use Custom Claims https://etryon.gitlab.io/techdocs/adr/0023-use-custom-claims/ (See 7.16 ADR-0023 Use Custom Claims)

3 Secret Manager

Secret manager is a Google Cloud Service for storing API keys, passwords, certificates, and other sensitive data. We will use the secret manager to store AWS keys for calling Metail services, and private keys for signing JWT claims. These secrets will be read by cloud functions, and access controlled through the Google Cloud IAM system.

Provisioning of the secret manager for each use case will be via terraform scripts run by GitLab CI/CD.

4 VR Designer System

4.1 User roles

The user roles in the VR designer app are:

Designer: a garment designer who uses Browzwear's VStitcher to develop garment designs.

Product Manager: responsible for selecting designs and guiding their improvement and development to optimise market success.

Users in both roles work for the same organisation, so the differentiation between them is considered for usage scenarios, but there are no permissions restrictions for each of the roles.

4.2 Components

4.2.1 Body Model Service & React client SDK

The body model service is described in OpenAPI format at https://etryon.gitlab.io/techdocs/apis/quantacorp/qc-api/ (Snapshot: 8.3 QuantaCorp API).

The specifications for the React client SDK are still a work in progress.

4.2.2 VR Data Store

We use Google Firebase to host the VR Data Store, in a cloud Firestore Database. It consists of 6 collections as described below:

- **Animations:** Features user saved animations, uploaded from the VR Designer Config app.
- **Avatars:** Features user saved avatars, uploaded from the VR Designer Config app.
- **Garments:** Features user uploaded garments from the VR Designer Config app.
- **Market Segments:** User created Market Segment configurations to be used when uploading a new garment entry, using the VR Designer Config app.
- **Market Segment Results:** Featuring results in the form of a score from Mallzee for a garment entry under specific market segments.
- **User Information:** All user and preference data are saved here.

The Firestore Database schemas describing in depth the VR Data Store are located in https://etryon.gitlab.io/techdocs/json-schemas/use-case-1/ and are included in ADR-0021 Storage Path Scheme.

4.2.3 VStitcher headless

Closed source

This component won't be delivered in the initial system as VStitcher is not yet able to create alembic animations in headless mode. Once it does, the component will be introduced with an execution script as follows:

Input	● Reference to avatar to use for garment (presigned URL to .fbx file) ● Name of animation to use (String) ● Reference to garment file (presigned URL to .bw file) ● Reference to write animation to (presigned URL)
Operation	1. Download .bw file. 2. Download avatar file. Start VStitcher 3. Open .bw file 4. Add avatar to .bw file. 5. Arrange garment on avatar 6. Dress avatar 7. Select animation from avatar file 8. Dress (simulate) whilst running animation 9. Export simulation results as ABC (Alembic file) and save locally 10. Upload animation to results location (presigned URL)
Output	● Error code / success code. ● Results are written to presigned URL as a side-effect.

4.2.4 Identity Service

The Identify Service is not a deployable component, but rather a set of policies on how we use Firebase Auth. Further description can be found in Authentication and Authorization.

4.2.5 Secret Manager

See 3 Secret Manager.

4.2.6 Rating Retrieval Function

A Google Cloud Function that is hosted on Firebase. It takes as input market segment data for each garment configuration, sends it to a Mallzee service and receives a performance score in percentage about each garment-market segment configuration. The performance score predicts how successful this garment entry will be in the selected market segment.

Input	● Reference to a garment file. ● Market segment properties (Age Target, color, price).
Operation	1. The function sends a reference URL of a garment along with its market segment to the Consumer Rating Prediction Service. 2. It receives a result in the form of a percentage score. 3. The function then writes to the VR Data Store, an entry that features the garment ID, the market segment ID, and the performance score.
Output	● Error code / success code. ● Results are written to market_segment_results Firebase collection in VR Data Store.

In depth documentation in: https://etryon.gitlab.io/techdocs/cloud-functions/rating-retrieval/ (See 10.5 Rating Retrieval)

The function repository is located in: https://gitlab.com/etryon/uc1/gcf-rating-retrieval

4.2.7 Token Service

The Token Service is implemented as a Google Cloud Function. It's operation is specified in https://etryon.gitlab.io/techdocs/adr/0015-token-service-for-quantacorp-api/ (See ADR-0015 Token Service for QuantaCorp API).

4.2.8 VR Asset Store

We use Google Firebase Storage to store all asset files in an appropriate structure for easier identification and management purposes. The folder structure used for every file needed is described in the following ADR published on our techdocs site, and appended to this document for convenience:

- ADR-0021 Storage Path Scheme https://etryon.gitlab.io/techdocs/adr/0021-storage-path-scheme-decision/#storage-access-paths-in-use-cases (See ADR-0021 Storage Path Scheme)

4.2.9 Consumer Rating Prediction Service

The consumer rating prediction service is specified at https://etryon.gitlab.io/techdocs/apis/mallzee/1.0.0/ (Snapshot: 8.1 Mallzee eTryOn API).

4.2.10 Avatar Creation Function

Open Source.

An up-to-date specification of the function is at https://etryon.gitlab.io/techdocs/cloud-functions/avatar-creation/, and a snapshot is included as 10.1 Avatar Creation.

4.2.11 Avatar Creation Service

Closed source

An up-to-date specification of the service is at https://etryon.gitlab.io/techdocs/apis/metail/scanatar-service/, and a snapshot is included as 8.2 Metail Scanatar Service.

5 DressMeUp System

5.1 User roles

The user roles in the Dress Me Up system are

eTryOn Project Staffer Responsible for uploading garment content into the system. In a production system this would be replaced by someone at an apparel brand, but it's not clear who in a brand organisation this would be.

Influencer Any social media user who likes to share images of themselves wearing fashion.

We will require Influencers to register and authenticate with the system. There will be strong auth controls on any content they upload, and preventing influencers and unauthenticated users from accessing brand-proprietary data.

5.2 Components

5.2.1 Odlo eCommerce product feed

ODLO provides a product feed in the form of a CSV file, that is updated daily with the latest garment information. The product feed is located at a provided URL location.

5.2.2 Catalogue Updater

A Google Cloud function has been utilized to parse information from the Odlo web feed and update the garment database in the Dress Me Up Data Store. The cloud function runs once per day, reads the updated CSV file and makes the necessary changes in the Google Firestore Database of the project.

Input	• Parsed CSV file
Operation	1. The service triggers a function once a day that retrieves and parses the CSV file from the link. 2. It then uploads new entries and refreshes the status of the garments.
Output	• Error code / success code. • Results are written to garments Firebase collection in the VR Data Store.

In depth documentation in: https://etryon.gitlab.io/techdocs/cloud-functions/catalogue-update/

The function repository is located at https://gitlab.com/etryon/shared/gcf-catalogue-updater.

5.2.3 DressMeUp Admin

Open source CLI. See Usage Scenarios and Source Code Repository

```
NAME:
   DressMeUp Admin CLI - Command-line tool for DressMeUp administrators
```

```
USAGE:
   dress-me-up-admin command [command options] [arguments...]

COMMANDS:
   init                        Initialize the CLI
   add-catalogue-garment       Add Browzwear file for a product that is
in
                               the catalogue
   add-non-catalogue-garment   Add Browzwear file and product details for
                               a product that is not in the catalogue
   help, h                     Shows a list of commands or help for one
                               command

GLOBAL OPTIONS:
   --help, -h  show help (default: false)
```

5.2.4 DressMeUp Asset Store

We use Google Firebase Storage to store all asset files in an appropriate structure for easier identification and management purposes. The folder structure used for every file needed is described in the following ADR published on our techdocs site, and appended to this document for convenience:

- ADR-0021 Storage Path Scheme - https://etryon.gitlab.io/techdocs/adr/0021-storage-path-scheme-decision/#storage-access-paths-in-use-cases

5.2.5 DressMeUp Data Store

We use Google Firebase to host the DressMeUp Data Store, in a cloud Firestore Database. It consists of 3 collections as described below:

- **Collection Items:** Features the created collection items of each user, that consists of all the information needed for the synthesized media entries.
- **Garment Suggestions:** Features the suggested garments list for each user, that is created by Mallzee using as input past garment interactions.
- **Garments:** A collection that provides all the information needed about available garments from ODLO that are used in the DressMeUp application.
- **User Information:** Features all user information that is needed for the application to work, like gender, size and generated avatar.

The Firestore Database schemas describing in depth the DressMeUp Data Store are located in https://etryon.gitlab.io/techdocs/json-schemas/use-case-2/ and are included in 11.3 UC2 Schemas.

5.2.6 Avatar Creation Function

Open source

An up-to-date specification of the function is at https://etryon.gitlab.io/techdocs/cloud-functions/avatar-creation/, and a snapshot is included as 10.1 Avatar Creation.

5.2.7 Avatar Creation

An up-to-date specification of the service is at https://etryon.gitlab.io/techdocs/apis/metail/scanatar-service/, and a snapshot is included as 8.2 Metail Scanatar Service.

5.2.8 DressMeUp (mobile) web app

The Dress Me Up application is a React.JS browser app with an optimized user interface to run on mobile devices emulating a mobile application experience. A first version of the application is deployed in: https://etryon-h2020.web.app

The usage scenarios are described in: https://etryon.gitlab.io/techdocs/usage-scenarios/2-dressmeup/

The application source code repository is located in: https://gitlab.com/etryon/dress-me-up

5.2.9 DressMeUp mobile app - Body Model SDK, Body Model Service

See 4.2.1 Body Model Service & React client SDK .

5.2.10 Consumer Ratings Function

A Google Cloud Function that handles the call to the Consumer Rating Prediction Service, by providing the necessary information and then writing back the results to the Dress Me Up Data Store.

Input	● An array of garment IDs that correspond to the ones that the user has selected to use in collection items.
Operation	1. The service creates an array of garment IDs based on a user's active collection. 2. The service sends the array to the Prediction service and gets back a list of garment IDs for a specific user. 3. The returned data is saved into the Dress Me Up Data Store.
Output	● Error code / success code. ● Results are written to garment_suggestions Data Store collection.

In depth documentation in: https://etryon.gitlab.io/techdocs/cloud-functions/consumer-ratings/ (Snapshot: 10.3 Consumer Ratings)

The function repository is located in https://gitlab.com/etryon/use-case-2/gcf-consumer-ratings

5.2.11 DressMeUp Composition Invoker

Up-to-date documentation can be found at https://etryon.gitlab.io/techdocs/cloud-functions/dress-me-up-composition/ . A snapshot is given in 10.4 DressMeUp Composition Invoker.

5.2.12 Composition Service

Up to date documentation can be found at https://etryon.gitlab.io/techdocs/apis/metail/uc2-composition-service/ . A snapshot is given in 8.4 UC2 DressMeUp Metail Composition Service.

6 Magic Mirror System

6.1 User Roles

The main roles in the Magic Mirror system are that of

3D Technical designer: Responsible for uploading 3D garment data into the system

User An end consumer who browses the magic mirror system, selects styles and tries them on.

As in the DressMeUp system, there will be strong security constraints around each user's data, particularly because there is PII under long-term storage. There also need to be careful constraints around the proprietary data uploaded by 3D technical designer, before it is transformed into shareable 3D assets by the system.

6.2 Components

6.2.1 Magic Mirror mobile app

Open Source

One of the more important API aspects of this app is how analytics events will be sent to the server, and how they can then be made available to Mallzee systems in order for them to train their preference prediction models. This is covered in https://etryon.gitlab.io/techdocs/adr/0024-uc3-analytics-events-to-server-eventbus-decision/ (Snapshot: 7.13 ADR-0024 Unity Analytics Events to Server Event Bus).

Google Firebase Analytics are used to track usage behavior and log a variety of events. This is implemented through the SDK provided by Google for using Firebase within Unity applications.

The Firebase SDK offers support for logging two primary types of information:

- *Events*: log user actions, system events, errors.
- *User properties*: attributes that describe user base, such as geographical information and language preference

In the context of the Magic Mirror mobile app, a variety of events will be, such as user clicks and user engagement time. Moreover, some user properties will be recorded, too, such as user Operating System, OS Version, geographical information, gender, and age.

The Firebase Analytics data are connected with Google Cloud Big Query[1], which makes them queryable. Google Cloud's Big Query is a serverless, highly scalable, and cost-effective multicloud data warehouse. It features a REST API which can be called to fetch our Firebase Analytics data.

We will be using the integration between Firebase Analytics and Google Cloud Big Query in order to feed Mallzee's user-specific garment suggestions function, with the events from the Magic Mirror app.

6.2.2 Magic Mirror Admin CLI

Open Source CLI. See Usage Scenarios and Source Code Repository.

[1] https://cloud.google.com/bigquery

```
NAME:
   MagicMirror Admin CLI - Command-line tool for MagicMirror
                            administrators

USAGE:
   magic-mirror-admin command [command options] [arguments...]

COMMANDS:
   init                         Initialize the CLI
   add-catalogue-garment        Add FBX file for a product that is in the
                                catalogue
   add-non-catalogue-garment    Add FBX file and product details for a
                                product that is not in the catalogue
   help, h                      Shows a list of commands or help for one
                                command

GLOBAL OPTIONS:
   --help, -h  show help (default: false)
```

6.2.3 Magic Mirror Data Store

We use Google Firebase to host the Magic Mirror Data Store, in a cloud Firestore Database. It consists of 2 collections as described below:

- **Garments:** A collection that provides all the information needed about available garments from ODLO that are used in the Magic Mirror application.
- **User Information:** Features all user information and app configuration that is needed for the application to work.

The Firestore Database schemas describing in depth the Magic Mirror Data Store are located in https://etryon.gitlab.io/techdocs/json-schemas/use-case-3/ and are included in 11.4 UC3 Schemas.

6.2.4 Magic Mirror Asset Store

We use Google Firebase Storage to store all asset files in an appropriate structure for easier identification and management purposes. The folder structure used for every file needed is described in the following ADR published on our techdocs site, and appended to this document for convenience:

- ADR-0021 Storage Path Scheme https://etryon.gitlab.io/techdocs/adr/0021-storage-path-scheme-decision/#storage-access-paths-in-use-cases (Snapshot: 7.11 ADR-0021 Storage Path Scheme)

6.2.5 Token Service

The Token Service is implemented as a Google Cloud Function. It’s operation is specified in (https://etryon.gitlab.io/techdocs/adr/0015-token-service-for-quantacorp-api/ Snapshot: 7.12 ADR-0023 Use Custom Claims).

6.2.6 Unity model creation fn

An up-to-date specification of the function is at https://etryon.gitlab.io/techdocs/cloud-functions/uc3-unity-model-creation/ (Snapshot: 10.6 UC3 Unity Model Creation).

6.2.7 Garment model creation service

An up-to-date specification of the function is at https://etryon.gitlab.io/techdocs/apis/metail/uc3-garment-model-creation/ (Snapshot: 8.5 UC3 Garment Model Creation Service).

6.2.8 Body Model Service / Magic Mirror mobile app - body model SDK

The API services used will be identical in specification and function as that described in 4.2.1 Body Model Service & React client SDK .

The SDK specifications are described at https://etryon.gitlab.io/techdocs/sdks/qc-obj-c-sdk/qc-obj-c-sdk/ (Snapshot: 13.1 QuantaCorp Objective-C SDK).

6.2.9 Identity Service

The Identify Service is not a deployable component, but rather a set of policies on how we use Firebase Auth. Further description can be found in Authentication and Authorization.

6.2.10 Secret Manager

See 3 Secret Manager.

6.2.11 Catalogue Update Function

A Google Cloud function has been utilized to parse information from the Odlo web feed and update the garment database in the Dress Me Up Data Store. The cloud function runs once per day, reads the updated CSV file and makes the necessary changes in the Google Firestore Database of the project.

Input	● Parsed CSV file
Operation	3. The service triggers a function once a day that retrieves and parses the CSV file from the link. 4. It then uploads new entries and refreshes the status of the garments.
Output	● Error code / success code. ● Results are written to garments Firebase collection in the Data Store.

In depth documentation in: https://etryon.gitlab.io/techdocs/cloud-functions/catalogue-update/ (Snapshot: 10.2 Catalogue Update).

The function repository is located at https://gitlab.com/etryon/shared/gcf-catalogue-updater.

7 Appendix A - Relevant ADRs from https://etryon.gitlab.io/techdocs/

7.1 ADR-0005 Use APIs

Date	2021-05-11
Status	Accepted

7.1.1 Context

Some functionality for the eTryOn project will be provided by commercial companies in the consortium (QuantaCorp, Mallzee, and Metail). These companies will not in general want to make details of their implementations public, but instead present a “black box” interface.

7.1.2 Decision

We will access services from partner companies through HTTP-based APIs.

7.1.3 Consequences

We need to ensure these third-party APIs are documented so they can be consumed by eTryOn applications.

Partner companies do not need to disclose implementation details.

The systems behind these APIs can be deployed and updated independently of eTryOn project resources, so long as API-compatibility is maintained.

Control of these services remains in the hands of each partner company. Note that this implies some risk to the project as, by definition, these services are not under the control of the project.

7.2 ADR-0006 Use an Application Development Platform

Date	2021-02-09
Status	Accepted

7.2.1 Context

Mobile and web application development are increasingly achieved by using high level services provided by cloud service providers like Google Cloud Platform (GCP) or Amazon Web Services (AWS). This has led to the development of SDKs and tooling that make it easier to compose these services into working applications through simple provisioning and client-side support libraries. Essentially they provide a path of least resistance for many concerns in application development for functions from object storage to application deployment, analytics to authentication.

The two application development platforms we considered are AWS Amplify and Google Firebase. Both of these make application development easier and are backed by rich service offerings in their respective cloud platforms.

Unity game engine will be the main development platform for the eTryOn project so we created two minimal Unity applications using the aforementioned toolkits to compare the two solutions in terms of ease of use, support, and capabilities. The demo applications use simple operations, namely an authentication process and communication with cloud storage for upload and download operations.

- Firebase was easy to set up, and has an up-to-date supported SDK for Unity.
- The aws.net SDK has support for Unity but requires a tedious and error-prone set-up process.
- The Firebase platform is easier to use with a much shorter learning curve when it comes to typical backend services such as authentication, analytics, cloud storage and databases.
- The AWS SDK offers more functionality, which at the moment do not seem to be of any particular use for our case.
- There are more resources (such as documentation and community guides) available for the Firebase than for Amplify.

Note: Amazon's Unity mobile SDK has not been mentioned so far since it has not been updated for 5 years and is now considered deprecated. The suggested Unity SDK from Amazon is the aws.net SDK considered above.

7.2.2 Decision

We will use Firebase for the eTryOn project.

7.2.3 Consequences

The documentation and community resources for Firebase will allow developers to get up and running quickly.

We will be able to use up-to-date and supported SDKs to consume cloud services in our Unity and Javascript applications.

The tooling and documentation available for Firebase will free up developers to focus on core facets of the applications.

Use of Firebase implies that we will use GCP-backed services for authentication (Firebase Auth), object storage (Cloud Storage), and NoSQL database (Cloud Firestore).

Consuming cloud services directly (via the provided SDK) means we do not need to write dedicated APIs for simple things like object and metadata storage.

7.3 ADR-0010 QuantaCorp SDK

Date	2021-03-09
Status	Accepted

7.3.1 Context

The eTryOn project will be developing multiple applications that require an avatar to work or to enrich the user experience. The creation of this avatar starts with the capture of two images. Those two images are sent as input to the QuantaCorp pipeline. After processing, the output of that pipeline is a 3D model.

QuantaCorp's current technology portfolio consists of an API, a web portal for B2B customers, and a mobile app for B2B customers on iOS and iPadOS.

Given that eTryOn is customer focused, we need a solution that will meet customer expectations while keeping the impact on QuantaCorp's architecture to a minimum.

To reach a broad public, we need to expand to other platforms like the web and Android. We should avoid having to make the user open a separate app to capture the required photos. Having to switch apps can have a negative effect on the overall user experience. Because there is no direct communication channel as there is in a B2B environment, guidance during a scan becomes very important. The user will also expect the scan interface to work in the same fashion across platforms. If this experience cannot be guaranteed to be similar, guidance during a scan becomes a must.

Developers should be provided with libraries that facilitate the consumption of the QuantaCorp API, and allow them to integrate QuantaCorp's pipeline in a loosely coupled way.

7.3.2 Decision

We will create an SDK for Android, iOS and web that will consume the QuantaCorp API and will include a scan component to facilitate photo capture.

7.3.3 Consequences

By creating an SDK for the various eTryOn apps, scanning functionality will be implemented in the apps themselves. This way we avoid having to force the user to use a separate app.

The SDK will be implemented in Unity (cf ADR-0006) which could pose a small development challenge.

We reduce the complexity of the project's architecture by avoiding the integration of yet another application, and yet another system.

7.4 ADR-0011 Mallzee Api

Date	2021-05-11
Status	Proposed

7.4.1 Context

As part of the eTryOn project it requires that clothing products should be evaluated for product market fit against target demographics and provide recommendations of other products based on a given product.

Mallzee has a datapool of consumer options again 4 million+ products and has experience in attempting to predict future product performance by using consumer data captured via the Mallzee shopping app.

Mallzee is looking to provide access to these insights so that other companies can benifit from the data and models produced. This in turn allows companies to make smarter decisions about the types of products they want to produce by using the vast consumer data available via the Mallzee apps.

7.4.2 Decision

Mallzee will produce an API that will be available to the eTryOn consortium. The API will consisit of a minimum of two endpoints to provide access to the product predictions and the product recommender.

7.4.3 Authentication

This Mallzee API is only required to be accessed via internal services. Not directly from the client applications. We will use a simple header based authentication method using the key x-api-key which will contain a string access key assigned to every consumer of the API. This allows Mallzee to track usage and identify which account is accessing the API.

The eTryOn services will store the given key in the Google Secret Manager so that access can be given to any service within the platform.

7.4.4 Product predictions

This endpoint will require that the user send product data and target market data so that the system can predict the popularty of the product and return a confidence score of how likely that product is going to prove popular with the target demographic.

7.4.5 Product recommendations

This endpoint will require that the consumer of the API sends the required product information along with the end users market preferences. This system will return an array of recommended product IDs based on the information given that will allow the consumer to fetch product information on the recommended products.

These API will be detailed in OpenAPI Spec 3.0 and can be found here (Link to be provided)

7.4.6 Consequences

By creating a general purpose API we can supply product performance predicitions to all applications that require it as part of the overall project.

7.5 ADR-0012 Metail Scanatar Creation

Date	2021-05-17
Status	Proposed

7.5.1 Context

The scanatars created by the QuantaCorp Body Model Service are post-processed by a Metail pipeline that performs clean-up of the scan, landmark detection, mesh fitting, and skeleton fitting. The output from the Metail pipeline is a Unity-compatible avatar that can be used in eTryOn applications. We would like the Metail pipeline to run every time a new QuantaCorp scanatar is uploaded to Cloud Storage.

7.5.2 Decision

Metail will implement an AWS Step Function to run the Metail pipeline.

This step function will:

- Read additional parameters (such as gender and height) from its invocation parameters;
- Read the QuantaCorp scan from Cloud Storage using a pre-signed URL;
- Run the Metail Scanatar Pipeline on this input;
- Write the finished scanatar to Cloud Storage using a pre-signed URL.

We will create an eTryOn service user in the Metail AWS IAM account and grant this user permissions to invoke the step function.

We will generate access keys for the AWS service user and store them in Google Secret Manager so they are available to services running in the eTryOn Google Cloud project.

We will implement a Google Cloud Function and configure a trigger to invoke this function whenever a QuantaCorp scanatar is written to Cloud Storage. This cloud function will:

- Create a record in Cloud Firestore to track the pending avatar creation;
- Create pre-signed Cloud Storage URLs for the input (QuantaCorp scanatar) and output (Unity-compatible avatar);
- Read the configured AWS keys from Secret Manager;
- Invoke the Metail Step Function and pass the pre-signed input/output URLs and additional metadata (such as the user's height and gender) as parameters.

We will implement another Google Cloud Function to be triggered when the finished avatar is written to Cloud Storage. This function will update the Cloud Firestore record to update the status of the new avatar.

7.5.3 Consequences

Providing Metail AWS credentials to eTryOn middleware functions enables them to call AWS APIs directly.

Using pre-signed URLs for input and output avoids us having to share credentials in the opposite direction.

Tracking scanatar creation status in Cloud Firestore allows client applications to show jobs in progress and be notified when the status changes.

We will have to ensure the pre-signed URLs have a long enough expiry for the Metail pipeline to complete and write its output back to Cloud Storage.

We will need a mechanism to signal an error if the Metail Pipeline cannot process the input QuantaCorp scanatar.

7.6 ADR-0013 Metail VStitcher Headless Service

Date	2021-05-17
Status	Proposed

7.6.1 Context

Several eTryOn use cases require background processing of Browzwear files to provide functionality (for example, to generate the Alembic animation file for use case 1). Processing of a

Browzwear file will always be triggered by a middleware function when a Browzwear file is uploaded to Cloud Storage.

7.6.2 Decision

We will follow the pattern described in ADR 12 with a Google Cloud Function in the eTryOn account invoking a Step Function in the Metail AWS account to process Browzwear files.

Metail will implement an AWS Step Function to read input from, and write output to, Cloud Storage using pre-signed URLs. The step function will invoke the desired script to run in VStitcher Headless running on an EC2 Windows instance.

7.6.3 Consequences

We will have to define the processing required by each use case and encapsulate this as scripts that can be run by VStitcher Headless.

We will need a way to identify which script to invoke for each use case (e.g. through a naming convention in Cloud Storage).

If we are unable to secure a license to run VStitcher Headless, the step function will have to initiate a manual process where jobs are completed by a human using VStitcher Desktop.

7.7 ADR-0014 Scanatar Creation

Date	2021-05-11
Status	Accepted
Amended By	ADR-0015 Token Service for QuantaCorp API
Amended By	ADR-0021 Storage Path Scheme

7.7.1 Context

In ADR-0010 QuantaCorp propose to write a cross-platform SDK (for Android, iOS and web) that will facilitate photo capture and interact with the QuantaCorp API. This document goes into more detail about the architecture supporting this SDK and the scanatar creation process.

We assume that all of the applications using the SDK will require the user to login using Firebase authentication. This gives the applications access to Cloud Storage and an ID token that can be used to interact with other APIs (including the QuantaCorp API).

We also assume that the QuantaCorp SDK will have access to client-side functionality of the Firebase SDK, as all of the client applications will use the Firebase SDK.

7.7.2 Decision

The QuantaCorp SDK will use the ID token obtained via Firebase Auth as a bearer token to authenticate with the QuantaCorp API. It is the responsibility of the QuantaCorp API to validate this token.

The application implementing the SDK will generate pre-signed URLs to store the output from the QuantaCorp system in Cloud Storage owned by eTryOn. We will use a naming scheme that stores all of the user's files under a prefix of their Firebase user ID, which simplifies the authorization rules controlling access to Cloud Storage.

The application will prompt the user to enter their height and gender and save these as metadata to Cloud Firestore.

The SDK will upload the photos required for avatar generation directly to the QuantaCorp API along with the metadata, the pre-signed URLs and the ID token.

On successful upload of metadata and photos, the QuantaCorp system will generate a thumbnail from the front view photo and a OBJ file for the scanatar.

The QuantaCorp system will use the pre-signed URLs obtained from the client to write the output thumbnail and OBJ file directly to the user's eTryOn Cloud Storage area.

Successful creation of the OBJ file in Cloud Storage will trigger a Cloud Function to hand off the OBJ file to Metail's systems for rigging and creation of the final scanatar. This Cloud Function will read the user's height and gender from Firestore to pass to the Metail API.

The Metail system will also use a pre-signed URL to fetch the input from the user's Cloud Storage and write back the rigged scanatar.

7.7.3 Consequences

- The eTryOn systems will not handle the user's photos, so we avoid unnecessary handling of personally identifying data.
- Once the thumbnail and scanatar have been written successfully to the user's eTryOn Cloud Storage, the input photographs can be deleted from QuantaCorp's systems.
- Terms and conditions for applications using the QuantaCorp SDK to generate an avatar must make explicit the retention period for the user's photos, whether they are deleted immediately on completion of avatar creation or retained; if retained, the terms and conditions must state the reason for retention.
- Using pre-signed URLs minimizes the trust between systems and removes the need to issue service credentials.
- The Cloud Function invoking the Metail API may need credentials as this is the one place we don't have access to the user's ID token.

7.8 ADR-0015 Token Service for QuantaCorp API

Date	2021-05-14
Status	Proposed
Amends	ADR-0014 Scanatar Creation

7.8.1 Context

The QuantaCorp API needs to be able to validate that calls are coming from a legitimate eTryOn user. In ADR 0014 we proposed using the ID token returned by Firebase Auth as a bearer token to authenticate to the QuantaCorp API, but as this is the same token used to authenticate to Firebase services it should not be exposed to a third party API.

7.8.2 Decision

We will implement a Google Cloud Function to generate short-lived tokens with scope limited to the QuantaCorp API.

These will be JSON Web Tokens containing the ID of the authenticated user in the `id` field and `QuantaCorp` in the `audience` field, and will be valid for 15 minutes.

We will generate a key pair for signing and validating the tokens. We will share the public key with QuantaCorp so they can verify the signature.

7.8.3 Consequences

Applications using the QuantaCorp SDK will have to retrieve a token (by calling the Cloud Function) before interacting with the QuantaCorp API. They will authenticate to the Cloud Function using the ID token retrieved from Firebase Auth.

The QuantaCorp API will be able to identify the user from the `id` field in the token claims. This claim can be validated by verifying the token signature.

7.9 ADR-0016 Authentication and Authorization

Date	2021-05-14
Status	Proposed

7.9.1 Context

In ADR 0006 a decision was made to use the Firebase platform to build the client-facing applications. This platform includes the Firebase Auth component which provides backend services, SDKs, and ready-made UI libraries for account management and authentication.

Even using a drop-in framework like this we need to understand how accounts will be created and how permissions will be granted.

We identified the following requirements for the three use cases:

UC1: VR Designer

- Account creation and permission granting will be done by eTryOn staff.
- Data configuration will be done by Brand staff.
- No creation of accounts possible through Ux.

UC2: Dress Me Up

- No brand staff accounts needed.
- eTryOn staff accounts and permissions will be created manually.
- Data configuration will be done through CLI by eTryOn staff.
- End users (influencers) need a registration Ux for account creation.
- No permissions granting / account approval step needed for end users.

UC3: Magic Mirror

- No brand staff accounts needed.
- eTryOn staff accounts and permissions created manually.
- Data configuration done through CLI by eTryOn staff.
- End users (shoppers) need a registration Ux for account creation.
- No permissions granting / account approval step needed for end users.

7.9.2 Decision

We will implement UI for end users (influencers) to sign up for an account in the Dress Me Up app.

We will implement UI for end users (shoppers) to sign up for an account in the Magic Mirror app.

We will create IAM accounts with appropriate permissions for eTryOn staff to perform administrative tasks (data configuration for Dress Me Up and Magic Mirror apps, brand user account creation for VR Designer app).

The applications will be single tenant. We will not implement group membership and management within an app.

7.9.3 Consequences

Using Firebase Auth and the supported SDKs makes it easy to implement UI components for sign-up and sign-in, and for the client applications to interact with server-side Firebase components.

We will have to define authorization rules for Cloud Storage and Cloud Firestore to control user access to resources.

When Cloud Functions are called directly from one of our apps, the auth token will be validated automatically but we may need to implement authorization checks in the function itself.

It is not clear from the Firebase Auth documentation that we will be able to prevent a malicious user from accessing Firebase backend services to self-register an account, even when we do not provide a UI for this. If it is not possible to disable those backend APIs we will have to ensure that accounts created in this way have no privileges within the system. This is an area requiring more research.

7.10 ADR-0020 Product Recommendations

Date	2021-05-25
Status	Accepted

7.10.1 Context

Mallzee is tasked with building a system for delivering product recommendations to the Magic Mirror and Dress Me Up Applications. Details on how the recommender should work were needing clarification as product recommendations can be done in multiple ways. This document details the choices made that the recommender should support for each application.

7.10.2 Decision

7.10.2.1 Magic Mirror

In the magic mirror application the recommender should be trying to select products that are more suited to the users preferences similar to how Netflix curates content for a user.

To achieve this we will look at the users profile starting with `age` and `gender` we will also take into consideration the following events that happen inside of the app.

- view product
- like product
- buy product

We will also attempt to include seasonality trends into the decision making

7.10.2.2 Dress Me Up

In dress me up the recommender should be trying to select products that fit the users profile again working with the age and gender. If the user had data from the magic mirror application we can also attempt to use preference data from the events to enhance the recommendations.

A nice to have would be to take into consideration the users body shape. The data is not currently available to perform this recommendation but the team will investigate the feasibility of it once the core tasks are done.

7.10.2.3 Cold start

Both applications have a problem with cold start recommendations. How do we know what to recommend without knowing anything about them?

An initial product selection process will be investigated to try and quickly get preference information on the user to start performing recommendations.

7.10.2.4 Event stream

It was suggested that the Unity analytics stream could be used as the source for in app events that the recommender will have to lift data from. This system needs more investigation to see if extracting what we need is simple. The backup plan is to provide an endpoint to send event data to to capture specifically for recommendations.

7.10.3 Consequences

With the basics in place the system will be able to provide customised recommendations for the users of each platform. The unknowns are in the body shape data and how that can be used to enhance the system.

7.11 ADR-0021 Storage Path Scheme

Date	2021-09-21
Status	Accepted
Amends	ADR-0014 Scanatar Creation

7.11.1 Context

We need to define the storage paths of all use cases in the asset store (Cloud Storage) and database (Cloud Firestore) so data can be managed and filtered, taking into account the following concerns:

- GDPR Compliance
- Access Control
- Event Pub/Sub

7.11.1.1 GDPR Compliance

We need to be able to identify all user-owned and PII-containing data so we can respond to subject access requests and right to be forgotten requests.

7.11.1.2 Access Control

Cloud Storage and Firestore use path-based rules for access control. For our use-cases, all data falls into one of three categories:

- public data: curated by an administrator, can be read by anyone (including unauthenticated users)
- shared data: curated by an administrator, can be read by any authenticated user
- internal data: curated by an administrator, can be read and updated by any administrator but is not visible to other users
- user data: this should only be read and written by the owning user

We need to be able to write path-based matching rules to define the desired access controls.

Note: the term "administrator" is used loosely here to mean a user with a role granting them elevated privileges, e.g. brand staff or marketing executives.

7.11.1.3 Event Pub/Sub

When certain assets and database records are created or updated, we need to be able to trigger a Cloud Function to run. This is done by configuring events to be published to a Pub/Sub topic, and subscribing a Cloud Function to the topic. The notifications are bucket-wide, but the subscriber can specify a filter to control which events they receive from the topic. The filter syntax is quite limited, allowing only exact or prefix match on the attributes in the event.

7.11.2 Decision

We will store public data under a prefix of `/{assetType}/public/`

We will store shared data under a prefix of `/{assetType}/shared/`

We will store internal data under a prefix of `/{assetType}/internal/`

We will store user data under a prefix of `/{assetType}/user/{userID}/`

There may be several levels of hierarchy within these prefixes.

7.11.2.1 Examples

- image/public/logo.png
- animation/user/enK0PQ8YY0hXDICJm1MKfjYTTvu1/1FYhdl4F9cWx0qT4EoVf.jpg
- animation/shared/1FYhdl4F9cWx0qT4EoVf.jpg
- texture/user/enK0PQ8YY0hXDICJm1MKfjYTTvu1/garment/1FYhdl4F9cWx0qT4EoVf/base.jpg

7.11.3 Conclusion

7.11.3.1 GDPR Compliance

We will be able to comply with GDPR requests as all user-owned data will match a path `/{assetType}/user/{userID}/**`.

7.11.3.2 Access Control

We will be able to write access control rules for user-owned files by writing a match rule with a condition requiring that the user id in the path matches that of the authenticated user. For example:

```
service firebase.storage {
  // Allow the requestor to read or delete any avatar on a path under the
  // user directory.
  match /avatar/user/{userId}/{anyUserFile=**} {
    allow read, delete: if request.auth != null && request.auth.uid == userId;
  }
}
```

For the shared data (admin read/write, user read) we could set up custom claims in Firebase Auth and use them in the rules:

```
service firebase.storage {
  // Allow authenticated users to read
  // Allow users with an `admin` flag set in the user's custom token to write
  match /animation/shared/{anySharedFile=**} {
    allow read: if request.auth != null;
    allow write: if request.auth.token.admin == true;
  }
}
```

7.11.3.3 Event Pub/Sub

7.11.3.4 Subscribers will be able to filter the events they see using a prefix match. For example, the filter attributes.eventType = "OBJECT_FINALIZE" AND hasPrefix(attributes.objectId, "/avatar/") would limit the events seen to avatar creation events.

7.11.4 Appendix

7.11.4.1 Storage Access Paths in Use Cases

7.11.4.1.1 Use Case 1

Filename	Description	Type	Folder Path

nimation-file-{timestamp}	Mixamo animation file	.fbx	`animation>shared`
animation-preview-{timestamp}	An image for preview purposes	.png or .jpg	`animation>shared`
avatar-file-{timestamp}	VStitcher file	.fbx	`avatar>shared`
avatar-preview-{timestamp}	An image for previewing purposes	.png or .jpg	`avatar>shared`
garment-file-{timestamp}	3D Object file	.fbx	`garment>shared`
garment-preview-{timestamp}	An image for previewing purposes	.png or .jpg	`garment>shared`
textures-{documentID}-{timestamp}	Zip file including all of the textures	.zip	`textures>shared`
scanatar-file-{timestamp}	A temporary 3D Object	.ply	`qcscan>shared`
scanatar-meta-{timestamp}	A JSON file that features details about the Scanatar	.json	`qcmeta>shared`

7.11.4.1.2 Use Case 2

Filename	Description	Type	Folder Path
vatar-file-{timestamp}	The avatar that is created for each user	.fbx	`avatar>user>{userID}`

garment-file-{timestamp}	The garment 3D design file	.bw	`garment>internal`
user-media-{timestamp}	A photo taken by the user	.jpg	`collection>user>{userID}`
output-{timestamp}	The synthesized image, output from the platform	.jpg	`collection>user>{userID}`
scanatar-file-{timestamp}	A temporary 3D Object	.ply	`qcscan>user>{userID}`
scanatar-meta-{timestamp}	A JSON file that features details about the Scanatar	.json	`qcmeta>user>{userID}`

7.11.4.1.3 Use Case 3

Filename	Description	Type	Folder Path
vatar-file-{timestamp}	The avatar that is created for each user	.fbx	`avatar>user>{userID}`
garment-file-{timestamp}	The garment 3D object file	.fbx	`garment>shared`
textures-{documentID}-{timestamp}	Zip file including all of the textures	.zip	`textures>shared`
photo-{documentID}-{timestamp}	Photos that are saved by the user when trying on a garment using the camera	.jpg	`photos>user>{userID}`

scanatar-file-{timestamp}	A temporary 3D Object	.ply	`qcscan>user>{userID}`
scanatar-meta-{timestamp}	A JSON file that features details about the Scanatar	.json	`qcmeta>user>{userID}`

7.11.4.1.4 Term Index

- `documentID`: Google Firestore Database saves and handles data into Collections. Data entries written in a collection are assigned a unique id. This ID is used in Firebase Storage to connect assets to collection items.
- `userID`: Each registered user has a unique identifier in Firestore Database. This user id is used in some parts in Firebase Storage path scheme to denote ownership of assets.
- `timestamp`: The timestamp is expressed in milliseconds since the Unix Epoch.

7.12 ADR-0023 Use Custom Claims

Date	2021-09-22
Status	Proposed

7.12.1 Context

In ADR 6 we note that we will use Firebase Authentication in all the user-facing applications. ADR 16 expands on this and describes the authentication requirements for each use-case. The following table summarizes the roles identified in ADR 16.

Use Case	Role	Firebase Auth?	Registration Ux?
	Designer	Yes	No
1	Product Manager	Yes	No
1	eTryOn Admin	No	No
2	Influencer	Yes	Yes

2	Marketing Exec	No	No
3	User	Yes	Yes
3	3D Tech Designer	No	No

The roles that don't use Firebase Auth will instead use standard Google Cloud IAM permissions and are out of scope of this document.

Use cases 2 and 3 allow users to self-register to use the eTryOn applications and all self-registered users have the same permissions: these cases require no special treatment.

For use case 1, we will not provide a Ux for users to self-manage their accounts: account creation will be done by eTryOn staff through the Firebase Admin Auth API. However, a malicious user might still be able to access the public Firebase APIs for our project and create unauthorized accounts.

7.12.2 Decision

We will use a Custom Claim when we create accounts through the Admin API. This could either be a `role` field or an `authorized` flag. These claims cannot be set by a self-registered user.

7.12.3 Consequences

- We will have to write security rules for the VR Asset Store (Cloud Storage) and VR Data Store (Firestore) with conditions restricting access to users with the required custom claim set. See Custom-claim attributes and roles for some examples.
- Cloud functions called from a Firebase app will have to check for the custom claim in the auth context and reject attempts at unauthorized access.

7.13 ADR-0024 Unity Analytics Events to Server Event Bus

Date	2021-09-29
Status	Proposed

7.13.1 Context

For Use Case 3 Magic Mirror app, Google Firebase Analytics is used to track usage behavior and log a variety of events. These events must be fed to Mallzee's service in order to generate user-specific garment suggestions.

Analytics Data tracking is implemented through the SDK provided by Google for using Firebase within Unity applications.

The Firebase SDK offers support for logging two primary types of information:

- Events: log user actions, system events, errors.
- User properties: attributes that describe user base, such as geographical information and language preference

In the context of the Magic Mirror mobile app, a variety of events will be tracked, such as user clicks and user engagement time. Moreover, some user properties will be recorded too, such as user Operating System, OS Version, geographical information, gender, and age.

Firebase Analytics does not offer an API that can be queried to get analytics data and feed them to Mallzee's service.

Thus another solution must be found in order to fetch analytics events.

7.13.2 Decision

Google Cloud's Big Query is a serverless, highly scalable, and cost-effective multicloud data warehouse, which can be integrated with Firebase Analytics. It features a REST API which can be called to fetch our Firebase Analytics data.

It requires a BLAZE (paid) Firebase plan.

7.13.3 Consequences

We will be using the integration between Firebase Analytics and Google Cloud Big Query in order to feed Mallzee's user-specific garment suggestions function, with the events from the Magic Mirror app.

8 Appendix B - API specifications from https://etryon.gitlab.io/techdocs/

8.1 Mallzee eTryOn API

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: Mallzee eTryOn
servers:
  - url: 'https://etryon.mallzee.com'
security:
  - ApiKeyAuth: []

paths:
  /products/performance:
    post:
      summary: Product performance prediction
      description: 'Predicts the performance of a product'
      operationId: productPerformance
      tags:
        - product
      requestBody:
          description: 'The details of the product and market segements being
targeted to make the product prediction'
          required: true
          content:
              application/json:
                  schema:
                      $ref: '#/components/schemas/PerformanceInput'
                  examples:
                    productPrediction:
                        $ref: '#/components/examples/PerformanceInput'
      responses:
        '200':
          description: 'Product performance results'
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/PerformanceOutput'
              example:
                value:
                  products:
                    -
                      id: d33e1e13-3828-4178-926b-a1ca24997136
                      score: 95
        default:
          description: Unexpected error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
              example:
                  value:
                    code: 400
                    message: 'Invalid product properties'
  /products/recommendations:
    post:
      summary: Product recommendations
      description: 'Returns recommended products based on the given product and user
information'
      operationId: productRecommendations
      tags:
        - products
      requestBody:
          description: 'The details of the product and current user information make
the product recommendations'
          required: true
          content:
              application/json:
                  schema:
                      $ref: '#/components/schemas/RecommendationsInput'
                  examples:
                      productRecommendation:
                        $ref: '#/components/examples/RecommendationsInput'
      responses:
        '200':
          description: 'Product recommendation results'
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RecommendationsOutput'
              example:
                  value:
                      products:
                        - 288517e2-3f17-4e31-b6e6-1f8d6ca5edf9
                        - 7eb602dc-975e-4642-91cf-6e069bc9cf03
                        - 29a7bb26-f826-4e06-86e1-b2fb11d1d317
        default:
          description: Unexpected error
          content:
            application/json:
              schema:
```

```yaml
                $ref: '#/components/schemas/Error'
              example:
                  value:
                    code: 400
                    message: 'Invalid product properties'
components:
  securitySchemes:
    ApiKeyAuth:
      type: apiKey
      name: x-api-key
      in: header
  schemas:
    MarketSegment:
        type: object
        required:
            - age_min
            - age_max
            - price_min
            - price_max
        properties:
            age_min:
                type: integer
                format: int32
            age_max:
                type: integer
                format: int32
            price_min:
                type: integer
                format: int32
            price_max:
                type: integer
                format: int32
    User:
      type: object
      properties:
        age:
            type: integer
            format: int32
    Product:
      type: object
      required:
        - name
        - description
        - colour
        - image
      properties:
        name:
            type: string
        description:
            type: string
        colour:
            type: string
        image:
            type: string
    PerformanceInput:
        type: object
        properties:
            products:
                type: array
                items:
                    $ref: '#/components/schemas/Product'
            market_segement:
                $ref: '#/components/schemas/MarketSegment'
    PerformanceOutput:
        type: object
        properties:
            products:
                type: array
                items:
                    type: object
                    properties:
                        id:
                            type: string
                        score:
                            type: integer
                            format: int32
    RecommendationsInput:
        type: object
        properties:
            product:
                $ref: '#/components/schemas/Product'
            user:
                $ref: '#/components/schemas/User'
    RecommendationsOutput:
        type: object
        properties:
            products:
                type: array
                items:
                    type: string
    Error:
      type: object
      required:
        - code
        - message
      properties:
        code:
          type: integer
          format: int32
        message:
          type: string
  examples:
    Product:
      value:
        id: 'd33e1e13-3828-4178-926b-a1ca24997136'
        name: Mini dress
        description: A green striped mini dress
        colour: green
        image:
'https://images.mallzee.com/products/d33e1e13-3828-4178-926b-a1ca24997136'
    User:
      value:
        age: 38
    MarketSegment:
```

```
   value:
    age_min: 21
    age_max: 35
    price_min: 30
    price_max: 60
  PerformanceInput:
   value:
    products:
      -
        id: 'd33e1e13-3828-4178-926b-a1ca24997136'
        name: Mini dress
        description: A green striped mini dress
        colour: green
        image: 'https://images.mallzee.com/products/d33e1e13-3828-4178-926b-a1ca24997136'
    market_segement:
      age_min: 21
      age_max: 35
      price_min: 30
      price_max: 60
  RecommendationsInput:
   value:
    products:
      -
        id: 'd33e1e13-3828-4178-926b-a1ca24997136'
        name: Mini dress
        description: A green striped mini dress
        colour: green
        image: 'https://images.mallzee.com/products/d33e1e13-3828-4178-926b-a1ca24997136'
    user:
      age: 38
```

8.2 Metail Scanatar Service

In ADR 12 we note that the Metail Scanatar Creation service will be invoked by starting an AWS Step Function execution.

The StartExecution API endpoint is documented here. The parameters of interest are:

Field	Description

input	JSON input data for the execution
name	The name of the execution (a random UUID)
stateMachineArn	ARN of the step function - will be known once the endpoint is in place

8.2.1 Input

The step function input JSON (serialized to a string in the input field above) must conform to the following schema:

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://etryon-h2020.eu/schema/metail/scanatar-service.json",
    "title": "Metail Scanatar Service",
    "description": "Step function input for Metail Scanatar creation",
    "type": "object",
    "properties": {
        "inputScan": {
            "description": "Presigned GET URL for the Quantacorp scan in Google Cloud Storage",
            "type": "string",
            "format": "uri"
        },
        "outputAvatar": {
            "description": "Presigned PUT URL for the output scanatar in Google Cloud Storage",
            "type": "string",
            "format": "uri"
        },
        "gender": {
            "description": "Gender of the subject",
            "type": "string",
            "enum": ["male", "female"]
        },
        "height": {
            "description": "Height of the subject in millimeters",
            "type": "integer"
        },
        "avatarCompatibility": {
            "description": "Type of avatar to create",
            "type": "string",
            "enum": ["vstitcher", "metail", "unity"]
        }
    },
    "required": ["inputScan", "outputAvatar", "gender", "height",
"avatarCompatibility"],
    "examples": [
        {
              "inputScan":
"https://storage.googleapis.com/example-bucket/scan/user/enK0PQ8YY0hXDICJm1MKfjYTTvu
1/d072dffc-1aeb-11ec-a8f1-ffa30997c493.obi?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-
Credential=example%40example-project.iam.gserviceaccount.com%2F20181026%2Fus-central
-1%2Fstorage%2Fgoog4_request&X-Goog-Date=20181026T181309Z&X-Goog-Expires=900&X-Goog-
SignedHeaders=host&X-Goog-Signature=247a2aa45f169edf4d187d54e7cc46e4731b1e6273242c4f
4c39a1d2507a0e58706e25e3a85a7dbb891d62afa8496def8e260c1db863d9ace85ff0a184b894b117fe
46d1225c82f2aa19efd52cf21d3e2022b3b868dcc1aca2741951ed5bf3bb25a34f5e9316a2841e8ff4c5
30b22ceaa1c5ce09c7cbb5732631510c20580e61723f5594de3aea497f195456a2ff2bdd0d13bad47289
d8611b6f9cfeef0c46c91a455b94e90a66924f722292d21e24d31dcfb38ce0c0f353ffa5a9756fc2a9f2
b40bc2113206a81e324fc4fd6823a29163fa845c8ae7eca1fcf6e5bb48b3200983c56c5ca81fffb151cc
a7402beddfc4a76b133447032ea7abedc098d2eb14a7",
            "outputAvatar":
"https://storage.googleapis.com/example-bucket/avatar/user/enK0PQ8YY0hXDICJm1MKfjYTT
vu1/d072dffc-1aeb-11ec-a8f1-ffa30997c493.fbx?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goo
g-Credential=example%40example-project.iam.gserviceaccount.com%2F20181026%2Fus-centr
al-1%2Fstorage%2Fgoog4_request&X-Goog-Date=20181026T181309Z&X-Goog-Expires=900&X-Goo
g-SignedHeaders=host&X-Goog-Signature=247a2aa45f169edf4d187d54e7cc46e4731b1e6273242c
4f4c39a1d2507a0e58706e25e3a85a7dbb891d62afa8496def8e260c1db863d9ace85ff0a184b894b117
fe46d1225c82f2aa19efd52cf21d3e2022b3b868dcc1aca2741951ed5bf3bb25a34f5e9316a2841e8ff4
c530b22ceaa1c5ce09c7cbb5732631510c20580e61723f5594de3aea497f195456a2ff2bdd0d13bad472
89d8611b6f9cfeef0c46c91a455b94e90a66924f722292d21e24d31dcfb38ce0c0f353ffa5a9756fc2a9
f2b40bc2113206a81e324fc4fd6823a29163fa845c8ae7eca1fcf6e5bb48b3200983c56c5ca81fffb151
cca7402beddfc4a76b133447032ea7abedc098d2eb14a7",
            "gender": "female",
            "height": 1752
        }
    ]
}
```

8.2.2 Note on implementation

Developers rarely need to be concerned with the underlying REST API for AWS services - it is much more common to use one of the AWS SDKs. For example, to invoke the scanatar step function from a Python program you would use the boto3 step function client and your code would look something like:

```python
import json
import uuid
import boto3

SFN_ARN =
'arn:aws:states:{region}:{accountId}:stateMachine:etryon-scanatar-creation'
INPUT_URL = 'https://storage.googleapis.com/example-bucket/scan/user/...'
OUTPUT_URL = 'https://storage.googleapis.com/example-bucket/avatar/user/...'

client = boto3.client('stepfunctions')
response = client.start_execution(
    stateMachineArn=SFN_ARN,
    name=str(uuid.uuid4()),
    input=json.dumps({
          "inputScan": INPUT_URL,
        "outputAvatar": OUTPUT_URL,
        "gender": "female",
        "height": 1752
    })
)
print(f'Started execution {response["executionArn"]}')
```

8.3 QuantaCorp API

8.3.1 Authorization API

```yaml
openapi: 3.0.0

info:
  title: QuantaCorp Authorization API
  description: This document contains the specs of the QuantaCorp Authorization API
used at the initial phase of eTryOn iterative testing.
  version: 1.28.12

servers:
  - url: 'https://api.quantacorp.io/authorization'

externalDocs:
  description: Find out more about the QuantaCorp Authorization API and how to use
it.
```

```
  url: 'https://docs.quantacorp.io'

security:
  - qc_basic: []

paths:
  /oauth2/token:
    post:
      summary: Create an oauth2 token.
      description: Request an oauth2 token by means of a username and password, or
by means of a refresh token.
      requestBody:
        content:
          application/x-www-form-urlencoded:
            schema:
              oneOf:
                - $ref: '#/components/schemas/UsernamePasswordForm'
                - $ref: '#/components/schemas/RefreshTokenForm'
        description: Form containg authentication data to request oauth2 token.
        required: true
      responses:
        '201':
          description: Created.
        '401':
          description: Unauthorized.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/OAuth2Error'
        '500':
          description: Internal server error.

components:
  securitySchemes:
    qc_basic:
      type: http
      scheme: basic
  schemas:
    OAuth2Error:
      type: object
      properties:
        error:
          type: string
          enum:
            - invalid_client
            - invalid_request
            - unauthorized_client
            - invalid_token
            - invalid_grant
        error_description:
          type: string

    RefreshTokenForm:
      type: object
      properties:
        grant_type:
          type: string
          enum:
            - refresh_token
        refresh_token:
          type: string

    UsernamePasswordForm:
      type: object
      properties:
        grant_type:
          type: string
          enum:
            - password
        username:
          type: string
        password:
          type: string
          format: password
    TokenDTO:
      type: object
      properties:
        token_type:
          type: string
          enum:
            - Bearer
        access_token:
          type: string
        refresh_token:
          type: string
        expires_in:
          type: integer
          format: int32
```

8.3.2 Public API

```
openapi: 3.0.0

info:
  title: QuantaCorp Public API
  description: This document contains the specs of the QuantaCorp Public API used at the initial phase of eTryOn iterative testing.
  version: 1.28.12

servers:
  - url: 'https://api.quantacorp.io/public-api'

externalDocs:
  description: Find out more about the QuantaCorp Public API and how to use it.
  url: 'https://docs.quantacorp.io'

security:
  - qc_oauth2: []

paths:
  /body:
    post:
      summary: Add a new body.
```

```yaml
      description: Scans can only be added to specific project and body. In order to add a scan, a body has to be created first.
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/CreateBodyDTO'
        description: Body object that needs to be created. Mind that the alias has to be unique within the company chain.
        required: true
      responses:
        '201':
          description: Created.
          headers:
            Location:
              schema:
                $ref: '#/components/schemas/Location'
        '400':
          description: Bad request.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RestExceptionDTO'
        '500':
          description: Internal server error.

  /scan/project/{projectId}/body/{bodyId}:
    post:
      summary: Add a new scan.
      description: Create a scan for a given project and body.
      parameters:
        - in: path
          name: projectId
          schema:
            $ref: '#/components/schemas/NumericIdentifier'
          required: true
          description: Numeric ID of the project for which to add a scan
        - in: path
          name: bodyId
          schema:
            $ref: '#/components/schemas/NumericIdentifier'
          required: true
          description: Numeric ID of the body for which to add a scan
      requestBody:
        content:
          multipart/form-data:
            schema:
              type: object
              properties:
                metadata:
                  $ref: '#/components/schemas/ScanMetadataDTO'
                front:
                  type: string
                  format: binary
                side:
                  type: string
                  format: binary
            encoding:
              metadata:
                contentType: application/json
              front:
                contentType: image/png
              side:
                contentType: image/png
        description: Body object that needs to be created. Mind that the alias has
to be unique within the company chain.
        required: true
      responses:
        '202':
          description: Accepted. All scan data is uploaded. The scan is being
processed and is expected to finish in the amount of seconds indicated by the
X-Processing-Time header.
          headers:
            Location:
              schema:
                $ref: '#/components/schemas/Location'
            X-Processing-Time:
              schema:
                $ref: '#/components/schemas/X-Processing-Time'
        '400':
          description: Bad request.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RestExceptionDTO'
        '500':
          description: Internal server error.

  /scan/{scanId}/callbacks:
    post:
      summary: Add callbacks to a scan.
      description: Create callbacks for a given scan. Handling callbacks is the last
step in the processing of a scan. Make sure to send the callbacks-request timely.
      parameters:
        - in: path
          name: scanId
          schema:
            $ref: '#/components/schemas/NumericIdentifier'
          required: true
          description: Numeric ID of the scan for which to add callbacks
      requestBody:
        content:
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/ScanCallbacksForm'
        description: Object containing all callbacks for a scan.
        required: true
      responses:
        '200':
          description: OK.
        '400':
          description: Bad request.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RestExceptionDTO'
        '500':
          description: Internal server error.

components:
  securitySchemes:
    qc_oauth2:
      type: oauth2
      flows:
        password:
          tokenUrl: 'https://api.quantacorp.io/authorization/oauth2/token'
          refreshUrl: 'https://api.quantacorp.io/authorization/oauth2/token'
          scopes: {}

  schemas:
    Location:
      type: string
      pattern: '^[a-z]+/[0-9]+$'
      description: Location header consists of the name of the resource followed by
a forward slash and the numeric identifier (int64).
      example:
        body:
          value: body/1234
          summary: The location header for a body with ID 1234.
        scan:
          value: scan/5678
          summary: The location header for a scan with ID 5678.

    X-Processing-Time:
      type: integer
      format: int32
      description: The amount of time in seconds the client should wait before
fetching scan result.

    NumericIdentifier:
      type: integer
      format: int64

    RestExceptionDTO:
      type: object
      properties:
        errorReason:
          type: string
        errorCode:
          type: integer
          format: int64
        externalMessage:
          type: string

    CreateBodyDTO:
      type: object
      properties:
        company_id:
          type: integer
          format: int64
        link_to_project:
          type: integer
          format: int64
        alias:
          type: string
        height:
          type: integer
          format: int32
        gender:
          type: string
          enum:
            - FEMALE
            - MALE
            - UNKNOWN
        weight:
          type: integer
          format: int32
      required:
        - company_id
        - link_to_project
        - alias
        - height
        - gender
```

```yaml
    ScanMetadataDTO:
      type: object
      properties:
        front:
          $ref: '#/components/schemas/PictureMetadataDTO'
        side:
          $ref: '#/components/schemas/PictureMetadataDTO'

    PictureMetadataDTO:
      type: object
      properties:
        gender:
          type: string
          enum:
            - F
            - M
            - U
        height:
          type: integer
          format: int32
        accelerometerX:
          type: number
          format: double
        accelerometerY:
          type: number
          format: double
        accelerometerZ:
          type: number
          format: double
        weight:
          type: integer
          format: int32
        fov_deg:
          type: number
          format: double
      required:
        - gender
        - height
        - accelerometerX
        - accelerometerY
        - accelerometerZ

    ScanCallbacksForm:
      type: object
      properties:
        ply_scape_callback:
          type: string
        etryon_meta_callback:
          type: string
```

8.4 UC2 DressMeUp Metail Composition Service

The Metail Composition service will be invoked by starting an AWS Step Function execution. The StartExecution API endpoint is documented here. The parameters of interest are:

Field	Description
input	JSON input data for the execution
name	The name of the execution (a random UUID)
stateMachineArn	ARN of the step function - will be known once the endpoint is in place

8.4.1 Input

The step function input JSON (serialized to a string in the input field above) must conform to the following schema:

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://etryon-h2020.eu/schema/metail/uc2-composition-service.json",
    "title": "Metail Composition Service",
    "description": "Step function input for composing 3D garments into 2D user
photos",
    "type": "object",
    "properties": {
        "userAvatar": {
            "description": "Presigned GET URL for the Metail avatar in Google Cloud Storage",
            "type": "string",
            "format": "uri"
        },
        "userPhoto": {
            "description": "Presigned GET URL for the photo of the user that the garment will be composed into, in Google Cloud Storage",
            "type": "string",
            "format": "uri"
        },
        "garment": {
            "description": "Presigned GET URL for the browzwear file containing the garment to be composed, in Google Cloud Storage",
            "type": "string",
            "format": "uri"
        },
        "outputImage": {
            "description": "Presigned PUT URL for the output image in Google Cloud Storage",
            "type": "string",
            "format": "uri"
        }
    },
    "required": ["userAvatar", "userPhoto", "garment", "outputImage"],
    "examples": [
        {
              "userAvatar": "https://storage.googleapis.com/see-adr-0021",
            "userPhoto": "https://storage.googleapis.com/see-adr-0021",
            "garment": "https://storage.googleapis.com/see-adr-0021",
            "outputImage": "https://storage.googleapis.com/see-adr-0021"
        }
    ]
}
```

8.4.2 Note on implementation

Developers rarely need to be concerned with the underlying REST API for AWS services - it is much more common to use one of the AWS SDKs. For example, to invoke the scanatar step function from a Python program you would use the boto3 step function client and your code would look something like:

```
import json
import uuid
import boto3

SFN_ARN =
'arn:aws:states:{region}:{accountId}:stateMachine:etryon-dress-me-up-composition'

client = boto3.client('stepfunctions')
response = client.start_execution(
    stateMachineArn=SFN_ARN,
    name=str(uuid.uuid4()),
    input=json.dumps({
          "userAvatar": "https://storage.googleapis.com/...",
        "userPhoto": "https://storage.googleapis.com/...",
        "garment": "https://storage.googleapis.com/...",
        "outputImage": "https://storage.googleapis.com/..."

    })
)
print(f'Started execution {response["executionArn"]}')
```

8.4.3 Operation

See the sequence diagram for more details. A high level view of the operation of this service is as follows: -

1. Download inputs
2. Estimates user pose in photo.
3. Finds closest reference avatar to user.
4. Loads garment bw file
5. Applies pose to the dressed reference avatar
6. Composition-raytrace render of garment solution
7. Compose raytrace render into original photo to create Result image

8.5 UC3 Garment Model Creation Service

The Metail Garment Model Creation service generates an OBI garment from an input Browzwear file. It will be invoked by starting an AWS Step Function execution.

The StartExecution API endpoint is documented here. The parameters of interest are:

Field	Description
input	JSON input data for the execution
name	The name of the execution (a random UUID)
stateMachineArn	ARN of the step function - will be known once the endpoint is in place

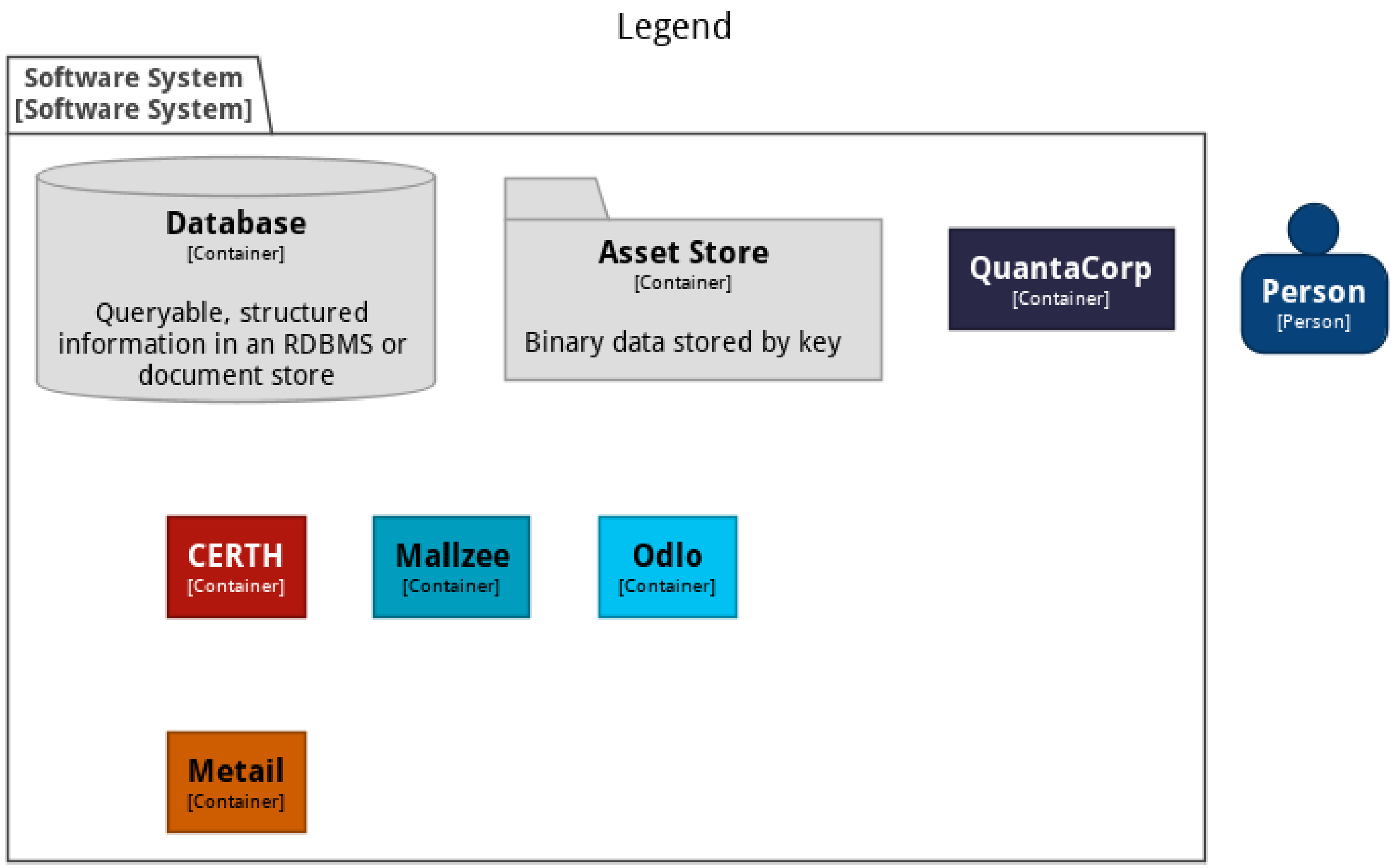
8.5.1 Input

The step function input JSON (serialized to a string in the input field above) must conform to the following schema:

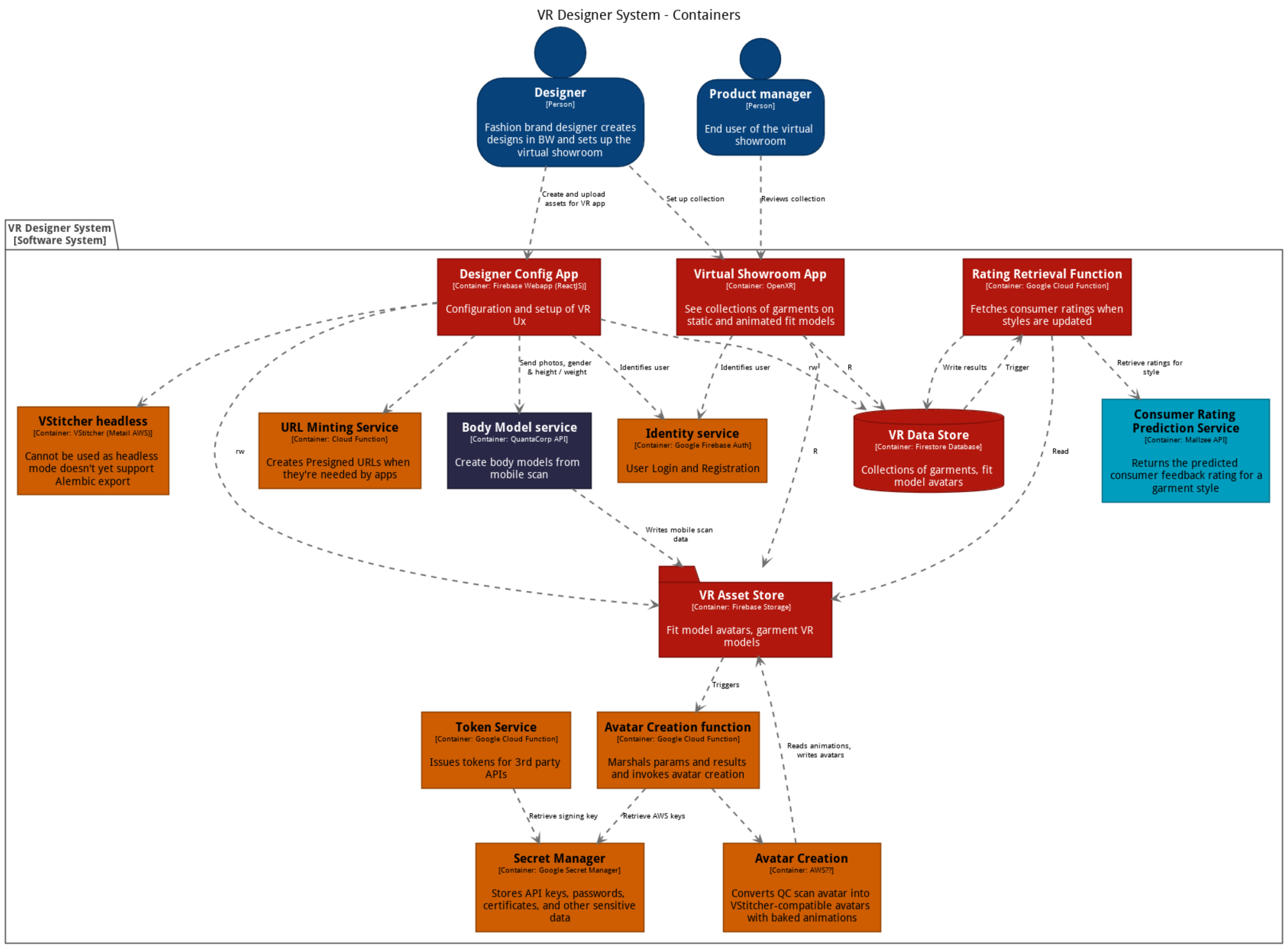
```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id":
"https://etryon-h2020.eu/schema/metail/uc3-garment-model-creation-service.json",
    "title": "Metail Garment Model Creation Service",
    "description": "Step function input for creating garment models",
    "type": "object",
    "properties": {
        "browzwearGarment": {
            "description": "Presigned GET URL for Browzwear file in Google Cloud
Storage",
            "type": "string",
            "format": "uri"
        },
        "obiGarment": {
            "description": "Presigned PUT URL for the output OBI garment in Google
Cloud Storage",
            "type": "string",
            "format": "uri"
        }
    },
    "required": ["browzwearGarment", "obiGarment"],
    "examples": [
        {
            "browzwearGarment": "https://storage.googleapis.com/see-adr-0021",
            "obiGarment": "https://storage.googleapis.com/see-adr-0021"
        }
    ]
}
```

9 Appendix C - Architecture overview diagrams from https://etryon.gitlab.io/techdocs/

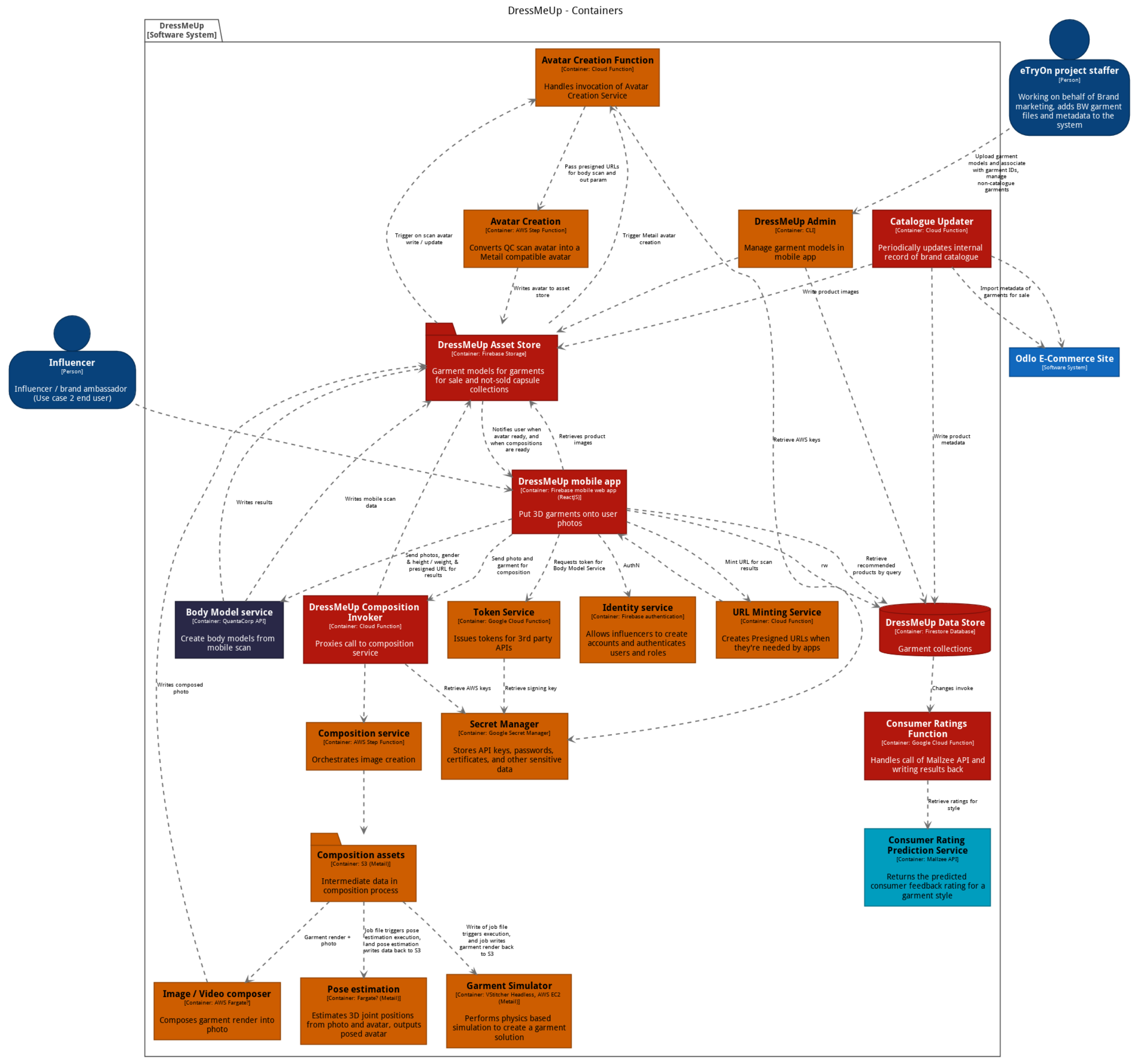
This legend describes the colour coding and shape keys used in the architecture diagrams.



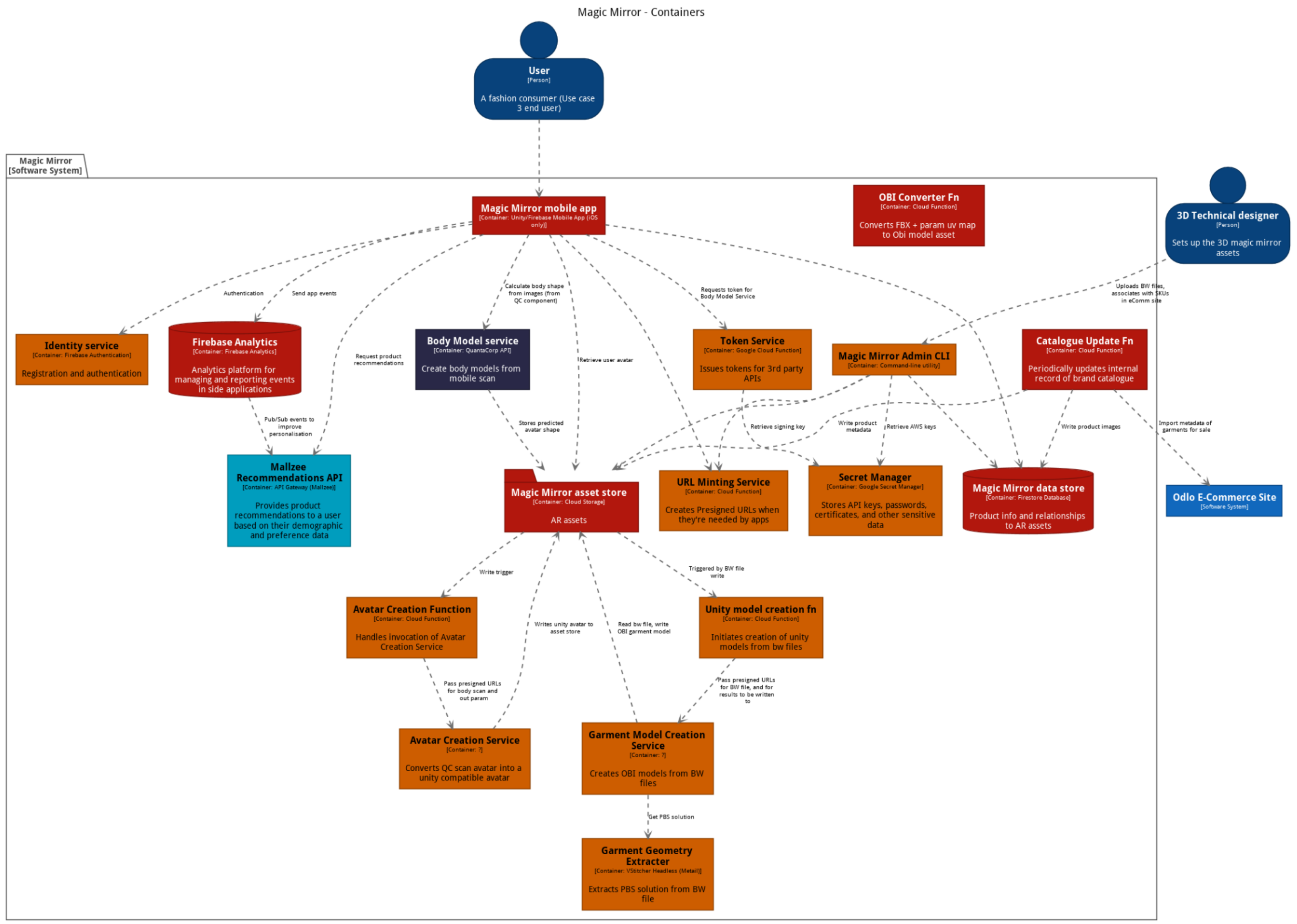
9.1 VR Designer System (https://etryon.gitlab.io/techdocs/arch-diags/use-case-1/)


VR Designer System - Containers
Designer
[Person]
Fashion brand designer creates designs in BW and sets up the virtual showroom
Product manager
[Person]
End user of the virtual showroom
Create and upload assets for VR app
Set up collection
Reviews collection
VR Designer System
[Software System]
Designer Config App
[Container: Firebase Webapp (ReactJS)]
Configuration and setup of VR Ux
Virtual Showroom App
[Container: OpenXR]
See collections of garments on static and animated fit models
Rating Retrieval Function
[Container: Google Cloud Function]
Fetches consumer ratings when styles are updated
Send photos, gender & height / weight
Identifies user
Identifies user
rw
R
Write results
Trigger
Retrieve ratings for style
VStitcher headless
[Container: VStitcher (Metail AWS)]
Cannot be used as headless mode doesn't yet support Alembic export
URL Minting Service
[Container: Cloud Function]
Creates Presigned URLs when they're needed by apps
Body Model service
[Container: QuantaCorp API]
Create body models from mobile scan
Identity service
[Container: Google Firebase Auth]
User Login and Registration
VR Data Store
[Container: Firestore Database]
Collections of garments, fit model avatars
Consumer Rating Prediction Service
[Container: Mallzee API]
Returns the predicted consumer feedback rating for a garment style
rw
R
Read
Writes mobile scan data
VR Asset Store
[Container: Firebase Storage]
Fit model avatars, garment VR models
Triggers
Reads animations, writes avatars
Token Service
[Container: Google Cloud Function]
Issues tokens for 3rd party APIs
Avatar Creation function
[Container: Google Cloud Function]
Marshals params and results and invokes avatar creation
Retrieve signing key
Retrieve AWS keys
Secret Manager
[Container: Google Secret Manager]
Stores API keys, passwords, certificates, and other sensitive data
Avatar Creation
[Container: AWS??]
Converts QC scan avatar into VStitcher-compatible avatars with baked animations

9.2 DressMeUp System (https://etryon.gitlab.io/techdocs/arch-diags/use-case-2/)


DressMeUp - Containers
DressMeUp
[Software System]
Avatar Creation Function
[Container: Cloud Function]
Handles invocation of Avatar Creation Service
eTryOn project staffer
[Person]
Working on behalf of Brand marketing, adds BW garment files and metadata to the system
Upload garment models and associate with garment IDs, manage non-catalogue garments
Pass presigned URLs for body scan and out param
Trigger on scan avatar write / update
Trigger Metail avatar creation
Avatar Creation
[Container: AWS Step Function]
Converts QC scan avatar into a Metail compatible avatar
DressMeUp Admin
[Container: CLI]
Manage garment models in mobile app
Catalogue Updater
[Container: Cloud Function]
Periodically updates internal record of brand catalogue
Writes avatar to asset store
Write product images
Import metadata of garments for sale
DressMeUp Asset Store
[Container: Firebase Storage]
Garment models for garments for sale and not-sold capsule collections
Odlo E-Commerce Site
[Software System]
Influencer
[Person]
Influencer / brand ambassador (Use case 2 end user)
Notifies user when avatar ready, and when compositions are ready
Retrieves product images
Retrieve AWS keys
Write product metadata
Writes results
Writes mobile scan data
DressMeUp mobile app
[Container: Firebase mobile web app (ReactJS)]
Put 3D garments onto user photos
Send photos, gender & height / weight, & presigned URL for results
Send photo and garment for composition
Requests token for Body Model Service
AuthN
Mint URL for scan results
rw
Retrieve recommended products by query
Body Model service
[Container: QuantaCorp API]
Create body models from mobile scan
DressMeUp Composition Invoker
[Container: Cloud Function]
Proxies call to composition service
Token Service
[Container: Google Cloud Function]
Issues tokens for 3rd party APIs
Identity service
[Container: Firebase authentication]
Allows influencers to create accounts and authenticates users and roles
URL Minting Service
[Container: Cloud Function]
Creates Presigned URLs when they're needed by apps
DressMeUp Data Store
[Container: Firestore Database]
Garment collections
Writes composed photo
Retrieve AWS keys
Retrieve signing key
Changes invoke
Composition service
[Container: AWS Step Function]
Orchestrates image creation
Secret Manager
[Container: Google Secret Manager]
Stores API keys, passwords, certificates, and other sensitive data
Consumer Ratings Function
[Container: Google Cloud Function]
Handles call of Mallzee API and writing results back
Retrieve ratings for style
Composition assets
[Container: S3 (Metail)]
Intermediate data in composition process
Consumer Rating Prediction Service
[Container: Mallzee API]
Returns the predicted consumer feedback rating for a garment style
Garment render + photo
Job file triggers pose estimation execution, and pose estimation writes data back to S3
Write of job file triggers execution, and job writes garment render back to S3
Image / Video composer
[Container: AWS Fargate?]
Composes garment render into photo
Pose estimation
[Container: Fargate? (Metail)]
Estimates 3D joint positions from photo and avatar, outputs posed avatar
Garment Simulator
[Container: VStitcher Headless, AWS EC2 (Metail)]
Performs physics based simulation to create a garment solution

9.3 Magic Mirror System (https://etryon.gitlab.io/techdocs/arch-diags/use-case-3/)


Magic Mirror - Containers
User
[Person]
A fashion consumer (Use case 3 end user)
Magic Mirror
[Software System]
Magic Mirror mobile app
[Container: Unity/Firebase Mobile App (iOS only)]
OBI Converter Fn
[Container: Cloud Function]
Converts FBX + param uv map to Obi model asset
3D Technical designer
[Person]
Sets up the 3D magic mirror assets
Authentication
Send app events
Calculate body shape from images (from QC component)
Requests token for Body Model Service
Uploads BW files, associates with SKUs in eComm site
Identity service
[Container: Firebase Authentication]
Registration and authentication
Firebase Analytics
[Container: Firebase Analytics]
Analytics platform for managing and reporting events in side applications
Request product recommendations
Body Model service
[Container: QuantaCorp API]
Create body models from mobile scan
Retrieve user avatar
Token Service
[Container: Google Cloud Function]
Issues tokens for 3rd party APIs
Magic Mirror Admin CLI
[Container: Command-line utility]
Catalogue Update Fn
[Container: Cloud Function]
Periodically updates internal record of brand catalogue
Pub/Sub events to improve personalisation
Stores predicted avatar shape
Retrieve signing key
Write product metadata
Retrieve AWS keys
Write product images
Import metadata of garments for sale
Mallzee Recommendations API
[Container: API Gateway (Mallzee)]
Provides product recommendations to a user based on their demographic and preference data
Magic Mirror asset store
[Container: Cloud Storage]
AR assets
URL Minting Service
[Container: Cloud Function]
Creates Presigned URLs when they're needed by apps
Secret Manager
[Container: Google Secret Manager]
Stores API keys, passwords, certificates, and other sensitive data
Magic Mirror data store
[Container: Firestore Database]
Product info and relationships to AR assets
Odlo E-Commerce Site
[Software System]
Write trigger
Triggered by BW file write
Avatar Creation Function
[Container: Cloud Function]
Handles invocation of Avatar Creation Service
Writes unity avatar to asset store
Read bw file, write OBI garment model
Unity model creation fn
[Container: Cloud Function]
Initiates creation of unity models from bw files
Pass presigned URLs for body scan and out param
Pass presigned URLs for BW file, and for results to be written to
Avatar Creation Service
[Container: ?]
Converts QC scan avatar into a unity compatible avatar
Garment Model Creation Service
[Container: ?]
Creates OBI models from BW files
Get PBS solution
Garment Geometry Extracter
[Container: VStitcher Headless (Metail)]
Extracts PBS solution from BW file

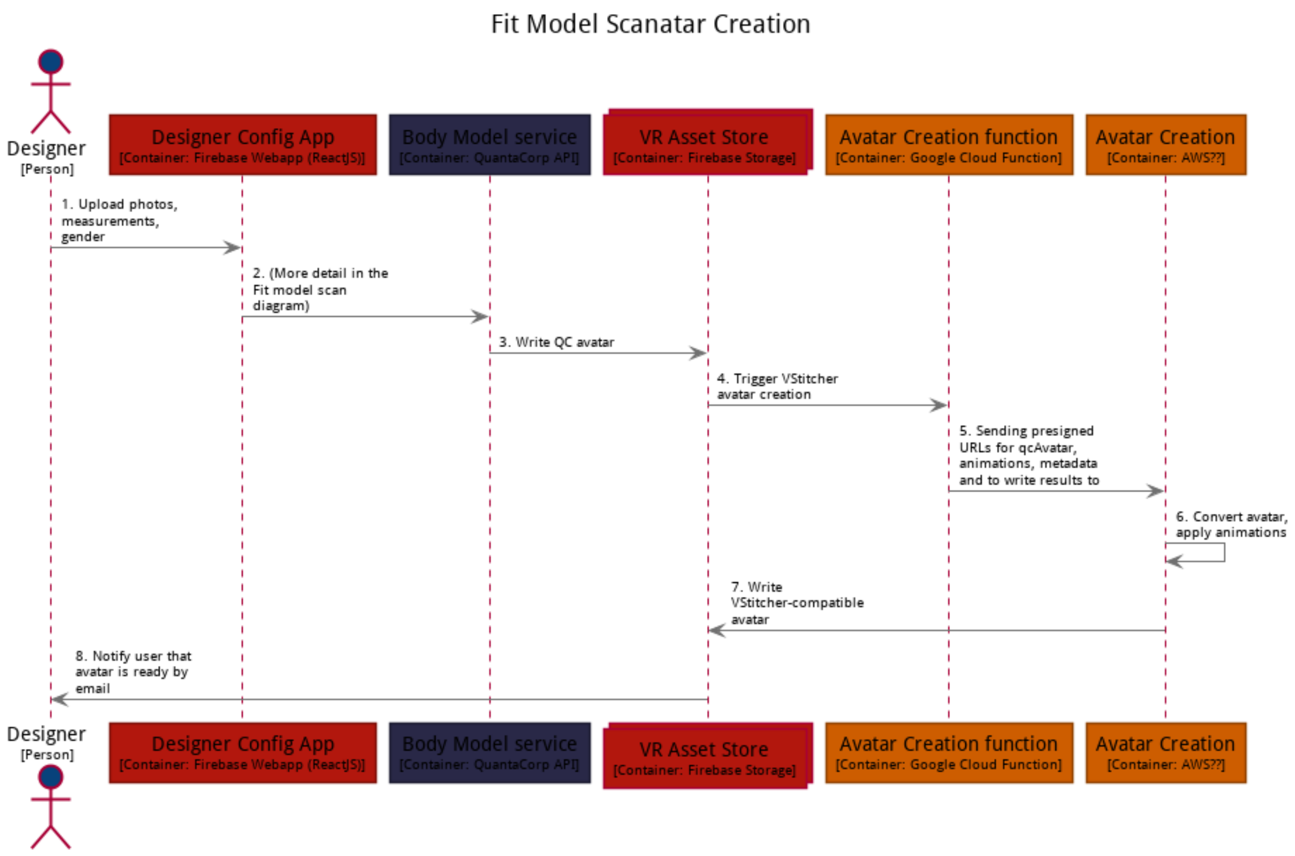
10 Appendix D - Cloud Function specifications from https://etryon.gitlab.io/techdocs/

10.1 Avatar Creation

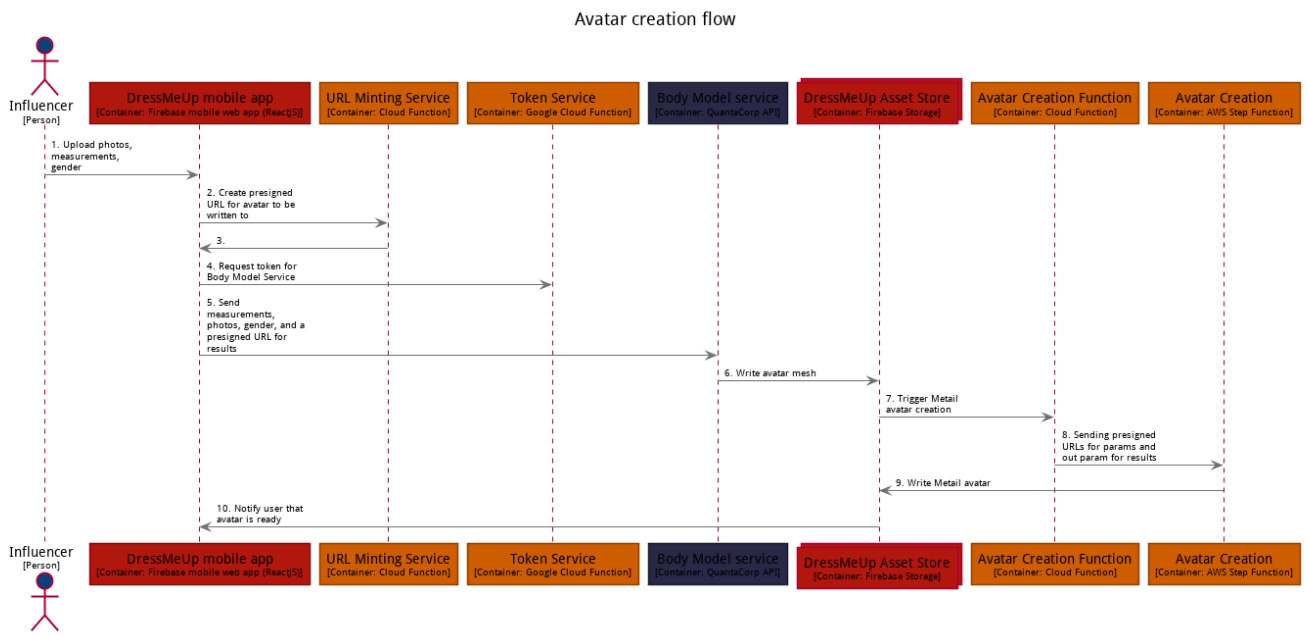
10.1.1 Context

An avatar creation cloud function is required for all three use-cases. It is the interface between the eTryOn systems and the Metail Scanatar Service.

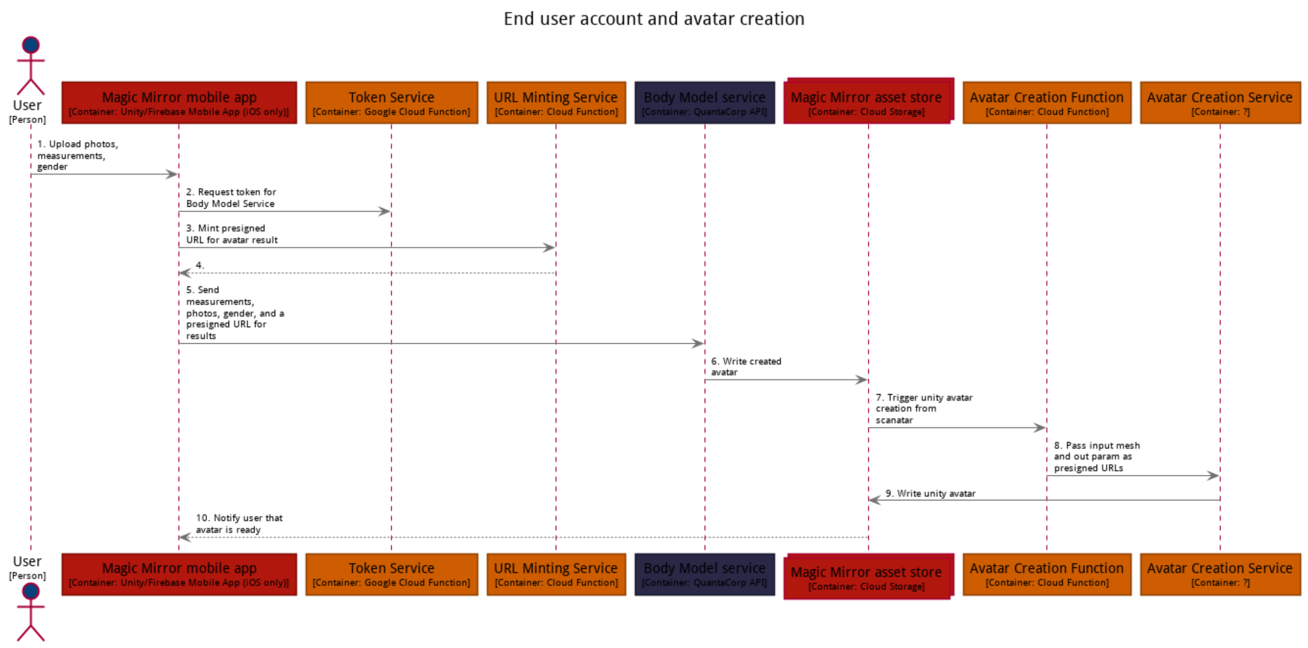
10.1.2 Use Case 1


Fit Model Scanatar Creation
Designer [Person]
Designer Config App [Container: Firebase Webapp (ReactJS)]
Body Model service [Container: QuantaCorp API]
VR Asset Store [Container: Firebase Storage]
Avatar Creation function [Container: Google Cloud Function]
Avatar Creation [Container: AWS??]
1. Upload photos, measurements, gender
2. (More detail in the Fit model scan diagram)
3. Write QC avatar
4. Trigger VStitcher avatar creation
5. Sending presigned URLs for qcAvatar, animations, metadata and to write results to
6. Convert avatar, apply animations
7. Write VStitcher-compatible avatar
8. Notify user that avatar is ready by email
Designer [Person]
Designer Config App [Container: Firebase Webapp (ReactJS)]
Body Model service [Container: QuantaCorp API]
VR Asset Store [Container: Firebase Storage]
Avatar Creation function [Container: Google Cloud Function]
Avatar Creation [Container: AWS??]

10.1.3 Use Case 2


Avatar creation flow
Influencer [Person]
DressMeUp mobile app [Container: Firebase mobile web app (ReactJS)]
URL Minting Service [Container: Cloud Function]
Token Service [Container: Google Cloud Function]
Body Model service [Container: QuantaCorp API]
DressMeUp Asset Store [Container: Firebase Storage]
Avatar Creation Function [Container: Cloud Function]
Avatar Creation [Container: AWS Step Function]
1. Upload photos, measurements, gender
2. Create presigned URL for avatar to be written to
3.
4. Request token for Body Model Service
5. Send measurements, photos, gender, and a presigned URL for results
6. Write avatar mesh
7. Trigger Metail avatar creation
8. Sending presigned URLs for params and out param for results
9. Write Metail avatar
10. Notify user that avatar is ready

10.1.4 Use Case 3


End user account and avatar creation
User [Person]
Magic Mirror mobile app [Container: Unity/Firebase Mobile App (iOS only)]
Token Service [Container: Google Cloud Function]
URL Minting Service [Container: Cloud Function]
Body Model service [Container: QuantaCorp API]
Magic Mirror asset store [Container: Cloud Storage]
Avatar Creation Function [Container: Cloud Function]
Avatar Creation Service [Container: ?]
1. Upload photos, measurements, gender
2. Request token for Body Model Service
3. Mint presigned URL for avatar result
4.
5. Send measurements, photos, gender, and a presigned URL for results
6. Write created avatar
7. Trigger unity avatar creation from scanatar
8. Pass input mesh and out param as presigned URLs
9. Write unity avatar
10. Notify user that avatar is ready

10.1.5 Invocation

The function is triggered by a Cloud Storage notification when a Quantacorp scan (PLY) and metadata (JSON) has been written to the asset store.

10.1.6 Input

A Google Cloud Storage Notification

10.1.7 Actions

1. Read metadata from asset store and extract subject height and gender
2. Construct presigned GET URL for scan
3. Construct presigned PUT URL for the output avatar
4. Retrieve AWS keys from secret store
5. Invoke the Avatar Creation Step Function

10.1.8 Output

This function is invoked asynchronously so produces no output. The generated avatar is written to the asset store using the presigned URL passed to the step function invocation.

10.2 Catalogue Update

10.2.1 Context

This function is required for Use Cases 2 & 3. It is automatically triggered once a day, to parse information from the Odlo web feed and updates the garment database in the Dress Me Up Data Store.

10.2.2 Invocation

The cloud function runs once per day, reads the updated CSV file and makes the necessary changes in the Google Firestore Database of the project.

10.2.3 Input

The input is a CSV file at a provided URL location.

10.2.4 Actions

1. The service triggers a function once a day that retrieves and parses the CSV file from the link.
2. It then uploads new entries and refreshes the status of the garments in the Firebase Datastore.

10.2.5 Output

- Success / Error code.
- Results are written to `garments` Data Store collection.

10.2.6 Repository

https://gitlab.com/etryon/shared/gcf-catalogue-updater

10.3 Consumer Ratings

10.3.1 Context

This function is required for Use Case 2. It is triggered by the DressMeUp app when a user creates a collection item, handling the call to the Consumer Rating Prediction Service, by providing the necessary information and then writing back the results to the Dress Me Up Data Store.

10.3.2 Invocation

The function is triggered by a Cloud Storage notification when a collection item entry has been written to the data store.

10.3.3 Input

The input is a JSON payload with the following field.

Field	Description
Array	An array of garment IDs

10.3.4 Actions

1. The service creates an array of `garment ids` based on a user's active collection.
2. The service sends the array to the Prediction Service and gets back a list of `garment ids` for a specific user.
3. The returned data is saved into the Dress Me Up Data Store.

10.3.5 Output

- Success / Error code.
- An object that features a `user id` and an array of `garment ids`.
- Results are written to `garment_suggestions` Data Store collection.
- Example return object:

```
{
  "user_id": "435",
  "garment_ids": [
        "43223432",
        "53246342342",
        "43635745234"
        ]
 }
```

10.3.6 Repository

https://gitlab.com/etryon/use-case-2/gcf-consumer-ratings

10.4 DressMeUp Composition Invoker

10.4.1 Context

This function is required for use case 2. It is invoked by the DressMeUp mobile app when a user tries on a garment, and is the interface between eTryOn systems and the Metail Composition Service.

10.4.2 Invocation

This function is invoked by the DressMeUp mobile app when a user selects a photo and garment for composition.

10.4.3 Input

The input is a JSON payload with the following fields.

Field	Description
Photo	Reference to a user submitted photo in the DressMeUp Asset Store
Garment	Reference to a garment in the DressMeUp Asset Store

10.4.4 Input Function JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "description": "The root schema comprises the entire JSON document.",
  "examples": [
    {
      "src": {
        "garment_id": "7611366000042",
        "user_media":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg"
      }
    }
  ],
  "required": [
    "src"
  ],
  "title": "The root schema",
  "type": "object",
  "properties": {
    "src": {
      "$id": "#/properties/src",
      "default": {},
      "description": "The source object features a garment unique id (EAN code) and
the url to Firebast Storage where a user submitted media item resides.",
      "examples": [
        {
          "garment_id": "7611366000042",
          "user_media":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg"
        }
      ],
      "required": [
        "garment_id",
        "user_media"
      ],
      "title": "The src object",
      "type": "object",
      "properties": {
        "garment_id": {
          "$id": "#/properties/src/properties/garment_id",
          "default": "",
          "description": "ODLO unique garment id (EAN code)",
          "examples": [
            "7611366000042"
          ],
          "title": "The garment_id property",
          "type": "string"
        },
        "user_media": {
          "$id": "#/properties/src/properties/user_media",
          "default": "",
          "description": "A url that points to Google Firebase Storage, where a
media item submitted by the user is stored",
          "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg"
```

```
          ],
          "title": "The user_media property",
          "type": "string"
        }
      }
    }
  }
}
```

10.4.5 Actions

1. Retrieve the user id of the caller from the authentication context
2. Locate the user's avatar in the DressMeUp Asset Store
3. Construct a presigned GET URL for the avatar
4. Construct a presigned GET URL for the photo
5. Construct a presigned GET URL for the garment browzwear file
6. Construct a presigned PUT URL for the output image
7. Invoke the Metail Composition Service step function.

10.4.6 Output

On success, returns null. On error throws a functions.https.HttpsError which will be returned to the client.

10.4.7 Implementation notes

This function should be implemented as an HTTPS callable function and called by the client using the Firebase SDK. This ensures that authentication tokens are automatically included in the request and validated by the server. The request body is also automatically deserialized.

The composition service currently involves some manual processing and is not guaranteed to return a result within 12 hours. This means the signed URL cannot be generated using the IAM signBlob method, as these URLs are only valid for a maximum of 12 hours.

10.5 Rating Retrieval

10.5.1 Context

This function is required for Use Case 1. It is triggered by the VR Designer config app when a user uploads a new garment or updates an existing one. It handles the call to the Consumer Rating Prediction Service, by providing the necessary information and then writing back the results to the VR Designer Data Store.

10.5.2 Invocation

The function is triggered by a Cloud Storage notification when a garment item has been written or updated in the data store.

10.5.3 Input

The input is a JSON payload with the following fields.

Field	Description
Photo	A downscaled garment photo in base64 encoding. The size must be of size 256 x 256
Metadata	Garment metadata
Market Segmentation Array	An array that features the current market segmentation properties like `age_target`, `color`, and `price`

10.5.4 Actions

1. The function sends a JSON payload that features a downscaled photos of a garment, the garment metadata, and an array with market segment details to the Consumer Rating Prediction Service.
2. It receives a result in the form of a percentage string.
3. The function then writes to the VR Data Store, an entry that features the garment ID, the market segment ID, and the performance score.

10.5.5 Output

- Success / Error code.
- An object that features a garment id, a segment id and a percentage result.
- Results are written to market_segments_results Data Store collection.
- Example return object:

```
 {
      "garment_id": "98237483247",
      "segment_id": "1236",
      "result": "95"
  }
```

10.5.6 Repository

https://gitlab.com/etryon/uc1/gcf-rating-retrieval

10.6 UC3 Unity Model Creation

An avatar creation cloud function is required for use case 3. It is the interface between the eTryOn systems and the Metail Garment Model Creation Service.

10.6.1 Invocation

The function is triggered by a Cloud Storage notification when a Browzwear file has been written to the asset store.

10.6.2 Input

A Google Cloud Storage Notification

10.6.3 Actions

1. Construct presigned GET URL for Browzwear file
2. Construct presigned PUT URL for the output OBI garment model
3. Retrieve AWS keys from secret store
4. Invoke the Garment Model Creation Step Function

10.6.4 Output

This function is invoked asynchronously so produces no output. The generated OBI garment is written to the asset store using the presigned URL passed to the step function invocation.

11 Appendix E - Firestore Database JSON Schemas from https://etryon.gitlab.io/techdocs/

11.1 Firestore Database JSON Schemas

In the following folders are the schemas for each use case data store.

Some useful information:

- Note that UC2 and UC3 share the same 'garments' db.

12.1.1 JSON Schema versioning

We use Semantic Versioning to safeguard backwards-compatibility between the new schema and existing data represented in earlier versions of the schema.

The filename follows the following pattern `MODEL-REVISION-ADDITION`, where you increment:

`MODEL` when you make a breaking schema change which will prevent interaction with any historical data,

`REVISION` when you make a schema change which may prevent interaction with some historical data,

`ADDITION` when you make a schema change that is compatible with all historical data.

11.2 UC1 Schemas

11.2.1 Animations Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc1/animations.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "name": "Runnning",
            "created": "1624521140042",
            "created_by": "egm9430mfg3403",
            "src": [
                {
                    "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/animation%public%filename.abc",
                    "photo":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/animation%public%filename.png"
                }
            ]
        }
    ],
    "required": [
        "name",
        "created",
        "created_by",
        "src"
    ],
    "properties": {
        "name": {
            "$id": "#/properties/name",
            "default": "",
            "description": "The animation name",
            "examples": [
                "Runnning",
                "Standing",
                "Sitting",
                "Walking"
            ],
            "title": "The name property",
            "type": "string"
        },
        "created": {
            "$id": "#/properties/created",
            "default": "",
            "description": "Timestamp in milliseconds, created when submitting an
animation",
            "examples": [
                "1624521140042",
                         "1519129853500"
            ],
            "title": "The created property",
            "type": "string"
        },
        "created_by": {
            "$id": "#/properties/created_by",
            "type": "string",
            "title": "The created_by object",
            "description": "It includes a user Id of the user that uploaded the
animation",
            "default": {},
            "examples": [
                "egm9430mfg3403"
            ]
        },
        "src": {
            "$id": "#/properties/src",
            "default": {},
            "description": "The src object contains a file and a photo property",
            "examples": [
                {
                    "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/animation%pu
blic%filename.fbx",
                    "photo": "https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/animation%public%filename.png"
                }
            ],
            "required": [
                "file",
                "photo"
            ],
            "title": "The src object",
            "type": "object",
            "properties": {
                "file": {
                    "$id": "#/properties/src/properties/file",
                    "default": "",
                    "description": "This is a URL that points to the animation file saved in Firebase Storage",
                    "examples": [
                        "https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/animation%public%filename.fbx"
                    ],
                    "title": "The file property",
                    "type": "string"
                },
                "photo": {
                    "$id": "#/properties/src/properties/photo",
                    "default": "",
                    "description": "A URL that points to a photo of the animation saved in Firebase Storage, for display purposes on the config app",
                    "examples": [
                        "https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/animation%public%filename.png"
                    ],
                    "title": "The photo property",
                    "type": "string"
                }
            }
        }
    }
```

```
}
```

11.2.2 Avatars Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc1/avatars.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "name": "Male 30s Caucasian",
            "created": "1624521140042",
            "created_by": "egm9430mfg3403",
            "src": {
                "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%publi
c%filename.fbx",
                "photo":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%publi
c%filename.jpg"
            }
        }
    ],
    "required": [
        "name",
        "created",
        "created_by",
        "src"
    ],
    "properties": {
        "name": {
            "$id": "#/properties/name",
            "default": "",
            "description": "The Avatar name",
            "examples": [
                "Male 30s Caucasian"
            ],
            "title": "The name property",
            "type": "string"
        },
        "created": {
            "$id": "#/properties/created",
            "default": "",
            "description": "Timestamp in milliseconds, created when submitting an
avatar",
            "examples": [
                "1624521140042"
            ],
            "title": "The created property",
            "type": "string"
        },
        "created_by": {
            "$id": "#/properties/created_by",
            "default": {},
            "description": "It includes a user Id of the user that uploaded the
avatar",
            "examples": [
               "egm9430mfg3403"
            ],
                   "type": "string"
        },
        "src": {
            "$id": "#/properties/src",
            "default": {},
            "description": "The src object contains a file and a photo property.",
            "examples": [
                {
                    "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%publi
c%filename.fbx",
                    "photo":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%publi
c%filename.png"
                }
            ],
            "required": [
                "file",
                "photo"
            ],
            "title": "The src object",
            "type": "object",
            "properties": {
                "file": {
                    "$id": "#/properties/src/properties/file",
                    "default": "",
                    "description": "This is a URL that points to the avatar file saved in Firebase Storage",
                    "examples": [
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%public%filename.fbx"
                    ],
                    "title": "The file property",
                    "type": "string"
                },
                "photo": {
                    "$id": "#/properties/src/properties/photo",
                    "default": "",
                    "description": "A URL that points to a photo of the avatar saved in Firebase Storage, for display purposes on the config app",
                    "examples": [
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%public%filename.png"
                    ],
                    "title": "The photo property",
                    "type": "string"
                }
            }
        }
    }
}
```

11.2.3 Garments Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc1/garments.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "uid": "98237483247",
            "name": "Pencil Skirt",
            "collection": [
                "F/W2020",
                "SPRING2021"
            ],
            "photo":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shared%filename.png",
            "category": "Bottoms",
            "type": "Skirt",
            "style": "Pencil",
            "src": {
                "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shared%filename.fbx",
                "texture":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/textures%shared%filename.zip",
                "meta": {
                    "colors": [
                        "Light Grey",
                        "Black",
                        "Red"
                    ]
                }
            },
            "market_segment_ids": [
                "3",
                "46",
                "54"
            ],
            "created": "1624521140042",
            "created_by": "egm9430mfg3403"
        }
    ],
    "required": [
        "uid",
        "name",
        "collection",
        "photo",
        "category",
        "type",
        "style",
        "src",
        "market_segment_ids",
        "created",
        "created_by"
    ],
    "properties": {
        "uid": {
            "$id": "#/properties/uid",
            "default": "",
            "description": "A unique id for each garment configuration",
            "examples": [
                "98237483247"
            ],
            "title": "The uid property",
            "type": "string"
        },
        "name": {
            "$id": "#/properties/name",
            "default": "",
            "description": "The garment name",
            "examples": [
                "Pencil Skirt"
            ],
            "title": "The name property",
            "type": "string"
        },
        "collection": {
            "$id": "#/properties/collection",
            "default": [],
            "description": "An array of strings, denoting one or multiple fashion
collections that the garment is included in.",
            "examples": [
                [
                    "F/W2020",
                    "SPRING2021"
                ]
            ],
            "title": "The collection array",
            "type": "array",
            "items": {
                "$id": "#/properties/collection/items",
                "anyOf": [
                    {
                        "$id": "#/properties/collection/items/anyOf/0",
                        "default": "",
                        "description": "A string, denoting a fashion collection that
the garment is included in.",
                        "examples": [
                            "F/W2020",
                            "SPRING2021"
                        ],
                        "title": "The first anyOf property",
                        "type": "string"
                    }
                ]
            }
        },
        "photo": {
            "$id": "#/properties/photo",
            "default": "",
            "description": "A URL of a photo of the specific garment that is saved
in Firebase Storage, under the UC1 subfolder",
            "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.png"
            ],
            "title": "The photo property",
            "type": "string"
        },
        "category": {
            "$id": "#/properties/category",
            "default": "",
            "description": "The category the garment belongs to",
            "examples": [
                "Bottoms"
            ],
            "title": "The category property",
            "type": "string"
        },
        "type": {
            "$id": "#/properties/type",
            "default": "",
            "description": "The type of garment",
            "examples": [
                "Skirt"
            ],
            "title": "The type property",
            "type": "string"
        },
        "style": {
            "$id": "#/properties/style",
            "default": "",
            "description": "The garment style",
            "examples": [
                "Pencil"
            ],
            "title": "The style property",
            "type": "string"
        },
        "src": {
            "$id": "#/properties/src",
            "type": "object",
            "title": "The src schema",
            "description": "Inside the src object, there is a specific garment file,
a texture zip and available colors. It includes a URL to Firebase Storage for the
garment file, a URL for the textures zip in Firebase Storage and an array that
features all colors available",
            "default": {},
            "examples": [
                {
                    "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.fbx",
                    "texture":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/textures%sha
red%filename.zip",
                    "meta": {
                        "colors": [
                            "Light Grey",
                            "Black",
                            "Red"
                        ]
                    }
                }
            ],
            "required": [
                "file",
                "texture",
                "meta"
            ],
            "properties": {
                "file": {
                    "$id": "#/properties/src/properties/file",
                    "type": "string",
                    "title": "The file property",
                    "description": "A Firebase Storage url that features the 3d
model of the garment",
                    "default": "",
                    "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.fbx"
                    ]
                },
                "texture": {
                    "$id": "#/properties/src/properties/texture",
                    "type": "string",
                    "title": "The texture property",
                    "description": "A Firebase Storage url that features the zip
file for the textures of the garment",
                    "default": "",
                    "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/textures%sha
red%filename.zip"
                    ]
                },
                "meta": {
                    "$id": "#/properties/src/properties/meta",
                    "type": "object",
                    "title": "The meta object",
                    "description": "The meta object contains metadata about the
garment",
                    "default": {},
                    "examples": [
                        {
                            "colors": [
                                "Light Grey",
                                "Black",
                                "Red"
                            ]
                        }
                    ],
                    "required": [
                        "colors"
                    ],
                    "properties": {
                        "colors": {
                            "$id":
"#/properties/src/properties/meta/properties/colors",
                            "type": "array",
                            "title": "The colors array",
                            "description": "An array that features the colors
available for the garment",
                            "default": [],
                            "examples": [
                                [
                                    "Light Grey",
                                    "Black"
                                ]
                            ],
                            "items": {
                                "$id":
"#/properties/src/properties/meta/properties/colors/items",
                                "anyOf": [
                                    {
                                        "$id":
"#/properties/src/properties/meta/properties/colors/items/anyOf/0",
                                        "type": "string",
                                        "title": "The first anyOf property",
                                        "description": "A color label string",
                                        "default": "",
                                        "examples": [
                                            "Light Grey",
                                            "Black"
                                        ]
                                    }
                                ]
                            }
                        }
                    }
                }
            }
        },
        "market_segment_ids": {
            "$id": "#/properties/market_segment_ids",
            "default": [],
            "description": "It contains market segment Ids that the garment is
associated with",
            "examples": [
                [
                    "3",
                    "46"
                ]
            ],
            "title": "The market_segment_ids array",
            "type": "array",
            "additionalItems": false,
            "items": {
                "$id": "#/properties/market_segment_ids/items",
                "anyOf": [
                    {
                        "$id": "#/properties/market_segment_ids/items/anyOf/0",
                        "default": "",
                        "description": "A market segment id string",
                        "examples": [
                            "3",
                            "46"
                        ],
                        "title": "The first anyOf property",
                        "type": "string"
                    }
                ]
            }
        },
        "created": {
            "$id": "#/properties/created",
            "default": "",
            "description": "Timestamp in milliseconds, created when submitting a garment entry",
            "examples": [
                "1624521140042"
            ],
            "title": "The created property",
            "type": "string"
        },
        "created_by": {
            "$id": "#/properties/created_by",
            "type": "string",
            "title": "The created_by property",
            "description": "An Id of the user that uploaded the specific garment",
            "default": {},
            "examples": ["egm9430mfg3403"]
        }
```

```
    }
}
```

11.2.4 Market Segments Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc1/market_segments.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "name": "Segment 1",
            "created": "1624521140042",
            "age_target": "18-24",
            "price": "200",
            "color": "white"
        }
    ],
    "required": [
        "name",
        "created",
        "age_target",
        "price",
        "color"
    ],
    "properties": {
        "name": {
            "$id": "#/properties/name",
            "default": "",
            "description": "The Market Segment name.",
            "examples": [
                "Segment 1"
            ],
            "title": "The name poperty",
            "type": "string"
        },
        "created": {
            "$id": "#/properties/created",
            "default": "",
            "description": "Timestamp in milliseconds, created when submitting a
market segment entry",
            "examples": [
                "1624521140042"
            ],
            "title": "The created property",
            "type": "string"
        },
        "age_target": {
            "$id": "#/properties/age_target",
            "default": "",
            "description": "The age target property",
            "examples": [
                "18-24"
            ],
            "title": "The age_target property",
            "type": "string"
        },
        "price": {
            "$id": "#/properties/price",
            "default": "",
            "description": "The price property",
            "examples": [
                "200"
            ],
            "title": "The price property",
            "type": "string"
        },
        "color": {
            "$id": "#/properties/color",
            "default": "",
            "description": "The color property",
            "examples": [
                "white"
            ],
            "title": "The color property",
            "type": "string"
        }
    }
}
```

11.2.5 Market Segments Results Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc1/market_segments.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "name": "Segment 1",
            "created": "1624521140042",
            "age_target": "18-24",
            "price": "200",
            "color": "white"
        }
    ],
    "required": [
        "name",
        "created",
        "age_target",
        "price",
        "color"
    ],
    "properties": {
        "name": {
            "$id": "#/properties/name",
            "default": "",
            "description": "The Market Segment name.",
            "examples": [
                "Segment 1"
            ],
            "title": "The name poperty",
            "type": "string"
        },
        "created": {
            "$id": "#/properties/created",
            "default": "",
            "description": "Timestamp in milliseconds, created when submitting a market segment entry",
            "examples": [
                "1624521140042"
            ],
            "title": "The created property",
            "type": "string"
        },
        "age_target": {
            "$id": "#/properties/age_target",
            "default": "",
            "description": "The age target property",
            "examples": [
                "18-24"
            ],
            "title": "The age_target property",
            "type": "string"
        },
        "price": {
            "$id": "#/properties/price",
            "default": "",
            "description": "The price property",
            "examples": [
                "200"
            ],
            "title": "The price property",
            "type": "string"
        },
        "color": {
            "$id": "#/properties/color",
            "default": "",
            "description": "The color property",
            "examples": [
                "white"
            ],
            "title": "The color property",
            "type": "string"
        }
    }
}
```

11.2.6 User Info Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc1/user_info.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "name": "John Doe",
            "role": "designer",
            "default_segment": "1236"
        }
    ],
    "required": [
        "name",
        "role",
        "default_segment"
    ],
    "properties": {
        "name": {
            "$id": "#/properties/name",
            "default": "",
            "description": "The provided Full name of the user",
            "examples": [
                "John Doe"
            ],
            "title": "The name property",
            "type": "string"
        },
        "role": {
            "$id": "#/properties/role",
            "default": "",
            "description": "The user role",
            "examples": [
                "user",
                "designer",
                "manager"
            ],
            "title": "The role property",
            "type": "string"
        },
        "default_segment": {
            "$id": "#/properties/default_segment",
            "default": "",
            "description": "If user sets a preferred Market Segment, then it is set
here as an Id. When creating a new garment entry, this default Segment will be
pre-set",
            "examples": [
                "1236"
            ],
            "title": "The default_segment property",
            "type": "string"
        }
    }
}
```

11.3 UC2 Schemas

11.3.1 Collection Items Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc2/collection_items.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "name": "My cool outfit",
            "created": "1519211809934",
            "status": "pending",
            "src": {
                "garment_id": "7611366000042",
                "user_media": 
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg"
            },
            "output": 
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/output%user%
enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg",
            "type": "photo"
        }
    ],
    "required": [
        "name",
        "created",
        "status",
        "src",
        "output",
        "type"
    ],


    "properties": {
        "name": {
            "$id": "#/properties/name",
            "default": "",
            "description": "A user generated title for the collection item",
            "examples": [
                "My cool outfit"
            ],
            "title": "The name property",
            "type": "string"
        },
        "created": {
            "$id": "#/properties/created",
            "default": "",
            "description": "Timestamp in milliseconds, created when submitting a
collection item",
            "examples": [
                "1519211809934"
            ],
            "title": "The created property",
            "type": "string"
        },
        "status": {
            "$id": "#/properties/status",
            "default": "pending",
            "description": "The status of a collection item. Default is 'pending'.
Changes to 'ready' when output file is submitted.",
            "examples": [
                "pending",
                "ready"
            ],
            "title": "The status property",
            "type": "string"
        },
        "src": {
            "$id": "#/properties/src",
            "default": {},
            "description": "The source object features a garment unique id (EAN
code) and the url to Firebast Storage where a user submitted media item resides.",
            "examples": [
                {
                    "garment_id": "7611366000042",
                    "user_media":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg"
                }
            ],
            "required": [
                "garment_id",
                "user_media"
            ],
            "title": "The src object",
            "type": "object",
            "properties": {
                "garment_id": {
                    "$id": "#/properties/src/properties/garment_id",
                    "default": "",
                    "description": "ODLO unique garment id (EAN code)",
                    "examples": [
                        "7611366000042"
                    ],
                    "title": "The garment_id property",
                    "type": "string"
                },
                "user_media": {
                    "$id": "#/properties/src/properties/user_media",
                    "default": "",
                    "description": "A url that points to Google Firebase Storage, 
where a media item submitted by the user is stored",
                    "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg"
                    ],
                    "title": "The user_media property",
                    "type": "string"
                }
            }
        },
        "output": {
            "$id": "#/properties/output",
            "default": "",
            "description": "A url that points to Google Firebase Storage, where the 
resulting image or video created by the service is stored.",
            "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/output%user%
enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.jpg"
            ],
            "title": "The output property",
            "type": "string"
        },
        "type": {
            "$id": "#/properties/type",
            "default": "photo",
            "description": "The type of a collection item. Can either be photo or video",
            "examples": [
                "photo",
                "video"
            ],
            "title": "The type property",
            "type": "string"
        }
    }
}
```

11.3.2 Garment Suggestions Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc2/garment_suggestions.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "user_id": "435",
            "garment_ids": [
                "43223432",
                "53246342342",
                "43635745234"
            ]
        }
    ],
    "required": [
        "user_id",
        "garment_ids"
    ],
    "properties": {
        "user_id": {
            "$id": "#/properties/user_id",
            "default": "",
            "description": "The user Id",
            "examples": [
                "435"
            ],
            "title": "The user_id property",
            "type": "string"
        },
        "garment_ids": {
            "$id": "#/properties/garment_ids",
            "default": [],
            "description": "An array that features ids for the garment suggestions of each user.",
            "examples": [
                [
                    "43223432",
                    "53246342342"
                ]
            ],
            "title": "The garment_ids array",
            "type": "array",
            "additionalItems": false,
            "items": {
                "$id": "#/properties/garment_ids/items",
                "anyOf": [
                    {
                        "$id": "#/properties/garment_ids/items/anyOf/0",
                        "default": "",
                        "description": "A garment id",
                        "examples": [
                            "43223432",
                            "53246342342"
                        ],
                        "title": "The first anyOf property",
                        "type": "string"
                    }
                ]
            }
        }
    }
}
```

11.3.3 Garments Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc2/garments.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "title": "REVOLUTION LIGHT langärmeliges Baselayer Shirt",
            "eshop_link": "https://www.odlo.com/7611366000042.html",
            "uid": "7611366000042",
            "group_id": "391772",
            "material": "100% Polyester",
            "product_type": {
                "full_type": "Men>Sporting Activity>Running & Trail",
                "extracted_type_0": "Men",
                "extracted_type_1": "Sporting Activity",
                "extracted_type_2": "Running & Trail"
            },
            "price": "64.95",
            "additional_information": {
                "custom_label_0": "",
                "custom_label_1": "Warm",
                "custom_label_2": "",
                "custom_label_3": "oekotex"
            },
            "gender": "male",
            "size": "M",
            "color": "directoire blue",
            "photo":
"https://click.cptrack.de/?rd=true&k=rXK5MERxr0OxIUPXDyoj7-2y22xETFU22pNdMFGuQXDUbYjHA_BLt09OLT8cF4k6v9egB14mTsTo9GB4eBPQwQ~~&rdlink=https%3A%2F%2Fwww.odlo.com%2Fon%2Fdemandware.static%2F-%2FSites-odlo-master-catalog%2Fdefault%2Fdw3895ef55%2Fimages%2Flarge%2F211874.jpg",
            "src": {
                "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.bw"
            }
        }
    ],
    "required": [
        "title",
        "eshop_link",
        "uid",
        "group_id",
        "material",
        "product_type",
        "price",
        "additional_information",
        "gender",
        "size",
        "color",
        "photo",
        "src"
    ],
    "properties": {
        "title": {
            "$id": "#/properties/title",
            "default": "",
            "description": "The garment entry title",
            "examples": [
                "REVOLUTION LIGHT langärmeliges Baselayer Shirt"
            ],
            "title": "The title property",
            "type": "string"
        },
        "eshop_link": {
            "$id": "#/properties/eshop_link",
            "default": "",
            "description": "A URL that links to the specific garment in the ODLO
e-shop",
            "examples": [
                "https://www.odlo.com/7611366000042.html"
            ],
            "title": "The eshop_link property",
            "type": "string"
        },
        "uid": {
            "$id": "#/properties/uid",
            "default": "",
            "description": "Each garment has a unique ID (EAN code)",
            "examples": [
                "7611366000042"
            ],
            "title": "The uid property",
            "type": "string"
        },
        "group_id": {
            "$id": "#/properties/group_id",
            "default": "",
            "description": "Each garment irregardless of size and color, has a
unique id which is the group id property.",
            "examples": [
                "391772"
            ],
            "title": "The group_id property",
            "type": "string"
        },
        "material": {
            "$id": "#/properties/material",
            "default": "",
            "description": "The garment material",
            "examples": [
                "100% Polyester"
            ],
            "title": "The material property",
            "type": "string"
        },
        "product_type": {
            "$id": "#/properties/product_type",
            "default": {},
            "description": "This is provided by ODLO as a full type. We further
break it down into 3 distinct types, for advanced filtering.",
            "examples": [
                {
                    "full_type": "Men>Sporting Activity>Running & Trail",
                    "extracted_type_0": "Men",
                    "extracted_type_1": "Sporting Activity",
                    "extracted_type_2": "Running & Trail"
                }
            ],
            "required": [
                "full_type",
                "extracted_type_0",
                "extracted_type_1",
                "extracted_type_2"
            ],
            "title": "The product_type object",
            "type": "object",
            "properties": {
                "full_type": {
                    "$id": "#/properties/product_type/properties/full_type",
                    "default": "",
                    "description": "the full type property of a garment. It is a
long string consisted of some tags",
                    "examples": [
                        "Men>Sporting Activity>Running & Trail"
                    ],
                    "title": "The full_type property",
                    "type": "string"
                },
                "extracted_type_0": {
                    "$id": "#/properties/product_type/properties/extracted_type_0",
                    "default": "",
                    "description": "The first type term extracted from full type
string",
                    "examples": [
                        "Men"
                    ],
                    "title": "The extracted_type_0 property",
                    "type": "string"
                },
                "extracted_type_1": {
                    "$id": "#/properties/product_type/properties/extracted_type_1",
                    "default": "",
                    "description": "The seconds type term extracted from full type
string",
                    "examples": [
                        "Sporting Activity"
                    ],
                    "title": "The extracted_type_1 property",
                    "type": "string"
                },
                "extracted_type_2": {
                    "$id": "#/properties/product_type/properties/extracted_type_2",
                    "default": "",
                    "description": "The third type term extracted from full type
string",
                    "examples": [
                        "Running & Trail"
                    ],
                    "title": "The extracted_type_2 property",
                    "type": "string"
                }
            }
        },
        "price": {
            "$id": "#/properties/price",
            "default": "",
            "description": "The garment price",
            "examples": [
                "64.95"
            ],
            "title": "The price property",
            "type": "string"
        },
        "additional_information": {
            "$id": "#/properties/additional_information",
            "default": {},
            "description": "This object consists of four different fields that
contain information that further describe the garment",
            "examples": [
                {
                    "custom_label_0": "",
                    "custom_label_1": "Warm",
                    "custom_label_2": "",
                    "custom_label_3": "oekotex"
                }
            ],
            "required": [
                "custom_label_0",
                "custom_label_1",
                "custom_label_2",
                "custom_label_3"
            ],
            "title": "The additional_information object",
            "type": "object",
            "properties": {
                "custom_label_0": {
                    "$id":
"#/properties/additional_information/properties/custom_label_0",
                    "default": "",
                    "description": "The first field that further describes the
garment item",
                    "examples": [
                        ""
                    ],
                    "title": "The custom_label_0 property",
                    "type": "string"
                },
                "custom_label_1": {
                    "$id":
"#/properties/additional_information/properties/custom_label_1",
                    "default": "",
                    "description": "The second field that further describes the
garment item",
                    "examples": [
                        "Warm"
                    ],
                    "title": "The custom_label_1 property",
                    "type": "string"
                },
                "custom_label_2": {
                    "$id":
"#/properties/additional_information/properties/custom_label_2",
                    "default": "",
                    "description": "The third field that further describes the
garment item",
                    "examples": [
                        ""
                    ],
                    "title": "The custom_label_2 property",
                    "type": "string"
                },
                "custom_label_3": {
                    "$id":
"#/properties/additional_information/properties/custom_label_3",
                    "default": "",
                    "description": "The fourth field that further describes the
garment item",
                    "examples": [
                        "oekotex"
                    ],
                    "title": "The custom_label_3 property",
                    "type": "string"
                }
            }
        },
        "gender": {
            "$id": "#/properties/gender",
            "default": "",
            "description": "This is male or female, for the fit to tailor male or
female bodies",
            "examples": [
                "male",
                "female"
            ],
            "title": "The gender property",
            "type": "string"
        },
        "size": {
            "$id": "#/properties/size",
            "default": "",
            "description": "The garment size property 'Small' to 'XXL'",
            "examples": [
                "S",
                "M",
                "L",
                "XL",
                "XXL"
            ],
            "title": "The size property",
            "type": "string"
        },
        "color": {
            "$id": "#/properties/color",
            "default": "",
            "description": "The garment color",
            "examples": [
                "directoire blue"
            ],
            "title": "The color property",
            "type": "string"
        },
        "photo": {
            "$id": "#/properties/photo",
            "default": "",
            "description": "A URL of the garment photo. The photo is hosted on ODLO server",
            "examples": [
"https://click.cptrack.de/?rd=true&k=rXK5MERxr0OxIUPXDyoj7-2y22xETFU22pNdMFGuQXDUbYjHA_BLt09OLT8cF4k6v9egB14mTsTo9GB4eBPQwQ~~&rdlink=https%3A%2F%2Fwww.odlo.com%2Fon%2Fdemandware.static%2F-%2FSites-odlo-master-catalog%2Fdefault%2Fdw3895ef55%2Fimages%2Flarge%2F211874.jpg"
            ],
            "title": "The photo property",
            "type": "string"
        },
        "src": {
            "$id": "#/properties/src",
            "default": {},
            "description": "This object contains the Browzwear file along with other
files that might be needed in the future",
            "examples": [
                {
                    "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.bw"
                }
            ],
            "required": [
                "file"
            ],
            "title": "The src object",
            "type": "object",
            "properties": {
                "file": {
                    "$id": "#/properties/src/properties/file",
                    "default": "",
                    "description": "A URL pointing to the file that is saved in
Firebase Storage",
                    "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.bw"
                    ],
                    "title": "The file property",
                    "type": "string"
                }
            }
        }
    }
}
```

11.3.4 User Info Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc2/user_info.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "gender": "male",
            "size": "L",
                   "avatar":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%user%
enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.fbx"
        }
    ],
    "required": [
        "gender",
        "size",
             "avatar"
    ],
    "properties": {
        "gender": {
            "$id": "#/properties/gender",
            "default": "",
            "description": "User set gender, to be used on filtering garments",
            "examples": [
                "male",
                "female"
            ],
            "title": "The gender property",
            "type": "string"
        },
        "size": {
            "$id": "#/properties/size",
            "default": "",
            "description": "User size, to be used when selecting a garment",
            "examples": [
                "S",
                "M",
                "L",
                "XL",
                "XXL"
            ],
            "title": "The size property",
            "type": "string"
        },
         "avatar": {
            "$id": "#/properties/avatar",
            "default": "",
            "description": "User avatar file in the form of a URL, from QC API, it
is saved in Firebase Storage",
            "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%user%
enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.fbx"
            ],
            "title": "The avatar property",
            "type": "string"
        }
    }
}
```

11.4 UC3 Schemas

11.4.1 Garments Schema

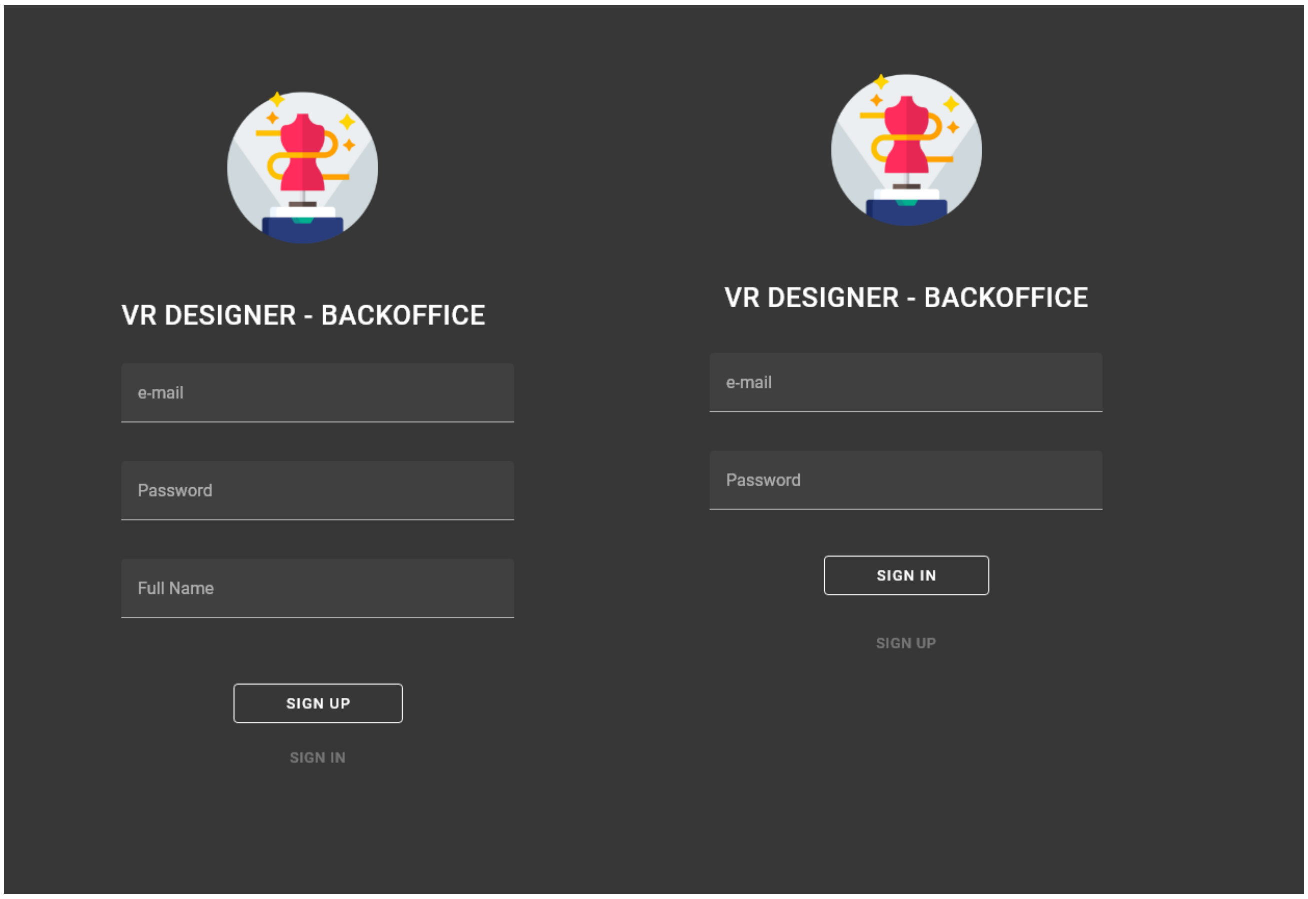
```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc3/garments.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "title": "REVOLUTION LIGHT langärmeliges Baselayer Shirt",
            "eshop_link": "https://www.odlo.com/7611366000042.html",
            "uid": "7611366000042",
            "group_id": "391772",
            "material": "100% Polyester",
            "product_type": {
                "full_type": "Men>Sporting Activity>Running & Trail",
                "extracted_type_0": "Men",
                "extracted_type_1": "Sporting Activity",
                "extracted_type_2": "Running & Trail"
            },
            "price": "64.95",
            "additional_information": {
                "custom_label_0": "",
                "custom_label_1": "Warm",
                "custom_label_2": "",
                "custom_label_3": "oekotex"
            },
            "gender": "male",
            "size": "M",
            "color": "directoire blue",
            "photo":
"https://click.cptrack.de/?rd=true&k=rXK5MERxr0OxIUPXDyoj7-2y22xETFU22pNdMFGuQXDUbYj
HA_BLt09OLT8cF4k6v9egB14mTsTo9GB4eBPQwQ~~&rdlink=https%3A%2F%2Fwww.odlo.com%2Fon%2Fd
emandware.static%2F-%2FSites-odlo-master-catalog%2Fdefault%2Fdw3895ef55%2Fimages%2Fl
arge%2F211874.jpg",
            "src": {
                "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.fbx",
                "texture":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/textures%sha
red%filename.zip",
                "meta": {
                    "colors": [
                        "Light Grey",
                        "Black",
                        "Red"
                    ]
                }
            }
        }
    ],
    "required": [
        "title",
        "eshop_link",
        "uid",
        "group_id",
        "material",
        "product_type",
        "price",
        "additional_information",
        "gender",
        "size",
        "color",
        "photo",
        "src"
    ],
    "properties": {
        "title": {
            "$id": "#/properties/title",
            "default": "",
            "description": "The garment entry title",
            "examples": [
                "REVOLUTION LIGHT langärmeliges Baselayer Shirt"
            ],
            "title": "The title property",
            "type": "string"
        },
        "eshop_link": {
            "$id": "#/properties/eshop_link",
            "default": "",
            "description": "A URL that links to the specific garment in the ODLO 
e-shop",
            "examples": [
                "https://www.odlo.com/7611366000042.html"
            ],
            "title": "The eshop_link property",
            "type": "string"
        },
        "uid": {
            "$id": "#/properties/uid",
            "default": "",
            "description": "Each garment has a unique ID (EAN code)",
            "examples": [
                "7611366000042"
            ],
            "title": "The uid property",
            "type": "string"
        },
        "group_id": {
            "$id": "#/properties/group_id",
            "default": "",
            "description": "Each garment irregardless of size and color, has a
unique id which is the group id property.",
            "examples": [
                "391772"
            ],
            "title": "The group_id property",
            "type": "string"
        },
        "material": {
            "$id": "#/properties/material",
            "default": "",
            "description": "The garment material",
            "examples": [
                "100% Polyester"
            ],
            "title": "The material property",
            "type": "string"
        },
        "product_type": {
            "$id": "#/properties/product_type",
            "default": {},
            "description": "This is provided by ODLO as a full type. We further
break it down into 3 distinct types, for advanced filtering.",
            "examples": [
                {
                    "full_type": "Men>Sporting Activity>Running & Trail",
                    "extracted_type_0": "Men",
                    "extracted_type_1": "Sporting Activity",
                    "extracted_type_2": "Running & Trail"
                }
            ],
            "required": [
                "full_type",
                "extracted_type_0",
                "extracted_type_1",
                "extracted_type_2"
            ],
            "title": "The product_type object",
            "type": "object",
            "properties": {
                "full_type": {
                    "$id": "#/properties/product_type/properties/full_type",
                    "default": "",
                    "description": "the full type property of a garment. It is a
long string consisted of some tags",
                    "examples": [
                        "Men>Sporting Activity>Running & Trail"
                    ],
                    "title": "The full_type property",
                    "type": "string"
                },
                "extracted_type_0": {
                    "$id": "#/properties/product_type/properties/extracted_type_0",
                    "default": "",
                    "description": "The first type term extracted from full type
string",
                    "examples": [
                        "Men"
                    ],
                    "title": "The extracted_type_0 property",
                    "type": "string"
                },
                "extracted_type_1": {
                    "$id": "#/properties/product_type/properties/extracted_type_1",
                    "default": "",
                    "description": "The seconds type term extracted from full type
string",
                    "examples": [
                        "Sporting Activity"
                    ],
                    "title": "The extracted_type_1 property",
                    "type": "string"
                },
                "extracted_type_2": {
                    "$id": "#/properties/product_type/properties/extracted_type_2",
                    "default": "",
                    "description": "The third type term extracted from full type
string",
                    "examples": [
                        "Running & Trail"
                    ],
                    "title": "The extracted_type_2 property",
                    "type": "string"
                }
            }
        },
        "price": {
            "$id": "#/properties/price",
            "default": "",
            "description": "The garment price",
            "examples": [
                "64.95"
            ],
            "title": "The price property",
            "type": "string"
        },
        "additional_information": {
            "$id": "#/properties/additional_information",
            "default": {},
            "description": "This object consists of four different fields that
contain information that further describe the garment",
            "examples": [
                {
                    "custom_label_0": "",
                    "custom_label_1": "Warm",
                    "custom_label_2": "",
                    "custom_label_3": "oekotex"
                }
            ],
            "required": [
                "custom_label_0",
                "custom_label_1",
                "custom_label_2",
                "custom_label_3"
            ],
            "title": "The additional_information object",
            "type": "object",
            "properties": {
                "custom_label_0": {
                    "$id":
"#/properties/additional_information/properties/custom_label_0",
                    "default": "",
                    "description": "The first field that further describes the
garment item",
                    "examples": [
                        ""
                    ],
                    "title": "The custom_label_0 property",
                    "type": "string"
                },
                "custom_label_1": {
                    "$id":
"#/properties/additional_information/properties/custom_label_1",
                    "default": "",
                    "description": "The second field that further describes the
garment item",
                    "examples": [
                        "Warm"
                    ],
                    "title": "The custom_label_1 property",
                    "type": "string"
                },
                "custom_label_2": {
                    "$id":
"#/properties/additional_information/properties/custom_label_2",
                    "default": "",
                    "description": "The third field that further describes the
garment item",
                    "examples": [
                        ""
                    ],
                    "title": "The custom_label_2 property",
                    "type": "string"
                },
                "custom_label_3": {
                    "$id":
"#/properties/additional_information/properties/custom_label_3",
                    "default": "",
                    "description": "The fourth field that further describes the
garment item",
                    "examples": [
                        "oekotex"
                    ],
                    "title": "The custom_label_3 property",
                    "type": "string"
                }
            }
        },
        "gender": {
            "$id": "#/properties/gender",
            "default": "",
            "description": "This is male or female, for the fit to tailor male or
female bodies",
            "examples": [
                "male",
                "female"
            ],
            "title": "The gender property",
            "type": "string"
        },
        "size": {
            "$id": "#/properties/size",
            "default": "",
            "description": "The garment size property 'Small' to 'XXL'",
            "examples": [
                "S",
                "M",
                "L",
                "XL",
                "XXL"
            ],
            "title": "The size property",
            "type": "string"
        },
        "color": {
            "$id": "#/properties/color",
            "default": "",
            "description": "The garment color",
            "examples": [
                "directoire blue"
            ],
            "title": "The color property",
            "type": "string"
        },
        "photo": {
            "$id": "#/properties/photo",
            "default": "",
            "description": "A URL of the garment photo. The photo is hosted on ODLO
server",
            "examples": [

"https://click.cptrack.de/?rd=true&k=rXK5MERxr0OxIUPXDyoj7-2y22xETFU22pNdMFGuQXDUbYj
HA_BLt09OLT8cF4k6v9egB14mTsTo9GB4eBPQwQ~~&rdlink=https%3A%2F%2Fwww.odlo.com%2Fon%2Fd
emandware.static%2F-%2FSites-odlo-master-catalog%2Fdefault%2Fdw3895ef55%2Fimages%2Fl
arge%2F211874.jpg"
            ],
            "title": "The photo property",
            "type": "string"
        },
        "src": {
            "$id": "#/properties/src",
            "type": "object",
            "title": "The src schema",
            "description": "Inside the src object, there is a specific garment file,
a texture zip and available colors. It includes a URL to Firebase Storage for the
garment file, a URL for the textures zip in Firebase Storage and an array that
features all colors available",
            "default": {},
            "examples": [
                {
                    "file":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.fbx",
                    "texture":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/textures%sha
red%filename.zip",
                    "meta": {
                        "colors": [
                            "Light Grey",
                            "Black",
                            "Red"
                        ]
                    }
                }
            ],
            "required": [
                "file",
                "texture",
                "meta"
            ],
            "properties": {
                "file": {
                    "$id": "#/properties/src/properties/file",
                    "type": "string",
                    "title": "The file property",
                    "description": "A Firebase Storage url that features the 3d
model of the garment",
                    "default": "",
                    "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/garment%shar
ed%filename.fbx"
                    ]
                },
                "texture": {
                    "$id": "#/properties/src/properties/texture",
                    "type": "string",
                    "title": "The texture property",
                    "description": "A Firebase Storage url that features the zip
file for the textures of the garment",
                    "default": "",
                    "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/textures%sha
red%filename.zip"
                    ]
                },
                "meta": {
                    "$id": "#/properties/src/properties/meta",
                    "type": "object",
                    "title": "The meta object",
                    "description": "The meta object contains metadata about the
garment",
                    "default": {},
                    "examples": [
                        {
                            "colors": [
                                "Light Grey",
                                "Black",
                                "Red"
                            ]
                        }
                    ],
                    "required": [
                        "colors"
                    ],
                    "properties": {
                        "colors": {
                            "$id":
"#/properties/src/properties/meta/properties/colors",
                            "type": "array",
                            "title": "The colors array",
                            "description": "An array that features the colors
available for the garment",
                            "default": [],
                            "examples": [
                                [
                                    "Light Grey",
                                    "Black"
                                ]
                            ],
                            "items": {
                                "$id":
"#/properties/src/properties/meta/properties/colors/items",
                                "anyOf": [
                                    {
                                        "$id":
"#/properties/src/properties/meta/properties/colors/items/anyOf/0",
                                        "type": "string",
                                        "title": "The first anyOf property",
                                        "description": "A color label string",
                                        "default": "",
                                        "examples": [
                                            "Light Grey",
                                            "Black"
                                        ]
                                    }
                                ]
                            }
                        }
                    }
                }
            }
        }
    }
}
```

11.4.2 User Info Schema

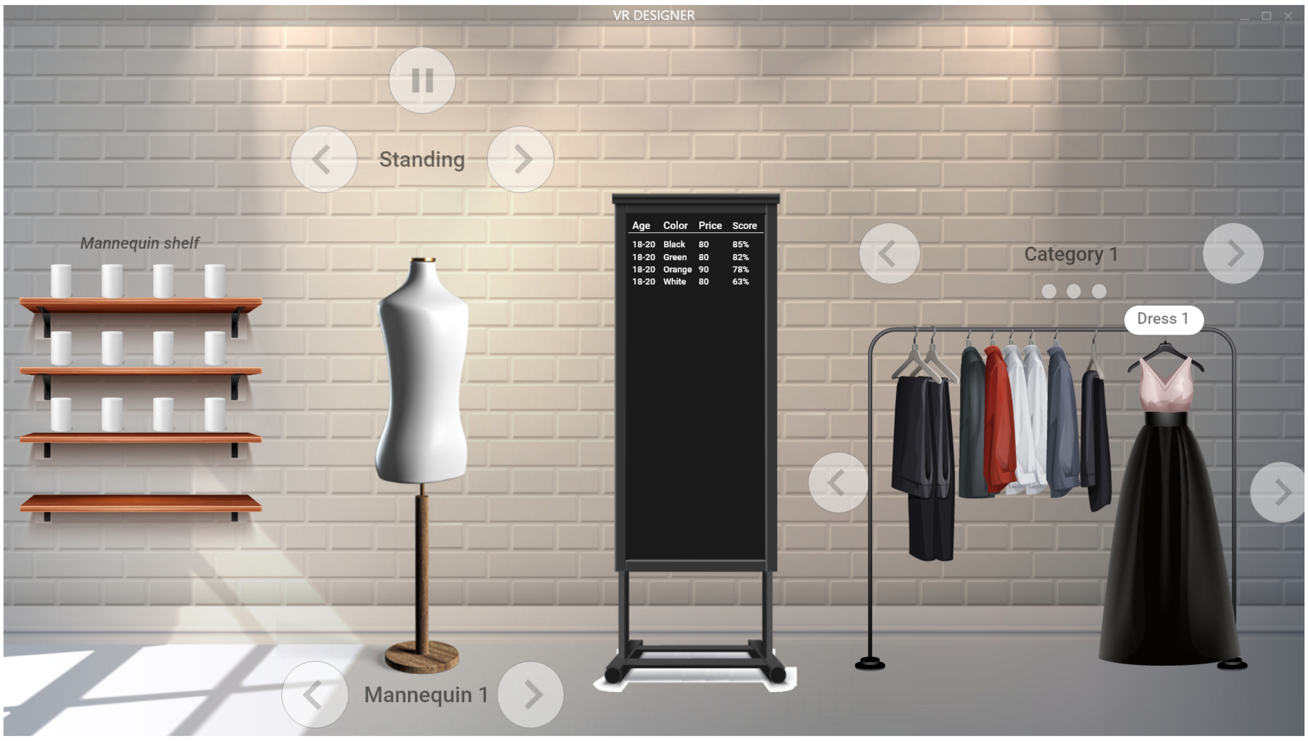
```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://etryon-h2020.eu/schema/uc3/user_info.json",
    "type": "object",
    "title": "The root schema",
    "description": "The root schema comprises the entire JSON document.",
    "default": {},
    "examples": [
        {
            "birth_date": "16/07/1985",
            "height": "183",
            "weight": "72",
            "units": "metric",
            "show_recommendations": false,
            "garment_recommendations": [
                "234435234",
                "234435234",
                "234435234"
            ],
            "gender": "male",
            "garment_size": "L",
            "saved_garments": {
                "garment_id": "234435234",
                "photos": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename1.png",

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename2.png",

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename3.png"
                ]
            },
            "avatar":
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%user%
enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.fbx"
        }
    ],
    "required": [
        "birth_date",
        "height",
        "weight",
        "units",
        "show_recommendations",
        "garment_recommendations",
        "gender",
        "garment_size",
        "saved_garments",
        "avatar"
    ],
    "properties": {
        "birth_date": {
            "$id": "#/properties/birth_date",
            "default": "",
            "description": "The user birth date in dd/mm/yyyy format",
            "examples": [
                "16/07/1985"
            ],
            "title": "The birth_date property",
            "type": "string"
        },
        "height": {
            "$id": "#/properties/height",
            "default": "",
            "description": "User height in centimeters",
            "examples": [
                "183"
            ],
            "title": "The height property",
            "type": "string"
        },
        "weight": {
            "$id": "#/properties/weight",
            "default": "",
            "description": "User weight in kilos",
            "examples": [
                "72"
            ],
            "title": "The weight property",
            "type": "string"
        },
        "units": {
            "$id": "#/properties/units",
            "default": "",
            "description": "Unit system (metric / imperial)",
            "examples": [
                "metric",
                "imperial"
            ],
            "title": "The units property",
            "type": "string"
        },
        "show_recommendations": {
            "$id": "#/properties/show_recommendations",
            "default": false,
            "description": "If true, recommended garments are shown to the user",
            "examples": [
                true,
                false
            ],
            "title": "The show_recommendations property",
            "type": "boolean"
        },
        "garment_recommendations": {
            "$id": "#/properties/garment_recommendations",
            "default": [],
            "description": "An array that contains recommended garment ids",
            "examples": [
                [
                    "234435234",
                    "234435234"
                ]
            ],
            "title": "The garment_recommendations array",
            "type": "array",
            "additionalItems": false,
            "items": {
                "$id": "#/properties/garment_recommendations/items",
                "anyOf": [
                    {
                        "$id": "#/properties/garment_recommendations/items/anyOf/0",
                        "default": "",
                        "description": "A garment id",
                        "examples": [
                            "234435234"
                        ],
                        "title": "The first anyOf property",
                        "type": "string"
                    }
                ]
            }
        },
        "gender": {
            "$id": "#/properties/gender",
            "default": "",
            "description": "User gender",
            "examples": [
                "male",
                "female"
            ],
            "title": "The gender schema",
            "type": "string"
        },
        "garment_size": {
            "$id": "#/properties/garment_size",
            "default": "",
            "description": "User preference on garment size",
            "examples": [
                "S",
                "M",
                "L",
                "XL",
                "XXL"
            ],
            "title": "The garment_size property",
            "type": "string"
        },
        "saved_garments": {
            "$id": "#/properties/saved_garments",
            "default": {},
            "description": "An array that consists of saved garment objects",
            "examples": [
                {
                    "garment_id": "234435234",
                    "photos": [
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename1.png",
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename2.png",
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename3.png"
                    ]
                }
            ],
            "required": [
                "garment_id",
                "photos"
            ],
            "title": "The saved_garments array",
            "type": "object",
            "properties": {
                "garment_id": {
                    "$id": "#/properties/saved_garments/properties/garment_id",
                    "default": "",
                    "description": "The garment id (EAN code)",
                    "examples": [
                        "234435234"
                    ],
                    "title": "The garment_id property",
                    "type": "string"
                },
                "photos": {
                    "$id": "#/properties/saved_garments/properties/photos",
                    "default": [],
                    "description": "An array that features photos taken by the user while trying on a garment",
                    "examples": [
                        [
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename1.png",
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename2.png",

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename3.png"
                                    ]
                              ],
                              "title": "The photos array",
                              "type": "array",
                              "additionalItems": false,
                              "items": {
                                    "$id":
"#/properties/saved_garments/properties/photos/items",
                                    "anyOf": [
                                          {
                                                "$id":
"#/properties/saved_garments/properties/photos/items/anyOf/0",
                                                "default": "",
                                                "description": "A URL that points to a photo in
Firebase Storage",
                                                "examples": [

"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/photo%user%e
nK0PQ8YY0hXDICJm1MKfjYTTvu1%filename1.png"
                                                ],
                                                "title": "The first anyOf property",
                                                "type": "string"
                                          }
                                    ]
                              }
                        }
                  }
            },
            "avatar": {
                  "$id": "#/properties/avatar",
                  "default": "",
                  "description": "A URL for the user avatar that resides in Firebase
Storage",
                  "examples": [
"https://firebasestorage.googleapis.com/v0/b/etryon-h2020.appspot.com/o/avatar%user%enK0PQ8YY0hXDICJm1MKfjYTTvu1%filename.fbx"
            ],
            "title": "The avatar property",
            "type": "string"
        }
    }
}
```

12 Appendix F - Usage Scenarios from https://etryon.gitlab.io/techdocs/

12.1 UC1 - VR Designer App



12.1.1 Sign up / Sign in screen



Age	Color	Price	Score
18-20	Black	80	85%
18-20	Green	80	82%
18-20	Orange	90	78%
18-20	White	80	63%

12.1.2 VR Designer app mockup

12.1.3 Usage Scenario 0.0

Use Case id: 1.0.0

Primary Actor: Designer

Description

The user must sign up or sign in to use the app.

Pre-Conditions:

None.

Task Sequence:

In order to use the app, the user must Sign Up providing a Full Name, e-mail and password.

Alternatively the user can Sign In using the credentials of an existing account.

12.1.4 Usage Scenario 1.0: Garment Interaction

Use Case id: 1.1.0

Primary Actor: Designer

Description

The user can shift through the garments offered inside the VR environment, preview them and select them in order to fit them on the mannequin that is located in the center of the environment.

Pre-Conditions:

1. The garments have been loaded successfully on the rack that features them.

Task Sequence:

While inside the VR environment, the user can browse through all the different garments/clothes available, that are presented to him on a rack.

If there are too many clothes for the rack to show, there are two buttons that the user can push, situated in the left and right of the rack, in order for the rack to shift to another batch of clothes.

The garment rack can be further filtered down by category using the two arrows on top. The category name is shown as a text label.

By having the user selector hover over a rack item, the cloth/garment is ‘expanded’ and a preview is shown in full.

By selecting this garment, it is fitted on the mannequin that is located in the center of the room.

12.1.5 Usage Scenario 2.0: Mannequin Selection

Use Case id: 1.2.0

Primary Actor: Designer

Description

The user can select a mannequin available inside the VR environment and replace the active current one.

Pre-Conditions:

1. The mannequin shelf has successfully loaded all available mannequins.

Task Sequence:

In the center of the room, resides the active mannequin. In order to change it the user can:

- Tap on the arrows in the bottom of the model to cycle through them, or
- In the left of the model is a shelf, featuring all mannequins available for overview. By selecting one, the model in the center of the room is updated.

12.1.6 Usage Scenario 3.0: Animation selection - playback

Use Case id: 1.3.0

Primary Actor: Designer

Description

The user can shift through default character animations and play or pause them.

Pre-Conditions:

1. None.

Task Sequence:

While inside the VR environment, the user can shift through available animations by clicking on the arrows that are located on top of the active mannequin, and watch the animations with the selected model and fitted garment.

By selecting the play/pause icon, the user can play or pause the selected animation sequence.

12.1.7 Usage Scenario 4.0: Export a creation

Use Case id: 1.4.0

Primary Actor: Designer

Description

The user can export a creation.

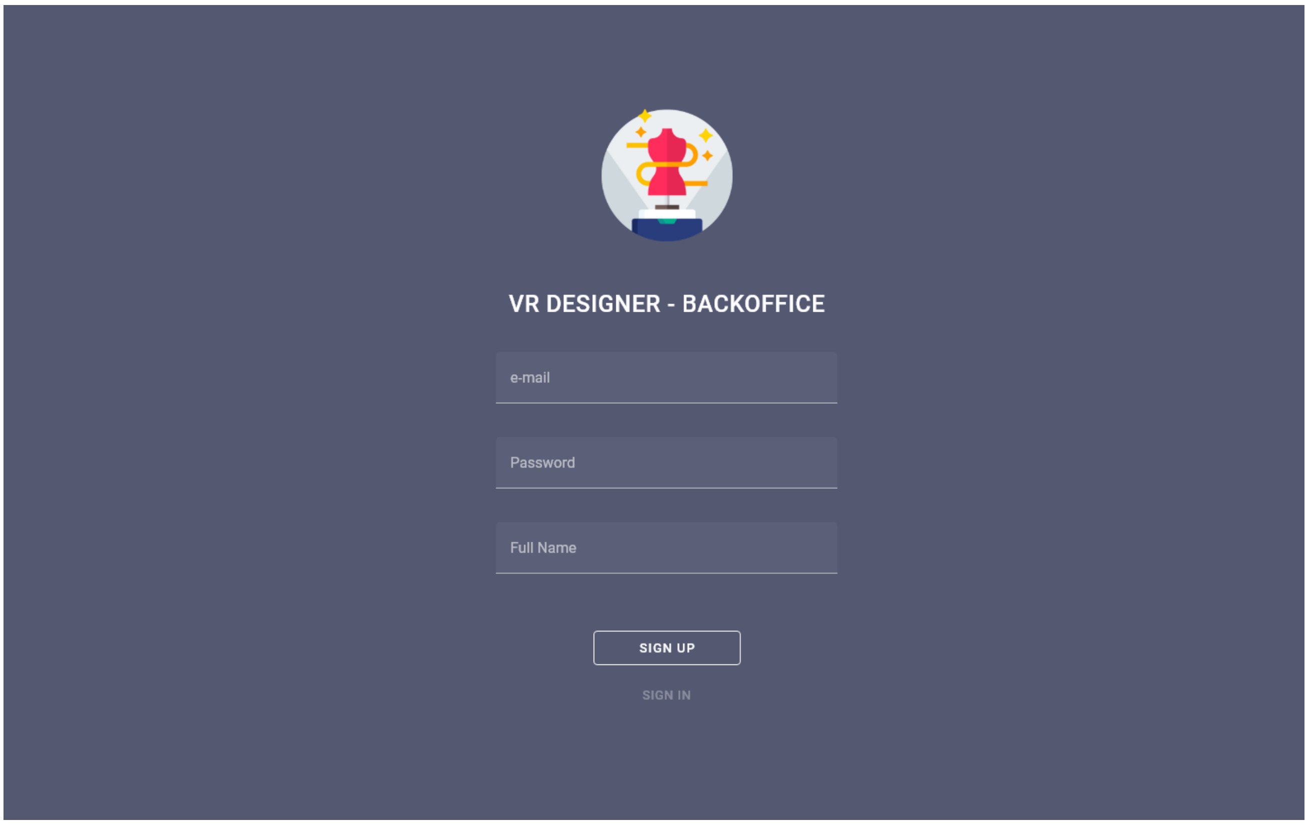
Pre-Conditions:

None.

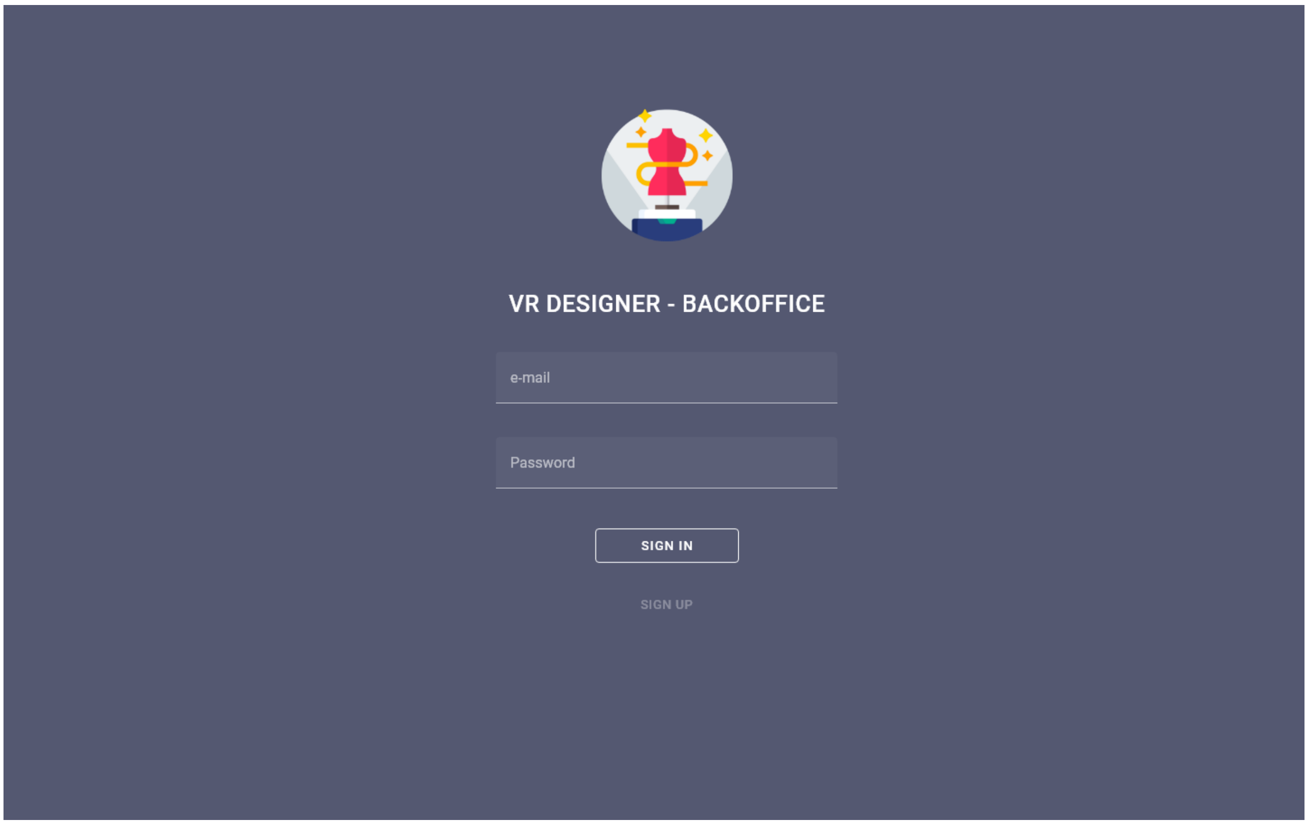
Task Sequence:

When the user is done inspecting the active mannequin and is finished with setting up garment(s), animation and pose (mannequin selection), the resulting creation can be exported into a serialized binary and subsequently shared with a product manager.

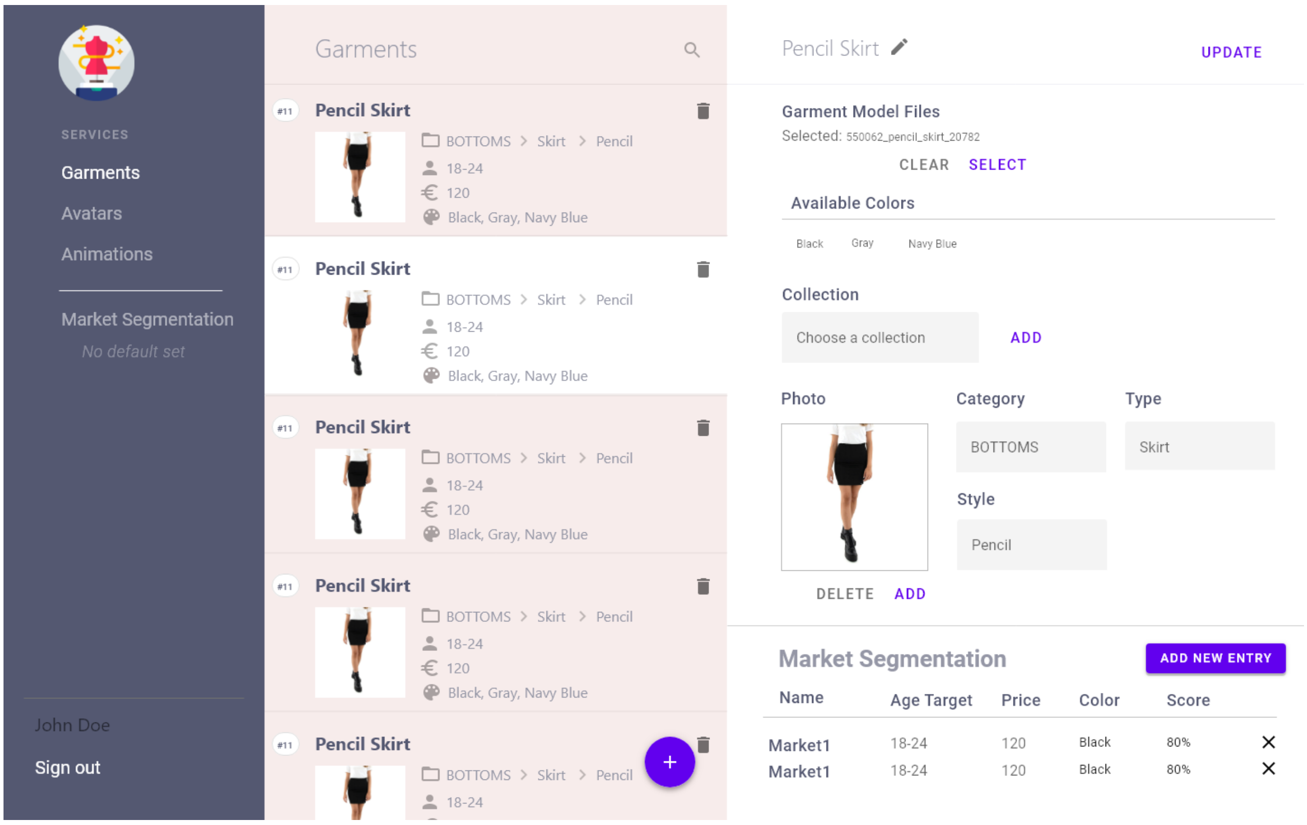
12.2 UC1 - VR Designer Backoffice App



Sign Up View



Sign In View



Garments

#11 Pencil Skirt	BOTTOMS > Skirt > Pencil; 18-24; 120; Black, Gray, Navy Blue
#11 Pencil Skirt	BOTTOMS > Skirt > Pencil; 18-24; 120; Black, Gray, Navy Blue
#11 Pencil Skirt	BOTTOMS > Skirt > Pencil; 18-24; 120; Black, Gray, Navy Blue
#11 Pencil Skirt	BOTTOMS > Skirt > Pencil; 18-24; 120; Black, Gray, Navy Blue
#11 Pencil Skirt	BOTTOMS > Skirt > Pencil; 18-24

Pencil Skirt

Garment Model Files
Selected: 550062_pencil_skirt_20782

Available Colors
Black Gray Navy Blue

Collection: Choose a collection

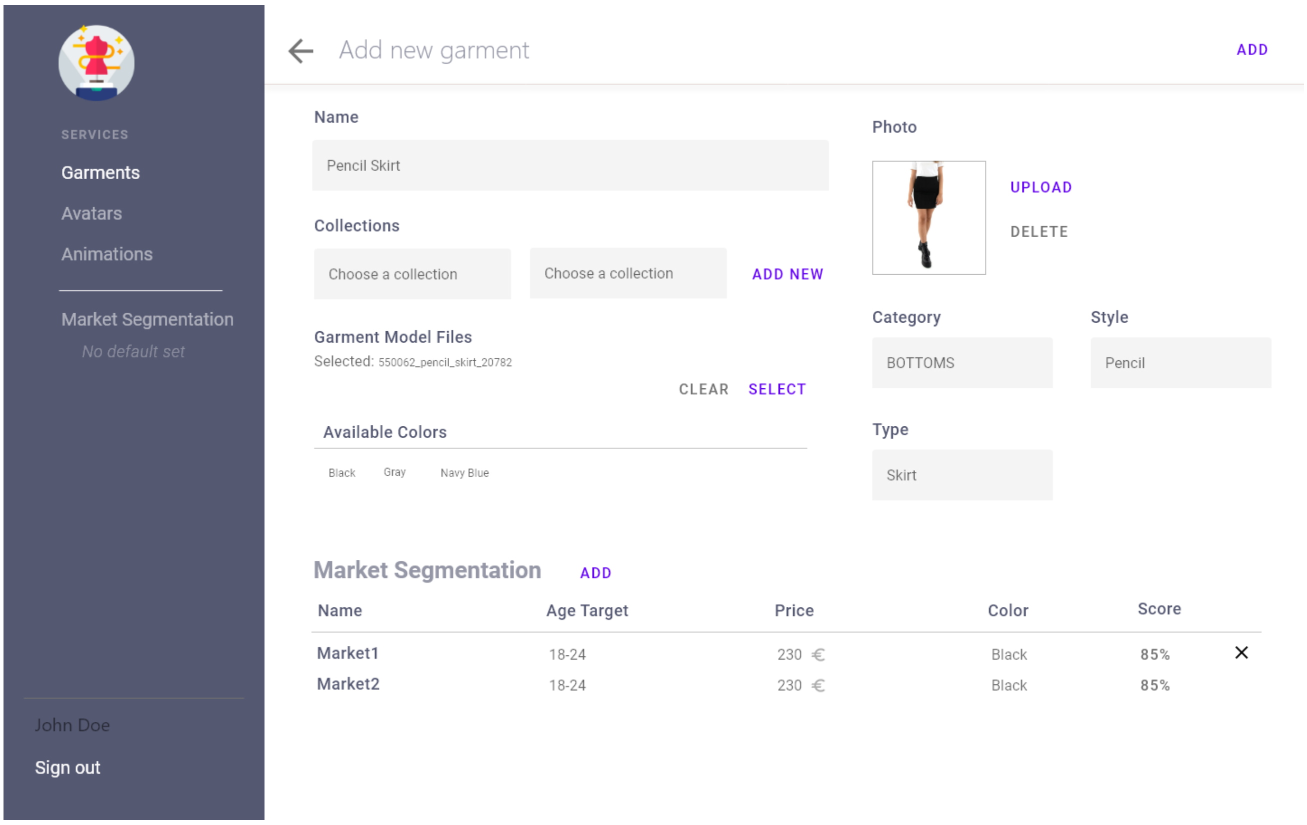
Photo

Category: BOTTOMS
Type: Skirt
Style: Pencil

Market Segmentation

Name	Age Target	Price	Color	Score
Market1	18-24	120	Black	80%
Market1	18-24	120	Black	80%

Garments View



Add new garment

Name: Pencil Skirt

Collections: Choose a collection | Choose a collection

Garment Model Files
Selected: 550062_pencil_skirt_20782

Available Colors
Black Gray Navy Blue

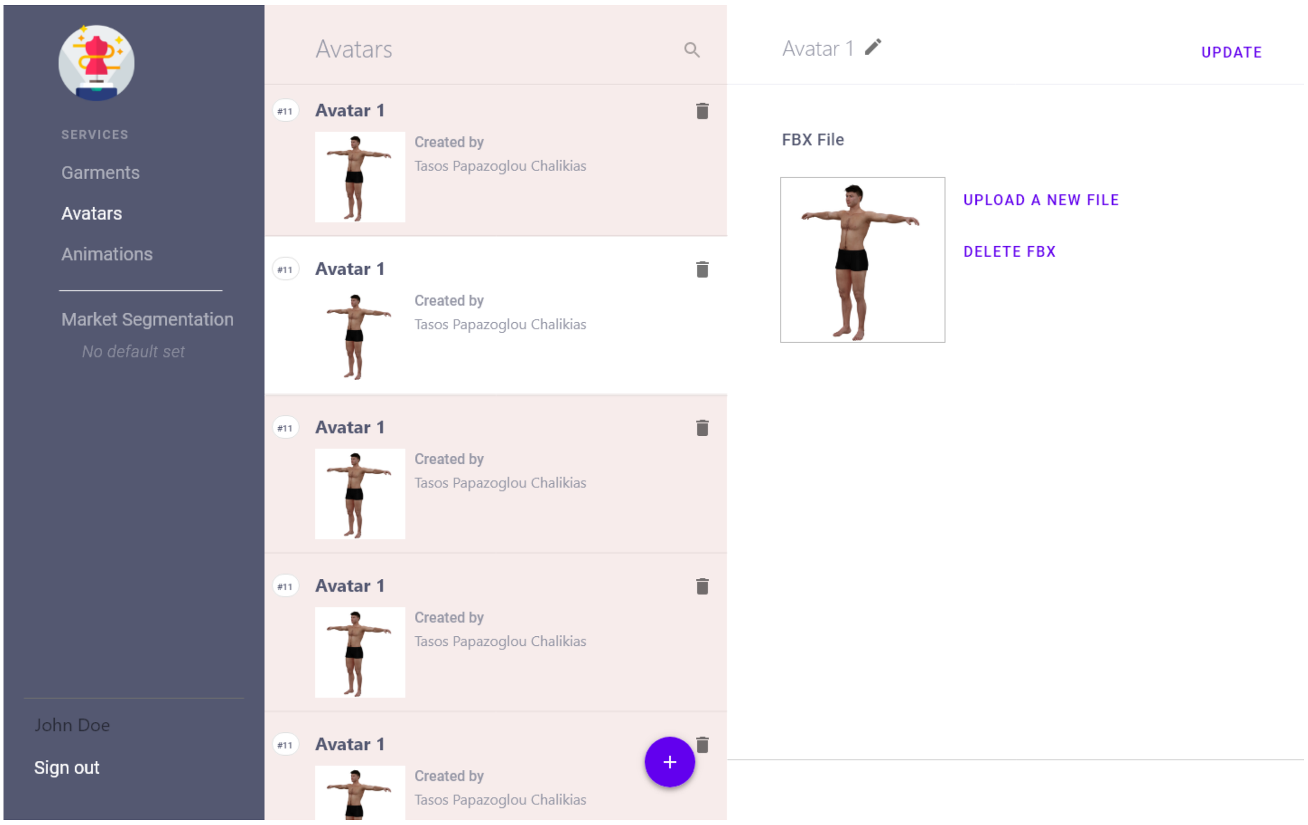
Photo

Category: BOTTOMS
Style: Pencil
Type: Skirt

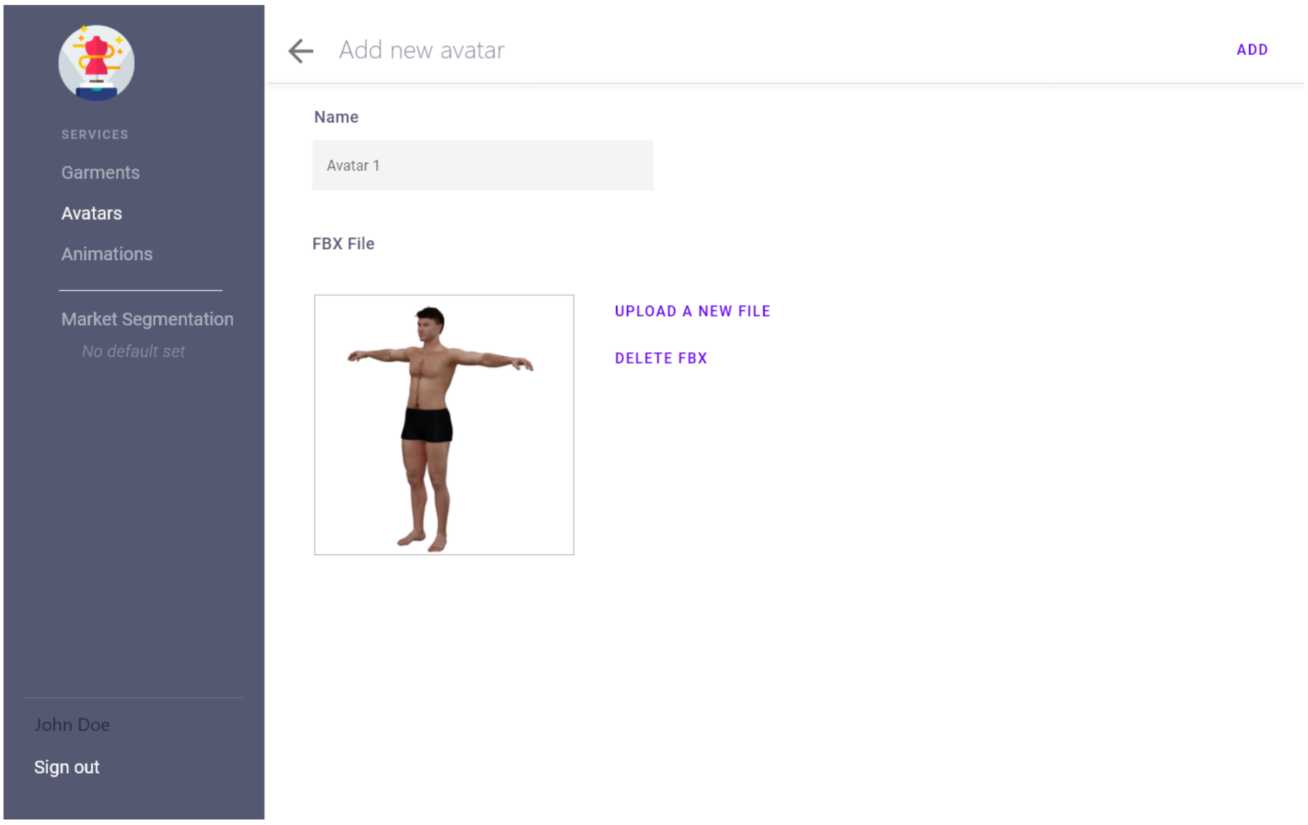
Market Segmentation

Name	Age Target	Price	Color	Score
Market1	18-24	230 €	Black	85%
Market2	18-24	230 €	Black	85%

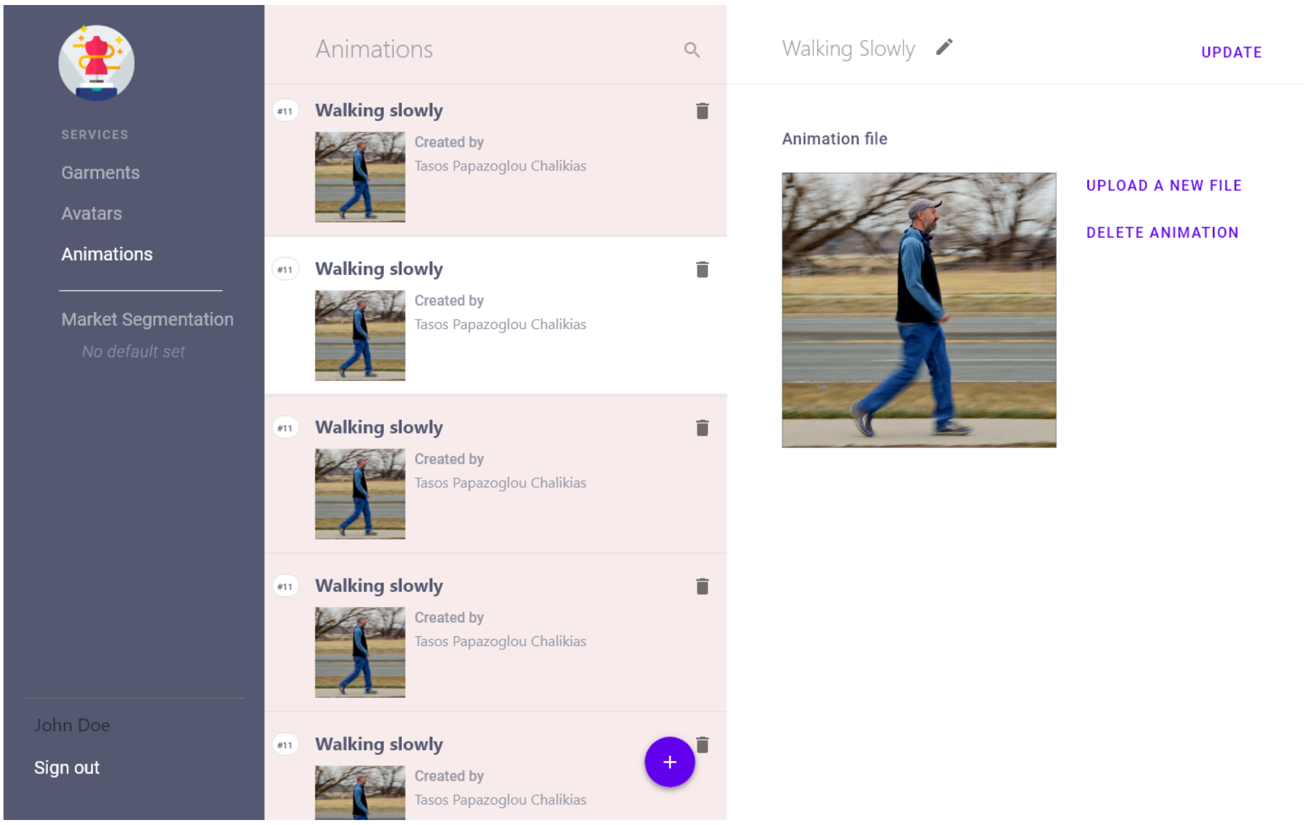
Add New Garment View



Avatars View



Add New Avatar View

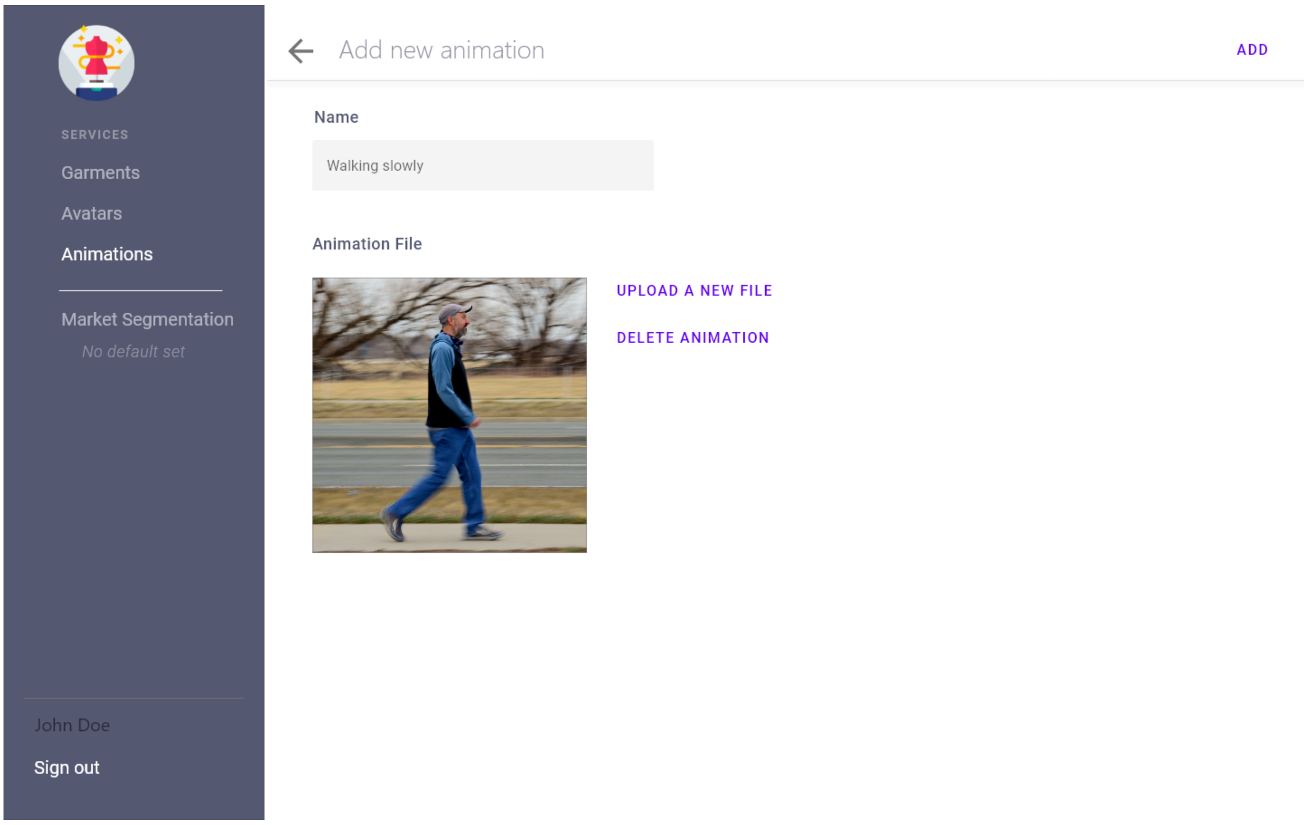

SERVICES
Garments
Avatars
Animations
Market Segmentation
No default set
John Doe
Sign out

Animations

#11	Walking slowly Created by Tasos Papazoglou Chalikias
#11	Walking slowly Created by Tasos Papazoglou Chalikias
#11	Walking slowly Created by Tasos Papazoglou Chalikias
#11	Walking slowly Created by Tasos Papazoglou Chalikias
#11	Walking slowly Created by Tasos Papazoglou Chalikias

Walking Slowly
UPDATE
Animation file
UPLOAD A NEW FILE
DELETE ANIMATION

Animations View


SERVICES
Garments
Avatars
Animations
Market Segmentation
No default set
John Doe
Sign out

Add new animation
ADD
Name
Walking slowly
Animation File
UPLOAD A NEW FILE
DELETE ANIMATION

Add New Animation View

12.2.1 Usage Scenario 1.0

Use Case id: 4.1.0

Primary Actor: Designer

Description

The user must sign up or sign in to use the app.

Pre-Conditions:

None.

Task Sequence:

In order to upload Garments, Animations or Avatars, the user must first login so that the files uploaded have a creator.

The user can Sign Up providing a Full Name, e-mail and password.

Alternatively the user can Sign In using the credentials of an existing account.

12.2.2 Usage Scenario 2.0: Garments View Interactions

Use Case id: 4.2.0

Primary Actor: Designer

Description

The user can see all the uploaded garments and perform various actions in the Garments View.

Pre-Conditions:

None.

Task Sequence:

After successfully logging in, the user is shown a Dashboard featuring three columns. The first column features the navigation of the Dashboard, so the user can easily switch between the different types of items. There is also a Sign Out button. The second column features a list of the uploaded garment items. The user can perform the following actions:

- Search for a garment item by name, using the search icon in the top right of the column.
- View all available garments in the form of a list.
- Delete a garment item.
- Add a new garment item by clicking on the FAB button floating in the bottom right.

- Select a garment item by clicking on it, where the third column is populated with editable garment item details.

By clicking on a garment item in the list, the third column of the Dashboard is populated with the editable details of the garment item. More specifically the user can:

- Change the Title of the garment item.
- Add or remove a Garment. Three things must be selected, an FBX file a zip containing the textures and a json file with the texture metadata.
- Add or remove a garment photo.
- Specify Categories.
- Specify the Collection(s) a Garment is part of.
- Add / Remove Market Segmentation data.

After all edits have been made, the user can click on the UPDATE button to update the garment item with the new information.

12.2.3 Usage Scenario 2.1: Add New Garment Item

Use Case id: 4.2.1

Primary Actor: Designer

Description

The user can upload a new garment item.

Pre-Conditions:

1. 4.2.0

Task Sequence:

While being on the Garments View the user can click on the FAB in the bottom right of the Garment List in order to launch the Add New Garment view.

In this view the user has to input the following information in order to add a new garment item to the system:

- Garment Name (Title).
- Select a Collection(s).
- Add a Garment. Three things must be selected, an FBX file a zip containing the textures and a json file with the texture metadata.
- Add a Garment photo.
- Select Categories.
- Specify Market Segmentation entries.

12.2.4 Usage Scenario 3.0: Avatars View Interactions

Use Case id: 4.3.0

Primary Actor: Designer

Description

The user can see all the uploaded avatars and perform various actions in the Avatars View.

Pre-Conditions:

None.

Task Sequence:

This view features a list of the uploaded avatars. The user can perform the following actions:

- Search for an avatar by name, using the search icon in the top right of the column.
- View all available avatars in the form of a list.
- Delete an avatar.
- Add a new avatar by clicking on the FAB button floating in the bottom right.
- Select an avatar by clicking on it, where the third column is populated with editable avatar details.

By clicking on an avatar item in the list, the third column of the Dashboard is populated with its editable details. More specifically the user can:

- Change the avatar title.
- Add or remove the FBX file of the avatar.

The user can click on the SAVE button to update the avatar.

12.2.5 Usage Scenario 3.1: Add New Avatar

Use Case id: 4.3.1

Primary Actor: Designer

Description

The user can upload a new avatar.

Pre-Conditions:

1. 4.3.0

Task Sequence:

While being on the Avatars View the user can click on the FAB in the bottom right of the Avatars List in order to launch the Add New Avatar view.

In this view the user has to input the following information in order to add a new avatar to the system:

- Avatar Name (Title).
- FBX File

12.2.6 Usage Scenario 4.0: Animations View Interactions

Use Case id: 4.4.0

Primary Actor: Designer

Description

The user can see all the uploaded animations and perform various actions in the Animations View.

Pre-Conditions:

None.

Task Sequence:

This view features a list of the uploaded animations. The user can perform the following actions:

- Search for an animation by name, using the search icon in the top right of the column.
- View all available animations in the form of a list.
- Delete an animation.
- Add a new animaton by clicking on the FAB button floating in the bottom right.
- Select an animation by clicking on it, where the third column is populated with editable animation details.

By clicking on an animation item in the list, the third column of the Dashboard is populated with its editable details. More specifically the user can:

- Change the animation title.
- Add or remove the FBX file of the animation.

The user can click on the SAVE button to update the animation.

12.2.7 Usage Scenario 4.1: Add New Animation

Use Case id: 4.4.1

Primary Actor: Designer

Description

The user can upload a new animation.

Pre-Conditions:

1. 4.4.0

Task Sequence:

While being on the Animations View the user can click on the FAB in the bottom right of the Animations List in order to launch the Add New Animation view.

In this view the user has to input the following information in order to add a new animation to the system:

- Animation Name (Title).
- FBX File

12.2.8 Usage Scenario 5.0: Manage Market Segmentation

Use Case id: 4.5.0

Primary Actor: Designer

Description

The user can create, delete and set a default Market Segmentation.

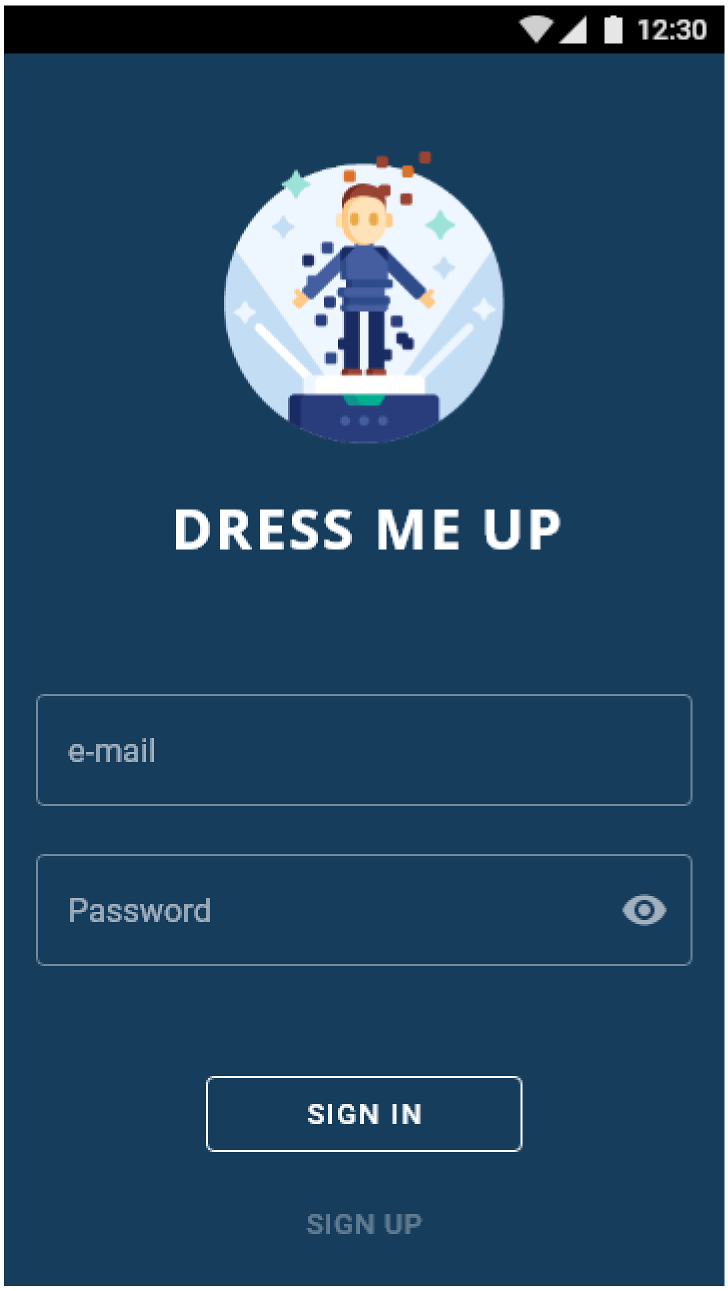
Pre-Conditions:

None.

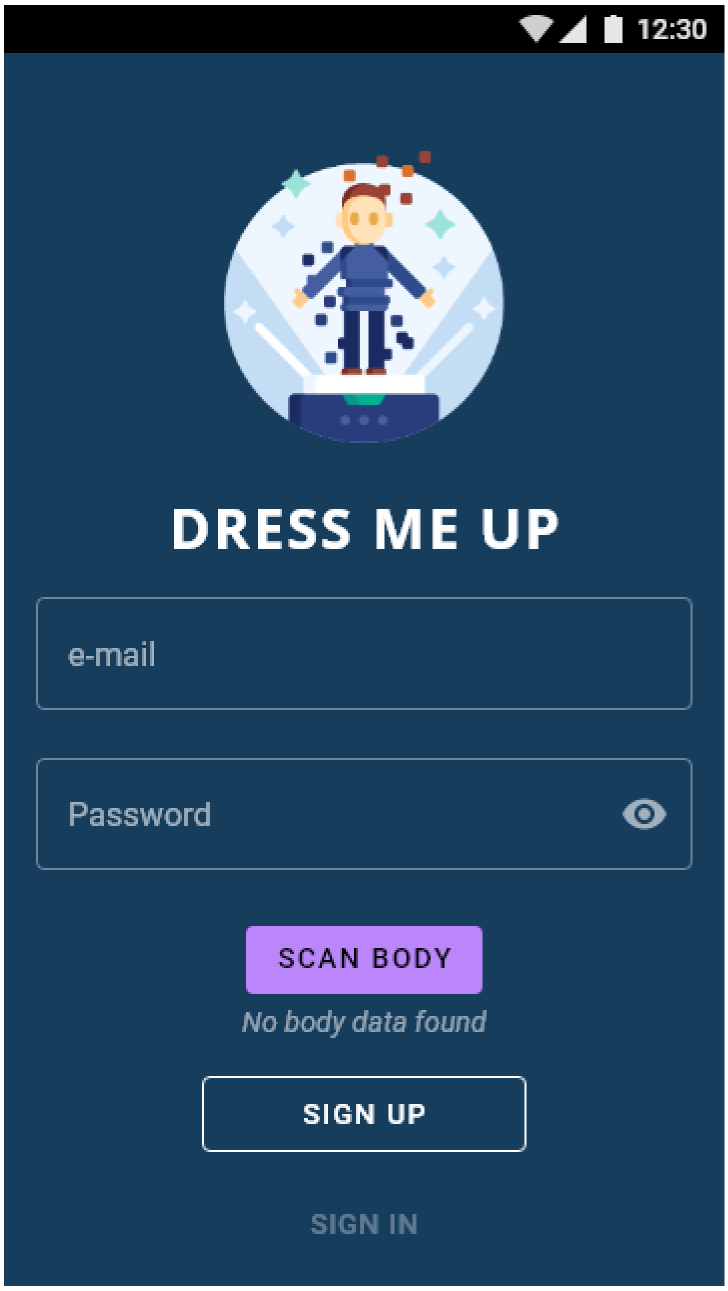
Task Sequence:

- While being on the Market Segmentation View the user can see a list of the available Market Segmentation entries. By clicking on the star icon, a Market Segment can be set as default. By clicking on the trash bin icon the entry can be deleted.
- To create a new entry, the user can click on the FAB in the bottom right, and a new entry will appear in the list.
- When a Market Segment is selected, its details are populated in the third column. There the user can change the Market Segment properties and save the entry.

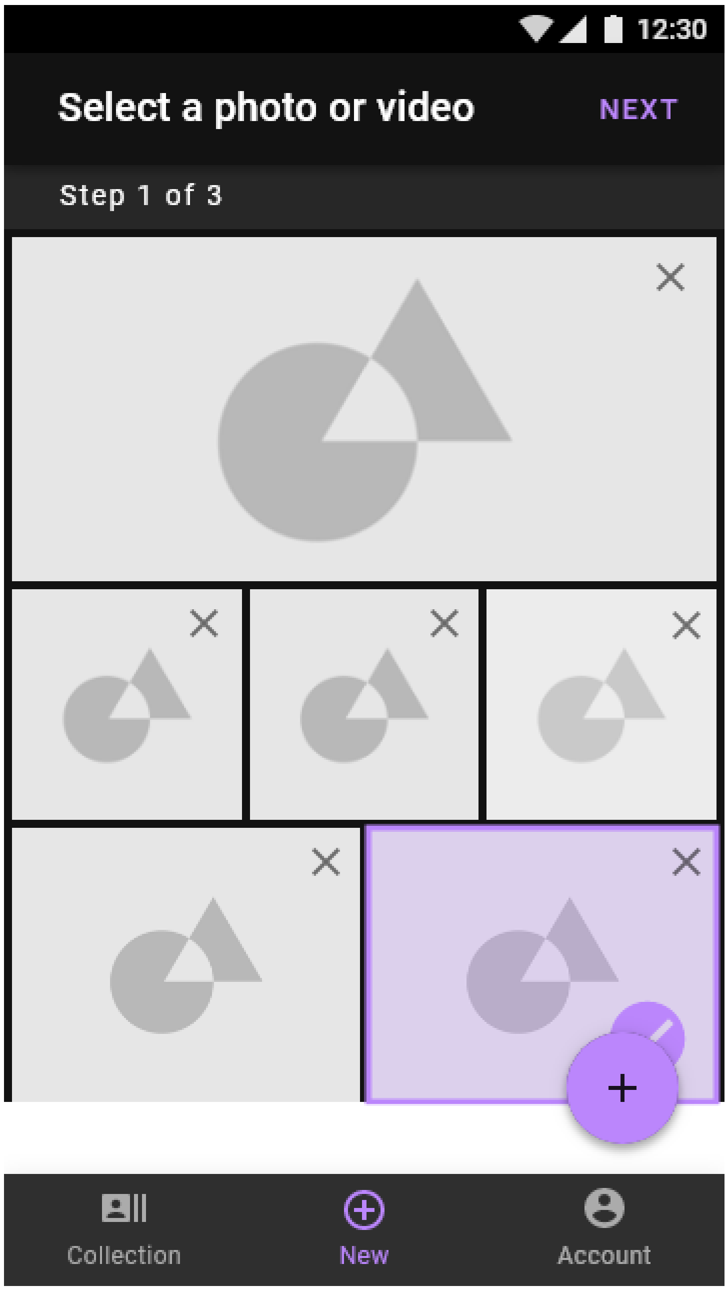
12.3 UC2 - Dress Me Up App



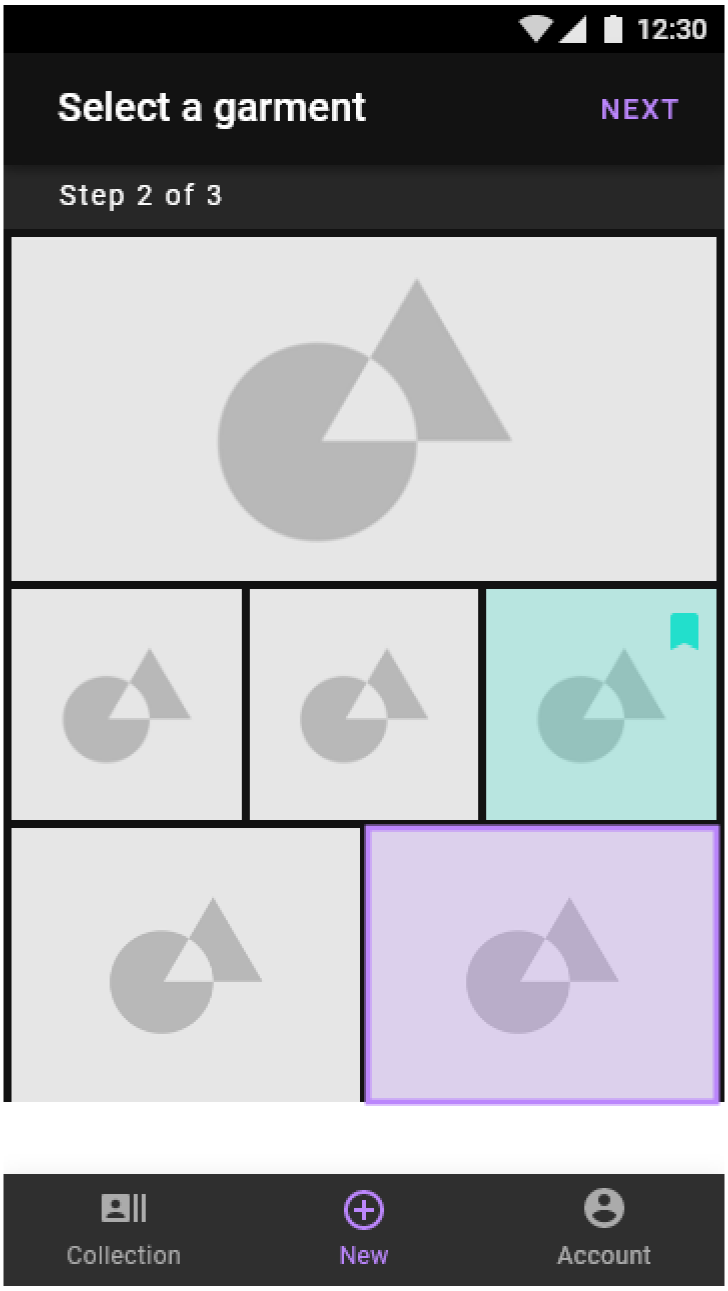
Sign In Screen



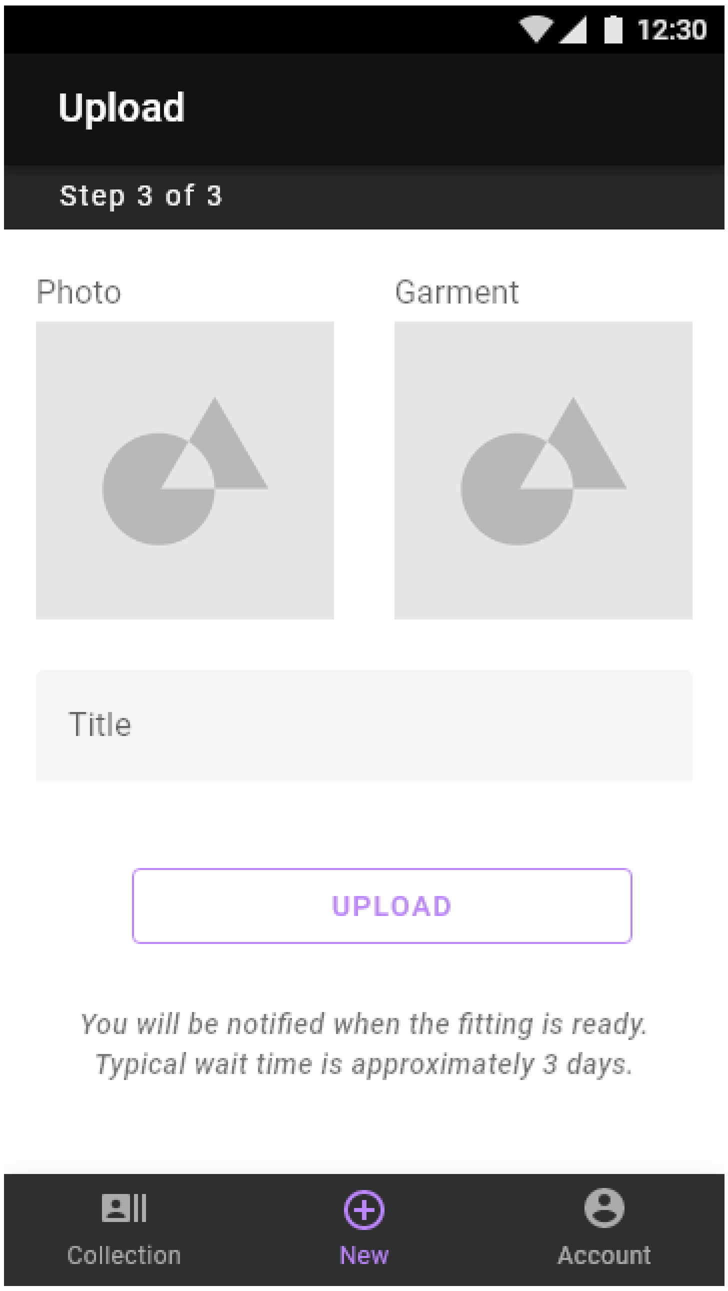
Sign Up Screen



Add New Collection Item 1

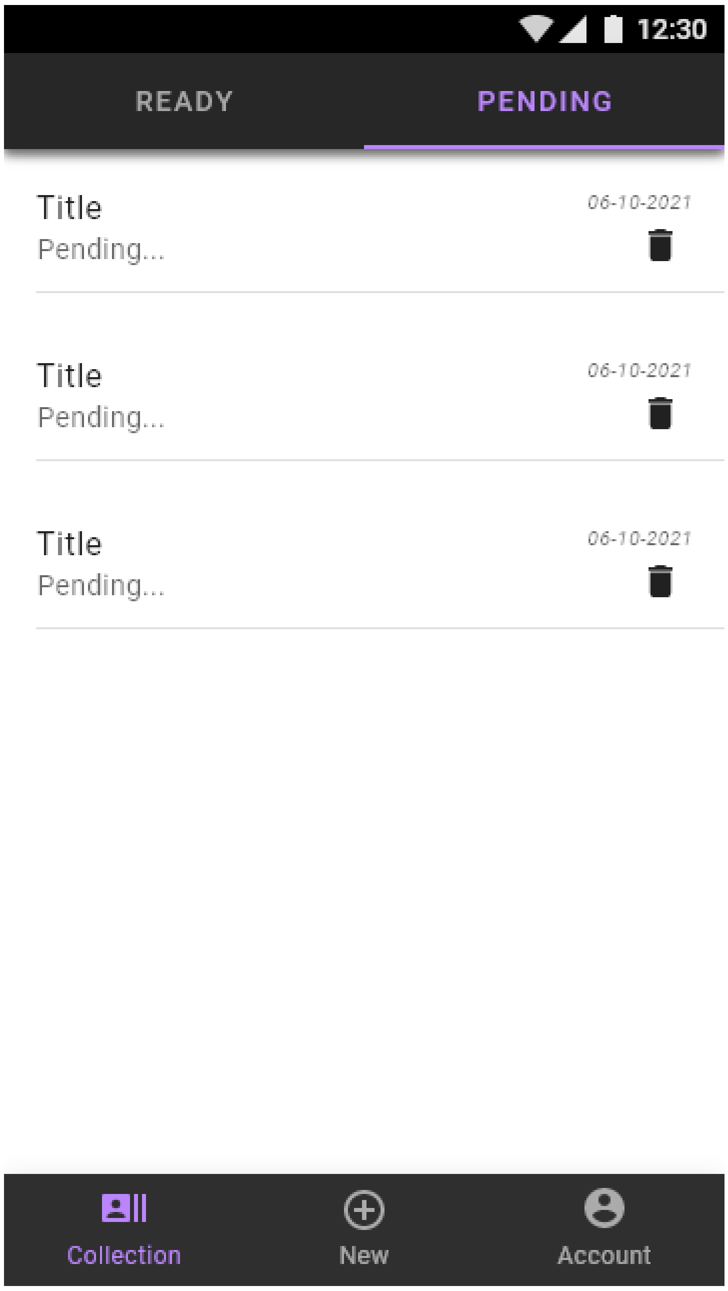


Add New Collection Item 2

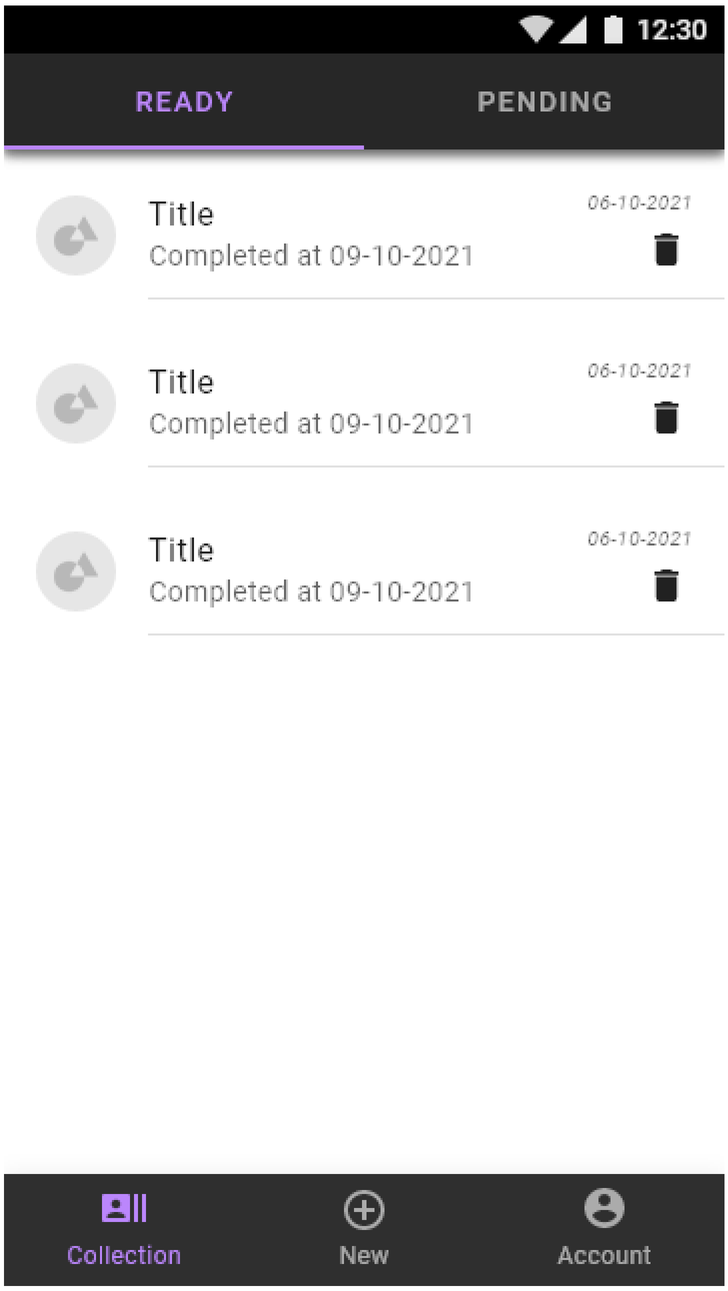


You will be notified when the fitting is ready.
Typical wait time is approximately 3 days.

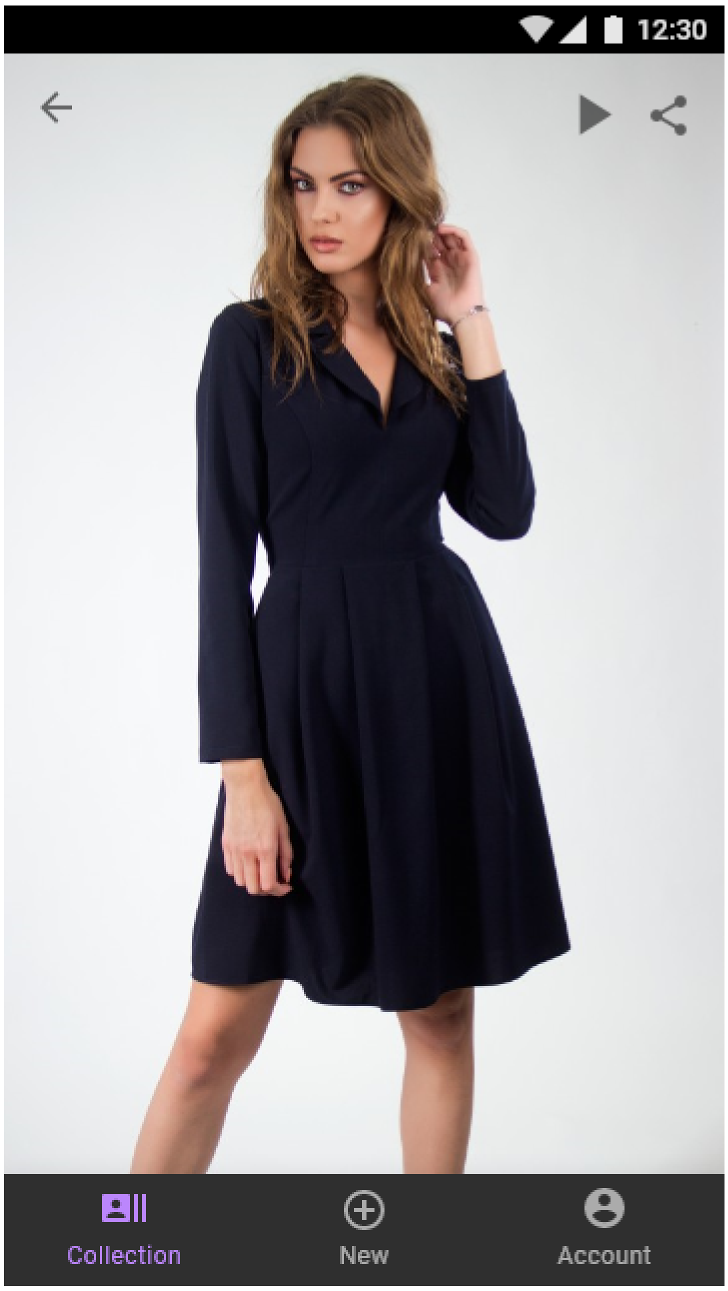
Add New Collection Item 3



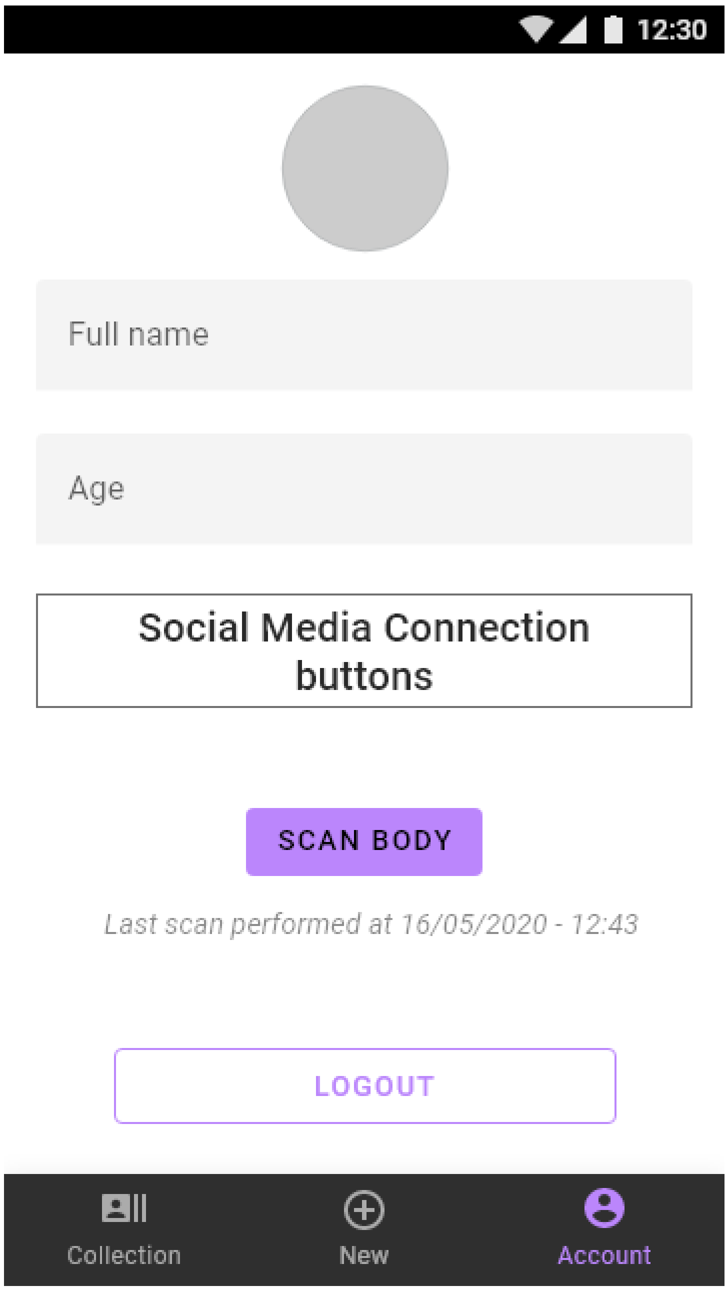
Collection - Pending View



Collection - Ready View



Collection Item Viewer


12:30
Full name
Age
Social Media Connection buttons
SCAN BODY
Last scan performed at 16/05/2020 - 12:43
LOGOUT
Collection
New
Account

Account View

12.3.1 Usage Scenario 1.0: Sign up - Sign in

Use Case id: 2.1.0

Primary Actor: End user (Consumer)

Description

The user must sign up or sign in to use the app.

Pre-Conditions:

None.

Task Sequence:

The user opens the app and its greeted with a sign up / sign in screen. Providing a combination of an e-mail and password, the user can sign up for an account, or sign in using an existing account. When signing up, the user must also scan their body.

12.3.2 Usage Scenario 2.0: Create a new collection item

Use Case id: 2.2.0

Primary Actor: Marketing exec (Brand)

Description

The user can create a new collection item to upload to the platform.

Pre-Conditions:

The product being added to the app is already available for sale, and has been imported into the system by the catalogue updater.

Task Sequence:

After logging in the app, the user must select the “New” navigation item from the menu bar in the bottom. To create a new collection item, 3 steps must be completed:

1. A grid of images or videos appear. The user can upload an image or a video by taping on the FAB in the bottom right of the screen, and delete already uploaded media. The user selects an uploaded media item and then taps on the ‘NEXT’ button at the top right to proceed.
2. In this view a garment item grid is populated. By tapping on one item it can be selected, then tapping on the ‘NEXT’ button at the top right proceeds to the next screen.
3. The selected media and garment item are shown. A title must be provided for the collection item, then by tapping on the ‘UPLOAD’ button, the files are uploaded to the platform.

The created collection item is now placed in a queue until it is processed and ready.

Use Case id: 2.2.1

Primary Actor: Marketing exec (Brand)

Description

The user can create a new collection item to upload to the platform.

Pre-Conditions:

The product being added to the app is not available for sale

Task Sequence:

12.3.3 Usage Scenario 3.0: Collection interactions

Use Case id: 2.3.0

Primary Actor: End user (Consumer)

Description

The user can view and interact with the pending or completed collection items in the Collection view.

Pre-Conditions:

None.

Task Sequence:

After logging in the app, the user must select the “Collection” navigation item from the menu bar at the bottom. The ‘Ready’ sub view is shown, which features all the computed collection items ready for viewing. The user can tap on a collection item to see it in a 3D Viewer. The user can also delete a collection item by tapping on the delete button that is situated in the right part of each collection list item. By tapping on the ‘Pending’ tab, the user can see which collection items are being processed and their creation date. The user can also delete a pending item to stop the process, by tapping on the delete icon button.

12.3.4 Usage Scenario 4.0: Collection Item interactions

Use Case id: 2.4.0

Primary Actor: End user (Consumer)

Description

The user can view and interact with the collection item once it is done processing.

Pre-Conditions:

1. A collection item must be in the “Ready” state.

Task Sequence:

By tapping on a collection item, the collection item view is loaded where the user can interact with the item in 3D. Actions that can be performed are the following:

- Using finger gestures the user can rotate the model or zoom in/out.
- If a video was selected when creating the collection item, the user can Play/Pause the item.
- The user can share the image or video to social media by tapping on the share button in the top right.

12.3.5 Usage Scenario 5.0: User Account actions

Use Case id: 2.5.0

Primary Actor: End user (Consumer)

Description

The user can input personal information in the Account view and perform various actions.

Pre-Conditions:

None.

Task Sequence:

By tapping on the ‘Account’ navigation item in the bottom navigation bar, the user can access the Account view. There the user can perform the following actions:

- Add user information,
- Connect social media accounts,
- Scan body
- Logout.

12.4 UC2 - DressMeUp Admin CLI

12.4.1 Scenario 1: Add Garment Browzwear file for a product in the catalogue

Actor

3D Designer

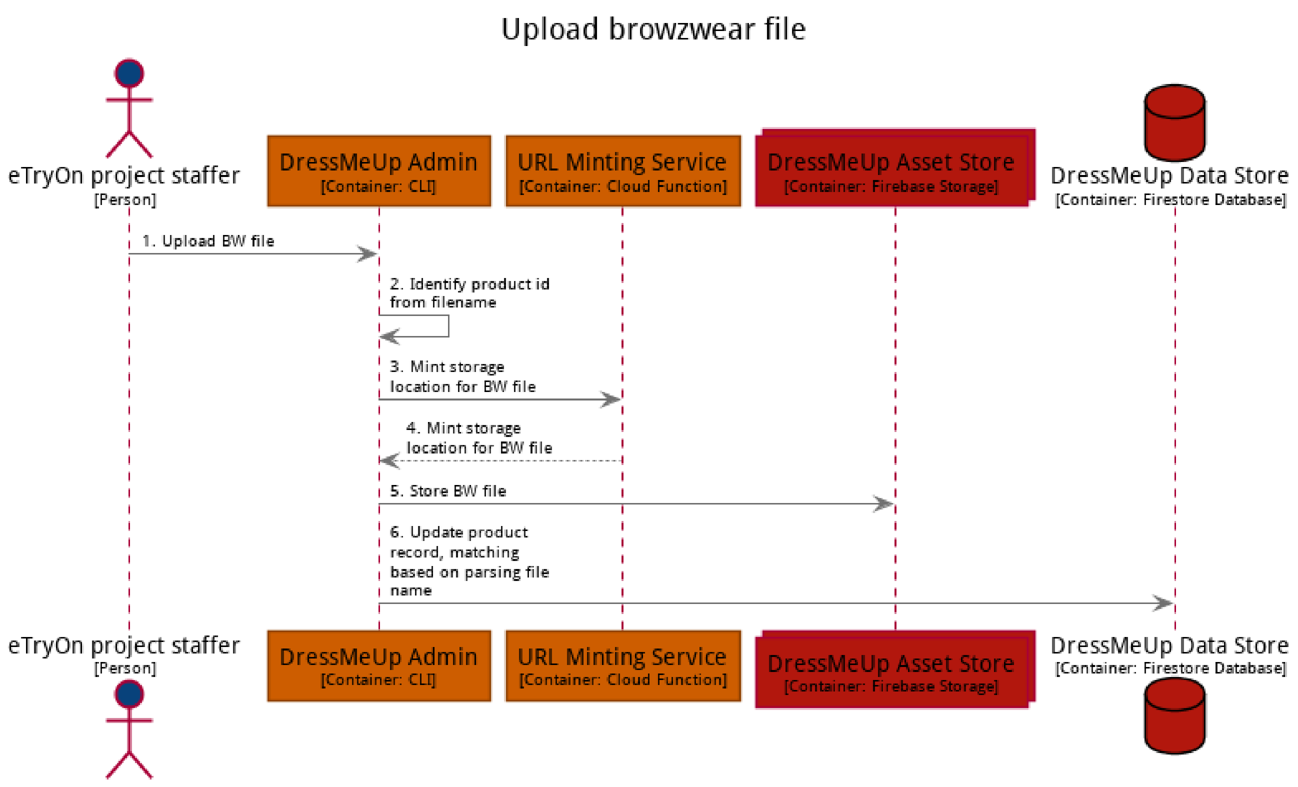
Overview

The designer uploads a browzwear file to add a garment so that end users can dress up in it. The metadata is looked up from the product feed data by product id.

Preconditions

- The designer must have arranged the panel pieces for the garment on the standard avatar
- The BW file must be named according to the Odlo file naming convention.

Operation


Upload browzwear file
eTryOn project staffer
[Person]
DressMeUp Admin
[Container: CLI]
URL Minting Service
[Container: Cloud Function]
DressMeUp Asset Store
[Container: Firebase Storage]
DressMeUp Data Store
[Container: Firestore Database]
1. Upload BW file
2. Identify product id from filename
3. Mint storage location for BW file
4. Mint storage location for BW file
5. Store BW file
6. Update product record, matching based on parsing file name

12.4.2 Scenario 2: Add Garment Browzwear file for a product not in the catalogue

Actor

3D Designer

Overview

The designer uploads a browzwear file to add a garment so that end users can dress up in it. They must provide the metadata.

Preconditions

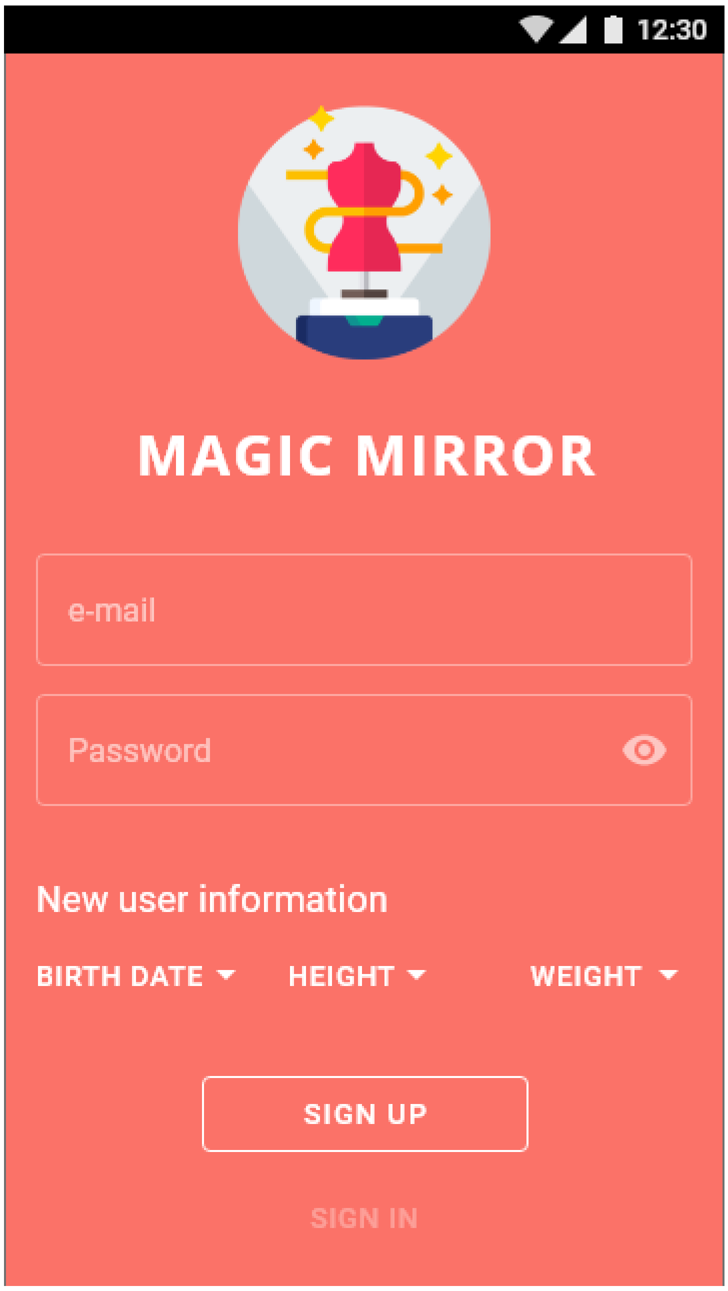
- The designer must have arranged the panel pieces for the garment on the standard avatar

Metadata

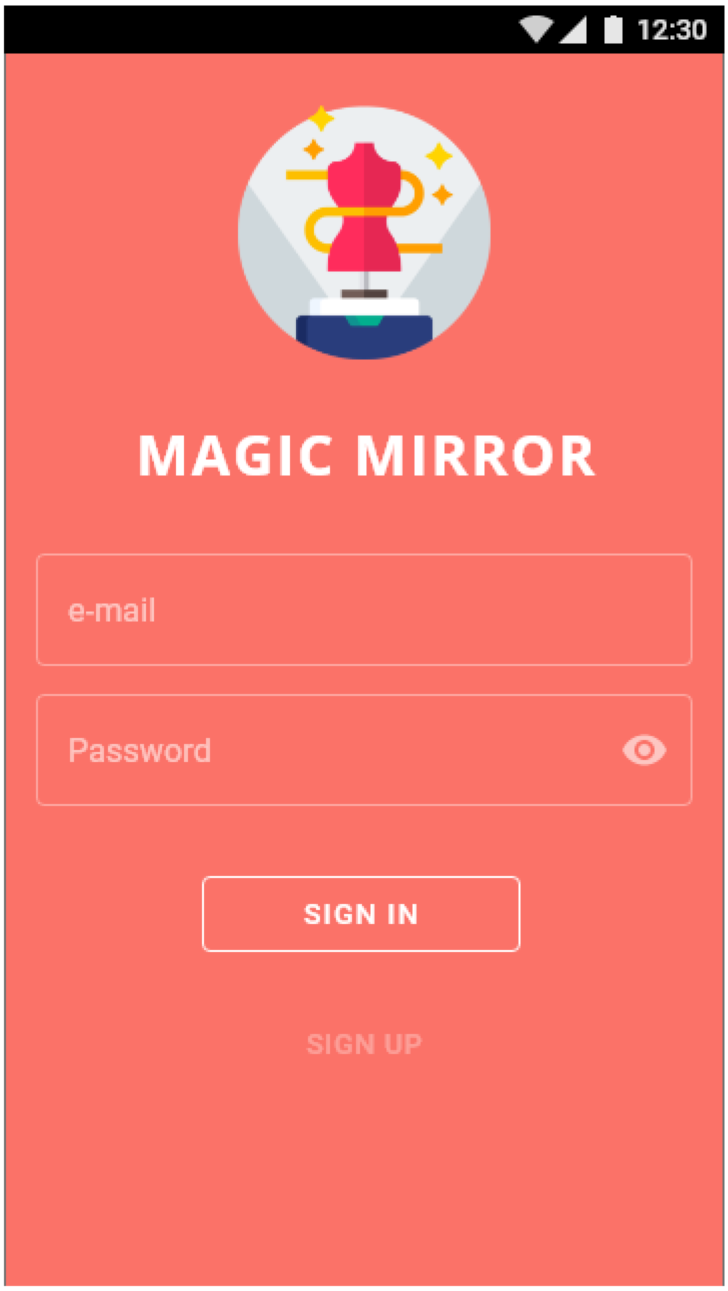
Field	Description
title	The garment title

uid	A garment unique id (EAN code)
group_id	This is the id that corresponds to the specific garment, regardless of color or size.
gender	`male` or `female`
size	`s`, `m`, `l`, `xl`, `xxl`
color	A string that denotes the color of the garment
photo	A url for a photo depicting the garment
src.file	Firebase Storage url for the 3D file
material	The garment material
product_type.full_type	A string that has the full type (path) of a garment (e.g.: `"Men>Sporting Activity>Running & Trail"`)
product_type.extracted_type_0	A string that features the first segment of a full_type string (if applicable)
product_type.extracted_type_1	A string that features the second segment of a full_type string (if applicable)
product_type.extracted_type_2	A string that features the third segment of a full_type string (if applicable)

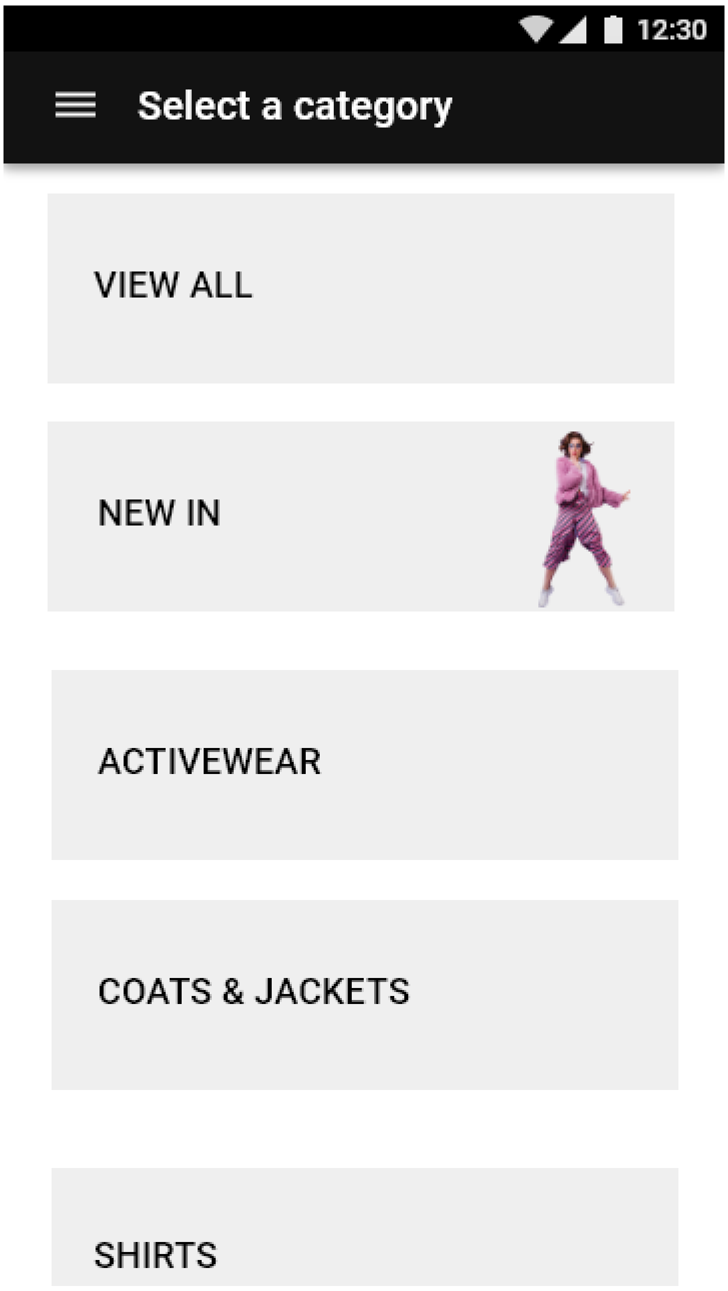
12.5 UC3 - Magic Mirror App



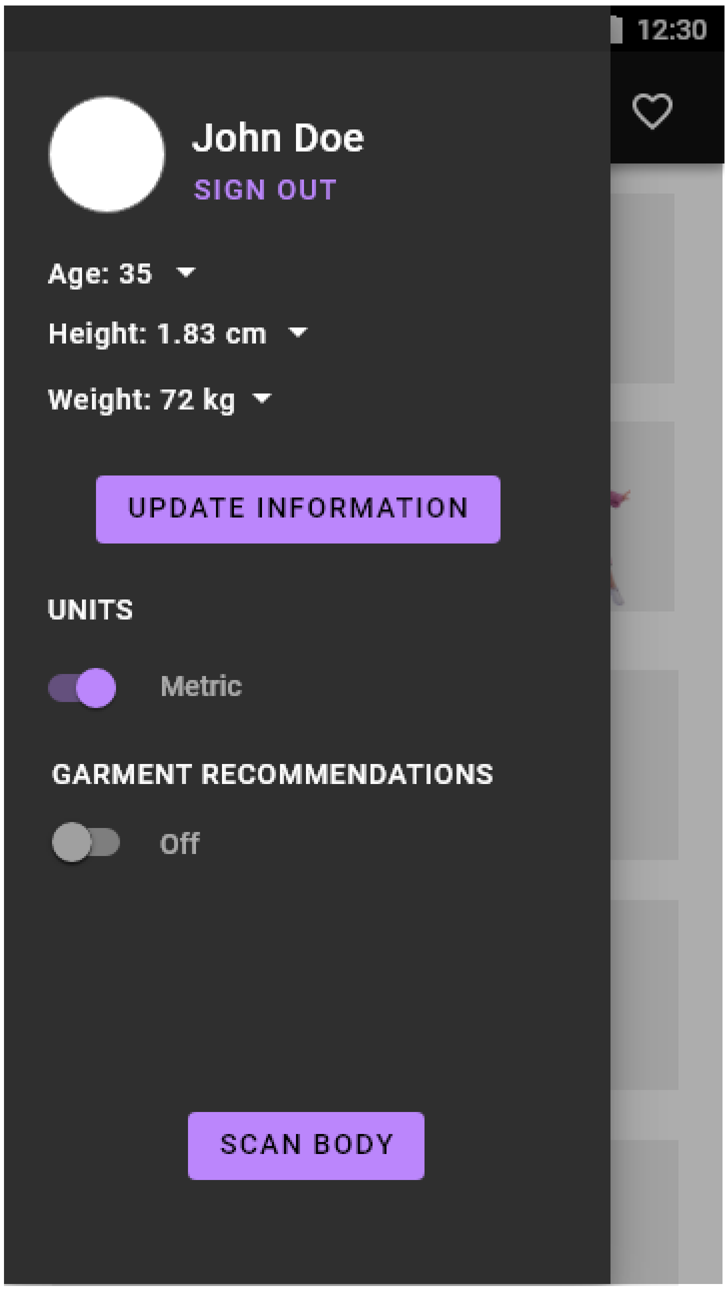
Sign Up View



Sign In View



Select a Category View


12:30

John Doe

Age: 35

Height: 1.83 cm

Weight: 72 kg

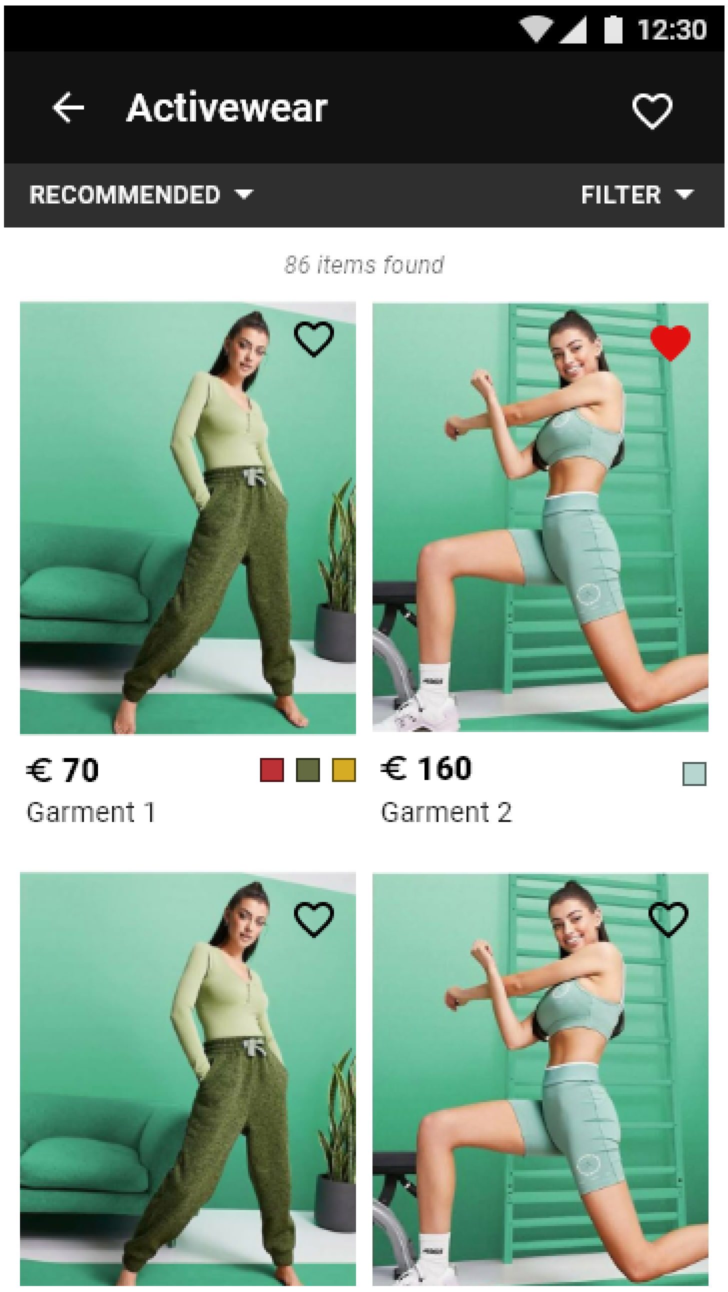
UNITS

Metric

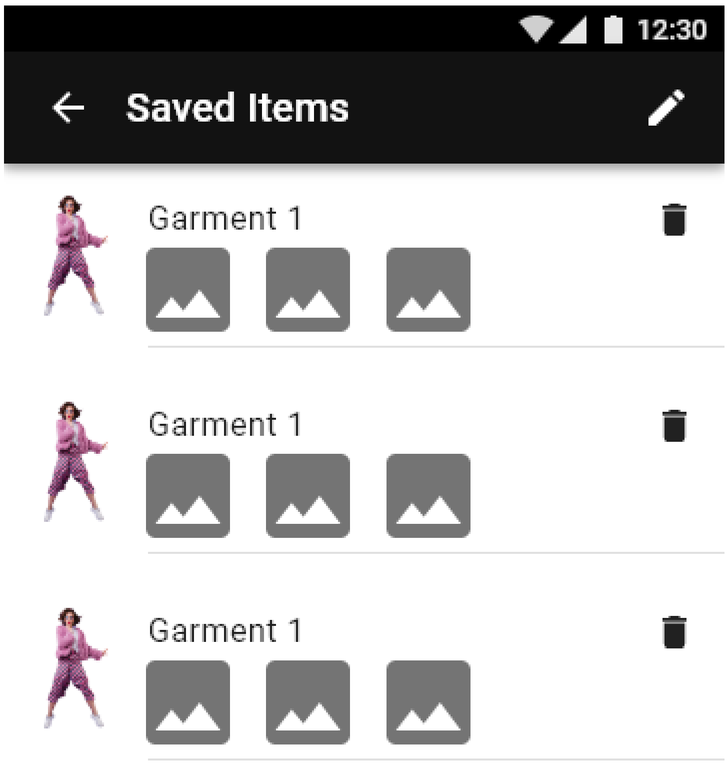
GARMENT RECOMMENDATIONS

Off

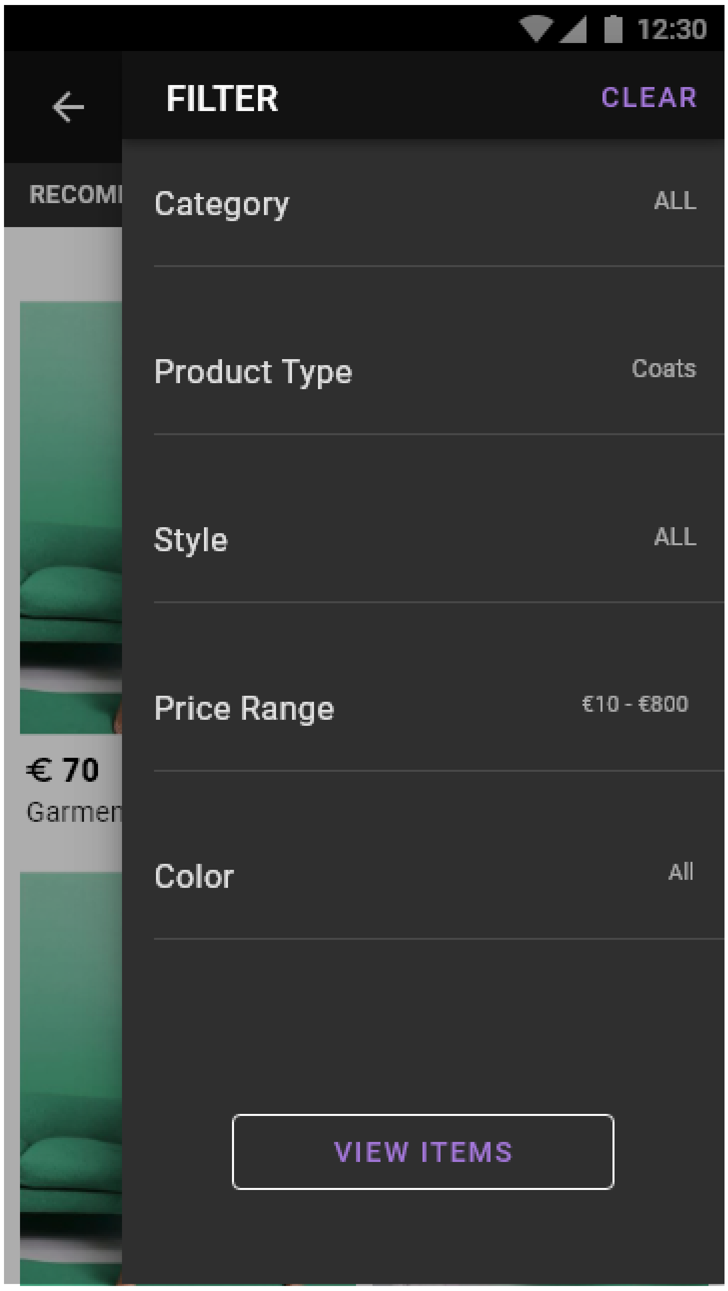
User Account Sidebar View



Browse Garments Grid View

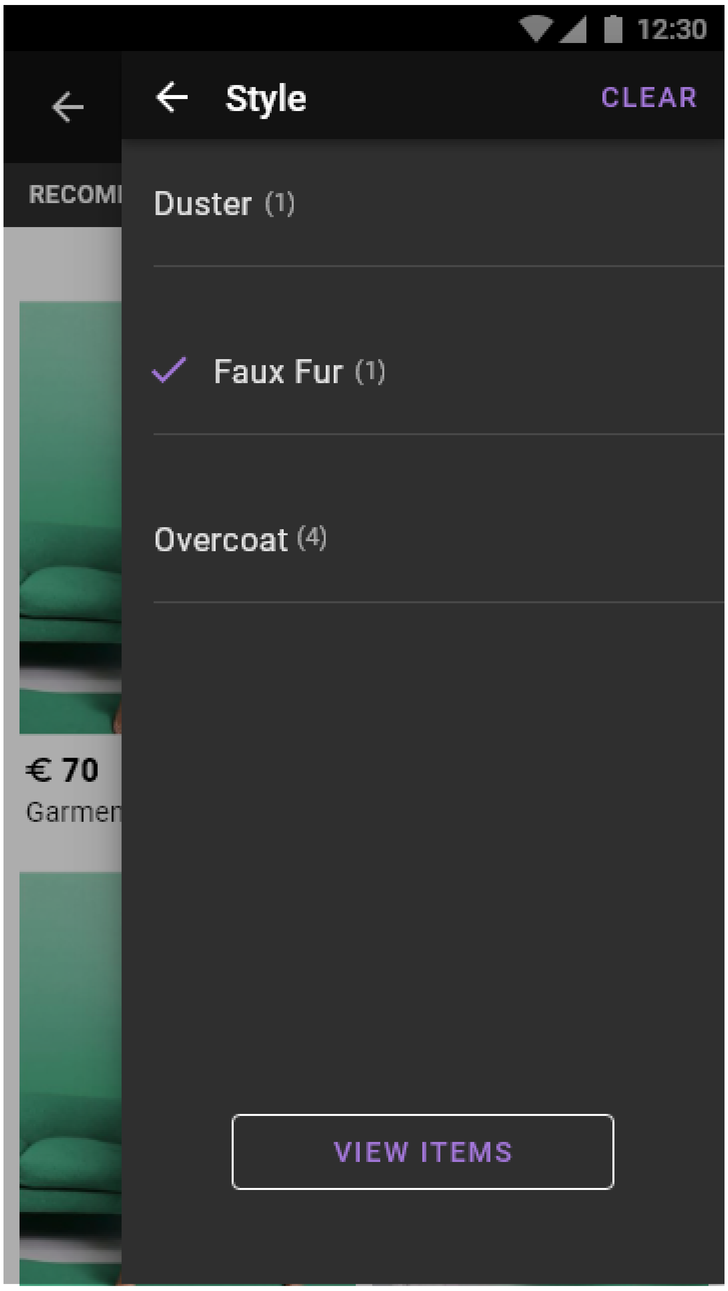


Saved Items View

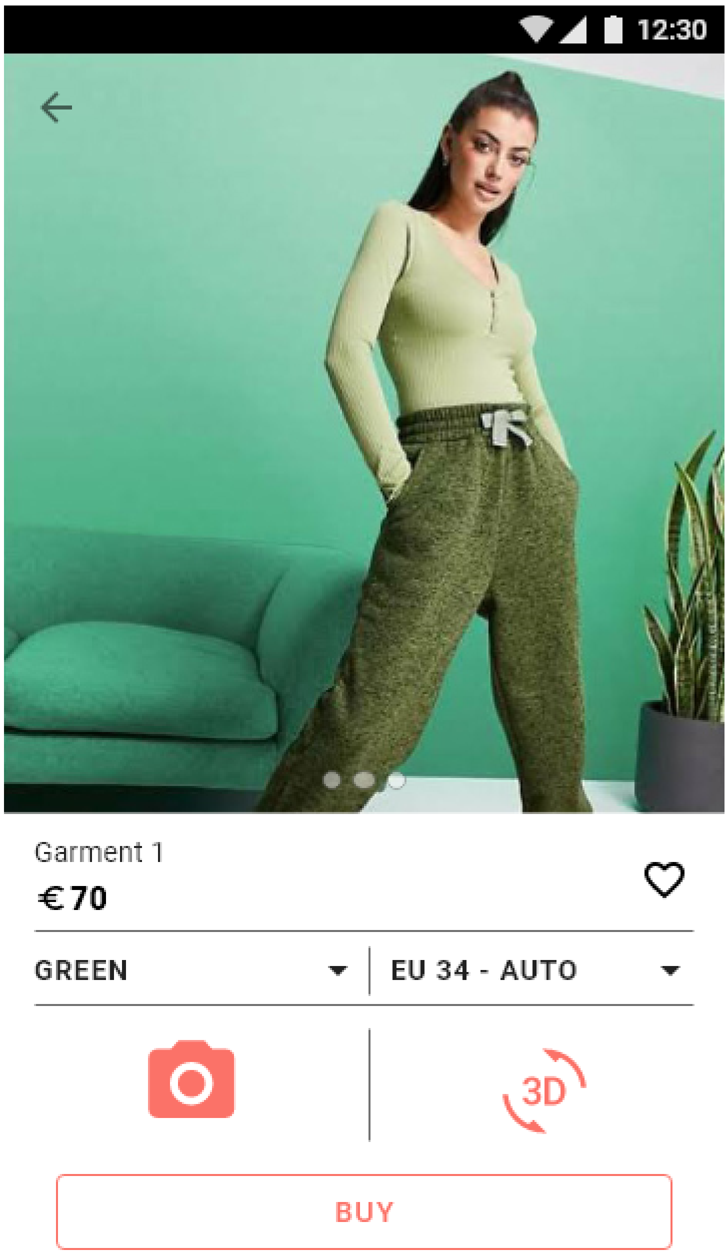


Category	ALL
Product Type	Coats
Style	ALL
Price Range	€10 - €800
Color	All

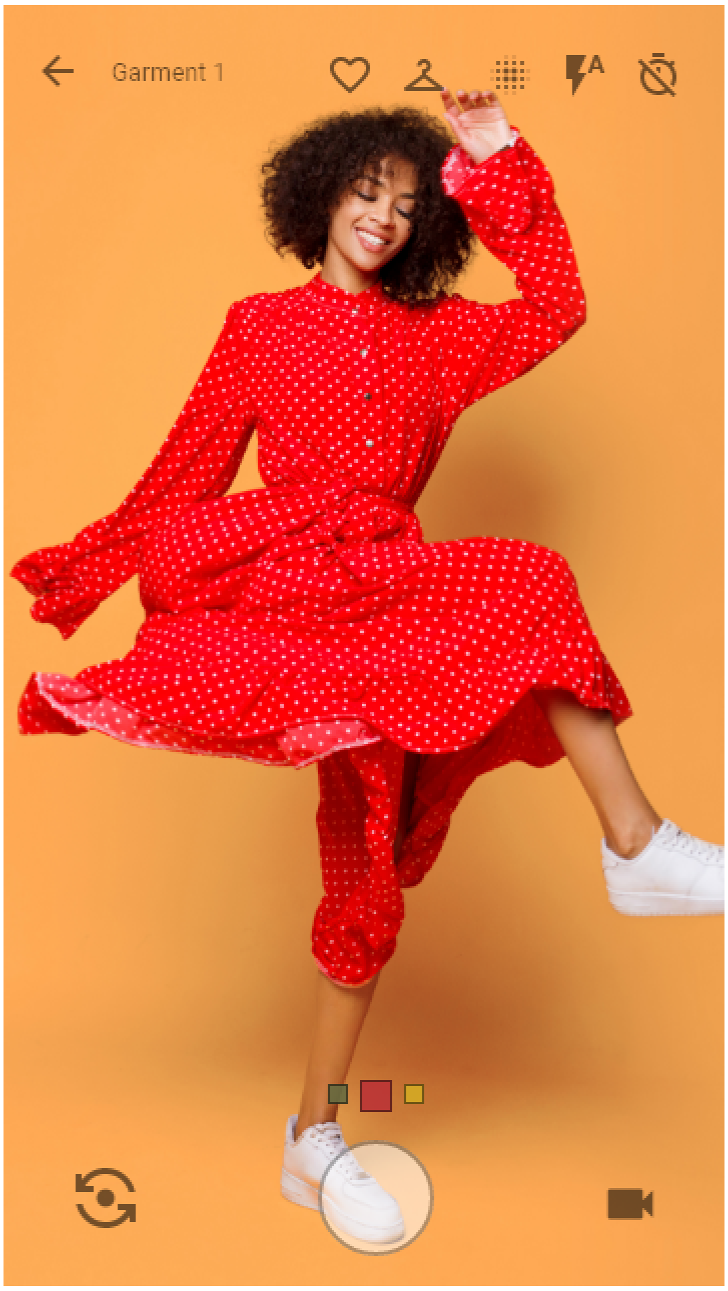
Filter Sidebar 1



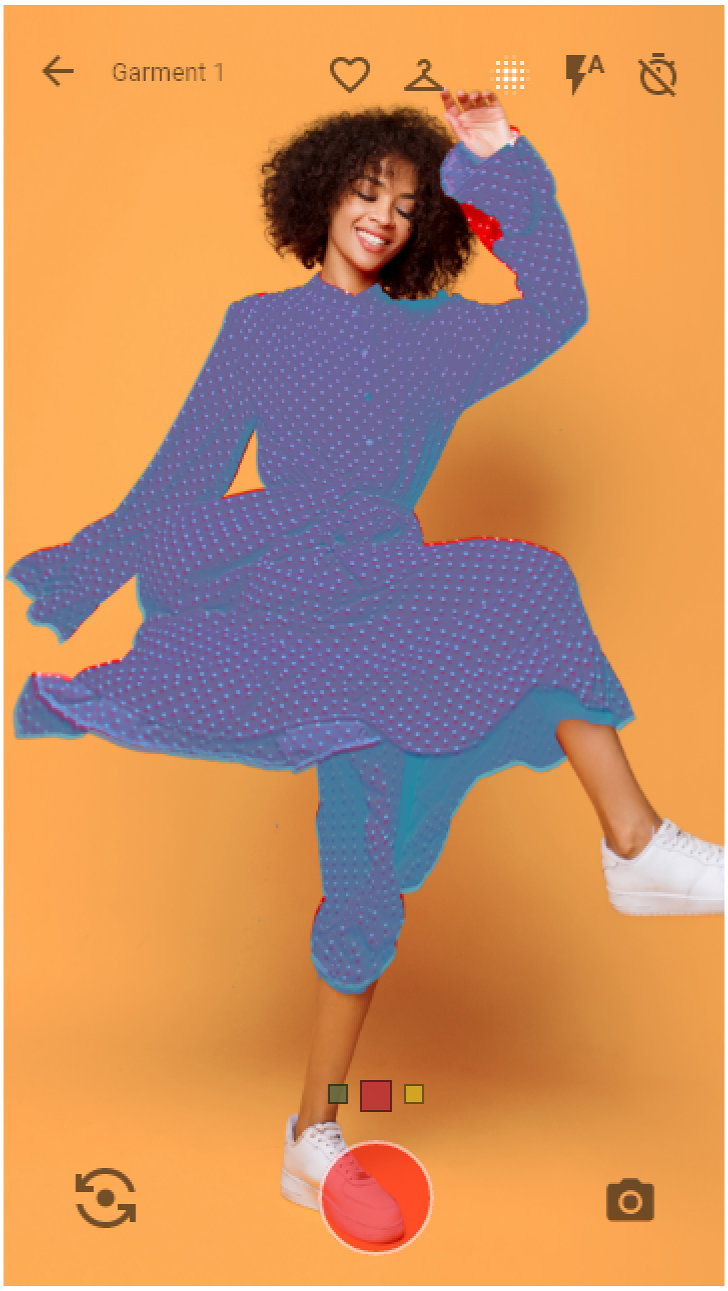
Filter Sidebar 2



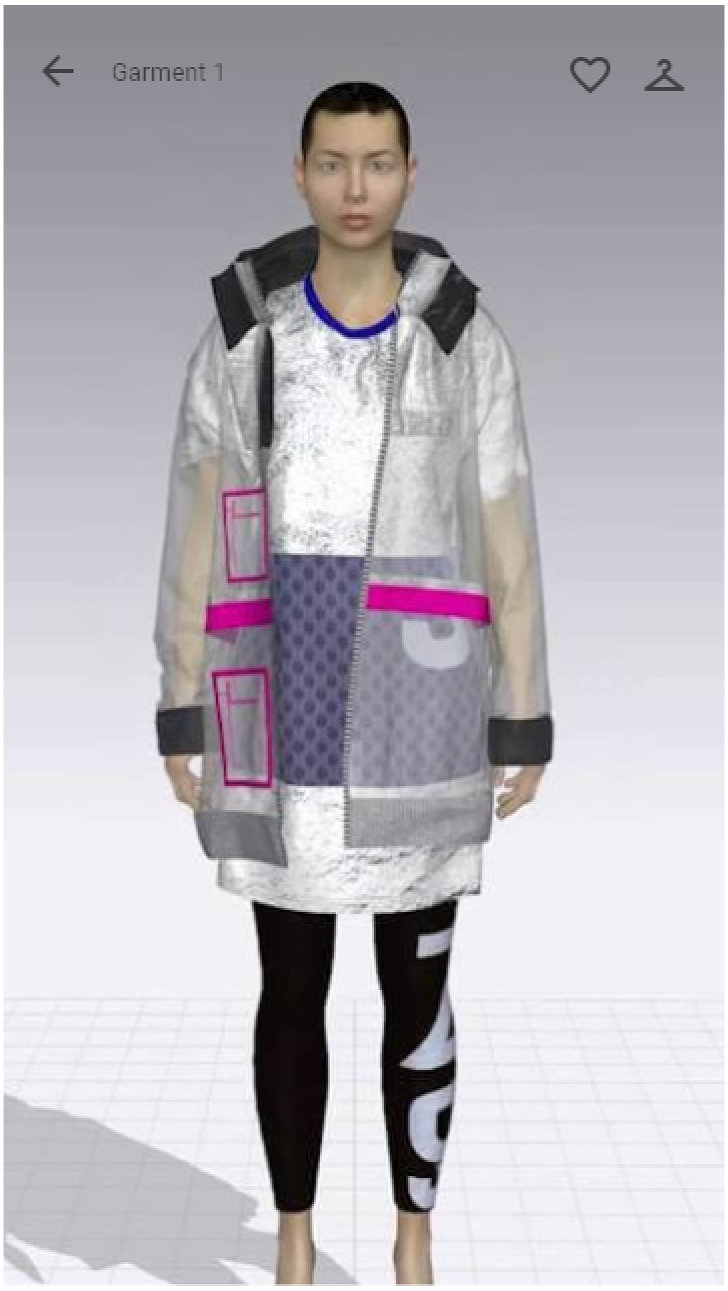
Garment Detailed View



Real Time Fitting View 1



Real Time Fitting View 2 - Tension map enabled



3D Preview View

12.5.1 Usage Scenario 1.0: Sign up - Sign in

Use Case id: 3.1.0

Primary Actor: End user (Consumer)

Description

The user must sign up or sign in to use the app.

Pre-Conditions:

None.

Task Sequence:

The user opens the app and its greeted with a sign up / sign in screen. Providing a combination of an e-mail and password, the user can sign up for an account. While signing up the user must also provide birth date, height and weight. The user can alternatively sign in using an existing account.

12.5.2 Usage Scenario 2.0: Sidebar User Account Actions

Use Case id: 3.2.0

Primary Actor: End user (Consumer)

Description

The user can perform a variety of actions in the User Account sidebar.

Pre-Conditions:

None.

Task Sequence:

After logging in and while on the "Select Category" screen, the user can press on the 'hamburger' icon button in the top left, or slide the screen to the right from the left edge, for the options panel to appear. There are various actions the user can perform: -

- Sign out
- Update personal information (Age, Height, Weight)
- Toggle Units (Metric / Imperial)
- Toggle Garment Reccomendations
- Scan body

12.5.3 Usage Scenario 2.0: Sidebar User Account Actions

Use Case id: 3.2.0

Primary Actor: End user (Consumer)

Description

The user can perform a variety of actions in the User Account sidebar.

Pre-Conditions:

None.

Task Sequence:

After logging in and while on the “Select Category” screen, the user can press on the ‘hamburger’ icon button in the top left, or slide the screen to the right from the left edge, for the options panel to appear. There are various actions the user can perform: -

- Sign out
- Update personal information (Age, Height, Weight)
- Toggle Units (Metric / Imperial)
- Toggle Garment Reccomendations
- Scan body

12.5.4 Usage Scenario 2.1: Scan Body

Use Case id: 3.2.1

Primary Actor: End user (Consumer)

Description

Users can use the app to scan their bodies to enable accurate fitting.

Pre-Conditions:

None.

Task Sequence:

From the ‘Select Category’ screen the user can tap on the hamburger icon on the top left to make the sidebar appear. There the user can tap on the ‘Scan body’ button to begin scanning.

12.5.5 Usage Scenario 3.0: Browse - Filter - Sort Garments

Use Case id: 3.3.0

Primary Actor: End user (Consumer)

Description

The user can browse garments and refine the browsing experience using sorting and filtering.

Pre-Conditions:

None.

Task Sequence:

After logging in in the app the user will be in the “Select a Category” screen. By selecting a base category, a grid of garment item thumbnails will appear.

The user can further refine this grid view by sorting or filtering.

To sort garment items:

- The user can tap on the left dropdown component near the top of the screen. The items can be sorted by recommendations or price (or …).

To filter garment items:

- The user can tap on the right dropdown component to enable a panel to slide from right to left. In this filtering panel the user can select and enable the filters that are needed, then press on the “View Items” button to make the results appear in the item grid view. By tapping on the clear button on the top right the user can reset the filters to default.

12.5.6 Usage Scenario 4.0: Add items to favorites

Use Case id: 3.4.0

Primary Actor: End user (Consumer)

Description

The user can add a garment to a list of favorites.

Pre-Conditions:

None.

Task Sequence:

When navigating the garments in the grid view, a user can tap on the ‘unfilled’ heart icon to add that garment item to a favorite list. By tapping it again, it is removed from the list. The garment item can be also added to favorites from the real-time fit view or the 360 preview view.

12.5.7 Usage Scenario 4.1: Favorites management

Use Case id: 3.4.1

Primary Actor: End user (Consumer)

Description

The user can access a section in the app to see and interact with all of the favorite garments.

Pre-Conditions:

None.

Task Sequence:

While being on the garment grid view, the user can tap the heart icon in the top right to access the Favorites view. There the user can perform the following actions:

- Tap on a garment item so the garment item page can load and see all the available information about it.
- Delete a garment item.
- Multi select garment items for deletion.
- Tap on saved photos the user may have snapped when viewing the garment.

12.5.8 Usage Scenario 5.0: Garment Detailed view interactions

Use Case id: 3.5.0

Primary Actor: End user (Consumer)

Description

The user can view the properties of each garment item and perform various actions.

Pre-Conditions:

None.

Task Sequence:

When navigating the garments in the grid view, a user can tap on a garment item to show its detailed view. In this view the user can perform the following actions:

- See all the available photos of the garment item by sliding left or right on the photo component.
- See the title and price of the garment.
- Add the garment to the list of favorites by tapping on the heart icon.
- Switch garment color if applicable.
- Switch size, or see which other sizes are available. Preselected is a size using an auto fit feature.
- View the garment in real-time fitting on the user self, using the phones front or back camera, by tapping on the camera button.
- Preview the garment on a 3D model, by tapping on the 3D button.
- Purchase the garment by tapping on the “Buy” button at the bottom.

12.5.9 Usage Scenario 5.1: Try on a garment - actions (Real Time Fitting)

Use Case id: 3.5.1

Primary Actor: End user (Consumer)

Description

Try on a garment that is available in the app and see the fit in near real time using the camera.

Pre-Conditions:

1. (3.5.0) The user must be on the garment detailed view.

Task Sequence:

While on the garment detailed view, the user can tap on the camera icon to try it on, in real time using the phone cameras. In the real-time fitting view the user can perform the following actions:

- Add the fitted garment to the favorites list
- Change garment size by tapping on the hanger icon. First tap increases the size by one, second tap lowers the size by one. Third tap resets to normal fitting.
- Capture a screenshot.
- Capture a video.
- Change camera Front / Back (Cases where someone else is operating the device).
- Set a timer for photo/video capture.
- Enable the tension map of the loaded garment as a colored layer with transparency on top of the garment.
- If the fitted garment is available in different colors, an indicator is shown to the bottom of the screen. By swiping left or right on the screen, the user can change the color of the garment.

12.5.10 Usage Scenario 5.2: 3D Preview a garment & actions

Use Case id: 3.5.2

Primary Actor: End user (Consumer)

Description

Preview a garment in 3D

Pre-Conditions:

1. (3.5.0) The user must be on the garment detailed view.

Task Sequence:

While on the garment detailed view, the user can tap on the 3D icon to preview it in the 3D space. In the 3D preview view the user can perform the following actions:

- Add the fitted garment to the favorites list.
- Change garment size by tapping on the hanger icon. First tap increases the size by one, second tap lowers the size by one. Third tap resets to normal fitting.
- Manipulate the 3D garment in X,Y,Z axes.

12.6 UC3 - Magic Mirror Admin CLI

12.6.1 Scenario 1: Add Garment FBX file for a product in the catalogue

Actor

3D Designer

Overview

The designer uploads an fbx file to add a garment so that end users can dress up in it. The metadata is looked up from the product feed data by product id.

Preconditions

- The designer must have arranged the panel pieces for the garment on the standard avatar
- The fbx file must be named according to the Odlo file naming convention.

Operation

12.6.2 Scenario 2: Add Garment FBX file for a product not in the catalogue

Actor

3D Designer

Overview

The designer uploads a fbx file to add a garment so that end users can dress up in it. They must provide the metadata.

Preconditions

- The designer must have arranged the panel pieces for the garment on the standard avatar

Metadata :

Field	Description
title	The garment title

uid	A garment unique id (EAN code)
group_id	This is the id that corresponds to the specific garment, regardless of color or size.
gender	`male` or `female`
size	`s`, `m`, `l`, `xl`, `xxl`
color	A string that denotes the color of the garment
photo	A url for a photo depicting the garment
src.file	Firebase Storage url for the 3D file
material	The garment material
product_type.full_type	A string that has the full type (path) of a garment (e.g.: `"Men>Sporting Activity>Running & Trail"`)
product_type.extracted_type_0	A string that features the first segment of a full_type string (if applicable)
product_type.extracted_type_1	A string that features the second segment of a full_type string (if applicable)

product_type.extracted_type_2	A string that features the third segment of a full_type string (if applicable)
price	A price string in euro with two decimals (e.g.: `64.95`)
eshop_link	A link to the ODLO e-shop linking to the specific garment in order for the user to purchase it
additional_information.custom_label_0	Any additional metadata the garment has available (for filtering purposes). Taken from the `custom_label` fields that are inside the .csv file that ODLO provides
additional_information.custom_label_1	Any additional metadata the garment has available (for filtering purposes). Taken from the `custom_label` fields that are inside the .csv file that ODLO provides
additional_information.custom_label_2	Any additional metadata the garment has available (for filtering purposes). Taken from the `custom_label` fields that are inside the .csv file that ODLO provides
additional_information.custom_label_3	Any additional metadata the garment has available (for filtering purposes). Taken from the `custom_label` fields that are inside the .csv file that ODLO provides

13 Appendix G - SDK specifications from https://etryon.gitlab.io/techdocs/

13.1 QuantaCorp Objective-C SDK

13.1.1 QCScanController

```objc
//
//  QCScanController.h
//  QuantaCorp
//
//  Created by Thomas De Wilde on 30/09/2021.
//

#import <UIKit/UIKit.h>
#import <CoreMotion/CoreMotion.h>
#import "QCScanDelegate.h"

extern NSErrorDomain const QCScanErrorDomain;

@interface QCScanController : NSObject

@property (weak) CMMotionManager *motionManager;
@property (weak) id <QCScanDelegate> delegate;

- (instancetype)init NS_UNAVAILABLE;
- (instancetype)initWithMotionManager:(CMMotionManager *)motionManager
                     withScanDelegate:(id <QCScanDelegate>)scanDelegate;

- (void)presentScanView:(UIViewController *)parent
               animated:(BOOL)animated
             completion:(void (^ __nullable)(void))completion;
- (void)dismissScanView;

@end
```

13.1.2 QCApiSession

```
//
//  QCApiSession.h
//  QuantaCorp
//
//  Created by Thomas De Wilde on 14/09/2021.
//

#import <Foundation/Foundation.h>
#import "QCApiToken.h"
#import "QCCreateBodyDTO.h"
#import "QCPicture.h"

extern NSErrorDomain const QCApiErrorDomain;

@interface QCApiSession : NSObject

@property (nonatomic) QCApiToken *token;

- (instancetype)init NS_UNAVAILABLE;
- (instancetype)initWithClient:(NSString *)client
                     andSecret:(NSString *)secret;

typedef void(^QCApiSessionTokenBlock)(QCApiToken *token, NSError *error);
typedef void(^QCApiSessionIdentifierBlock)(NSNumber *identifier, NSError *error);
typedef void(^QCApiSessionSuccessBlock)(BOOL success, NSError *error);

- (void)authenticateWithUsername:(NSString *)username
                     andPassword:(NSString *)password
                    withCallback:(QCApiSessionTokenBlock)callback;

- (void)authenticateWithRefreshToken:(NSString *)refreshToken
                        withCallback:(QCApiSessionTokenBlock)callback;

- (void)createBody:(QCCreateBodyDTO *)body
      withCallback:(QCApiSessionIdentifierBlock)callback;

- (void)createScanForProject:(NSNumber *)projectId
```

```
                     andBody:(NSNumber *)bodyId
            withFrontPicture:(QCPicture *)frontPicture
              andSidePicture:(QCPicture *)sidePicture
                withCallback:(QCApiSessionIdentifierBlock)callback;

- (void)createCallbacks:(NSDictionary *)callbacks
                forScan:(NSNumber *)scanId
           withCallback:(QCApiSessionSuccessBlock)callback;

@end
```

13.1.3 QCScanDelegate

```
//
//  QCScanDelegate.h
//  QuantaCorp
//
//  Created by Thomas De Wilde on 28/09/2021.
//

#import <Foundation/Foundation.h>
#import "QCPicture.h"
#import "QCScanError.h"

@protocol QCScanDelegate <NSObject>

@optional
- (void)scanViewWillAppear;
- (void)scanViewWillDisappear;

@required
- (void)didCancelScanCapture:(QCScanError *)reason;
- (void)didCaptureFrontPicture:(QCPicture *)picture;
- (void)didCaptureSidePicture:(QCPicture *)picture;

@end
```

13.1.4 QCApiToken

```
//
//  QCApiToken.h
//  QuantaCorp
//
//  Created by Thomas De Wilde on 14/09/2021.
//

#import <Foundation/Foundation.h>
#import "QCJsonSerializableObject.h"

@interface QCApiToken : NSObject

- (instancetype)init NS_UNAVAILABLE;
- (instancetype)initWithAccessToken:(NSString *)accessToken
                       refreshToken:(NSString *)refreshToken
                          expiresIn:(NSTimeInterval)seconds
                          tokenType:(NSString *)tokenType;
- (instancetype)initWithAccessToken:(NSString *)accessToken
                       refreshToken:(NSString *)refreshToken
                     expirationDate:(NSDate *)expirationDate
                          tokenType:(NSString *)tokenType;

@property (readonly) NSString *accessToken;
@property (readonly) NSString *tokenType;
@property (readonly) NSDate *expirationDate;
@property (readonly) NSString *refreshToken;
@property (readonly) BOOL isExpired;

@end
```

13.1.5 QCCreateBodyDTO

```
//
//  QCCreateBodyDTO.h
//  QuantaCorp
//
//  Created by Thomas De Wilde on 16/09/2021.
//

#import <Foundation/Foundation.h>
#import "QCJsonSerializableObject.h"

@interface QCCreateBodyDTO : NSObject <QCJsonSerializableObject>

- (instancetype)init NS_UNAVAILABLE;
- (instancetype)initWithCompany:(NSNumber *)companyId
                    withProject:(NSNumber *)projectId
                      withAlias:(NSString *)alias
                     withGender:(NSString *)gender
                     withHeight:(NSNumber *)height;

@property NSString *alias;
@property NSString *firstName;
@property NSString *lastName;
@property NSString *crmId;
@property NSNumber *userId;
@property NSNumber *companyId;
@property NSNumber *height;
@property NSNumber *weight;
@property NSString *gender;
@property NSString *notes;
@property NSNumber *linkToProject;
@property NSDictionary *custom1;
@property NSDictionary *custom2;
@property NSDictionary *custom3;
@property NSDictionary *custom4;
@property NSDictionary *custom5;
@property NSDictionary *custom6;
@property NSDictionary *custom7;
```

```
@property NSDictionary *custom8;
@property NSDictionary *custom9;
@property NSDictionary *custom10;

@end
```

13.1.6 QCPicture

```
//
//  QCPicture.h
//  QuantaCorp
//
//  Created by Thomas De Wilde on 16/09/2021.
//

#import <Foundation/Foundation.h>

@interface QCPicture : NSObject

@property NSURL *location;
@property NSDictionary *metadata;
@property (readonly) NSData *png;

@end
```

13.1.7 QCScanError

```
//
//  QCScanError.h
//  QuantaCorp
//
//  Created by Thomas De Wilde on 04/10/2021.
//

typedef NS_ENUM(NSUInteger, QCScanError) {
    QCScanErrorBadInputDevice,
    QCScanErrorBadOutputDevice
};
```