DASH-Fit

Software Design (SD)

GitHub Repository

BY

Team Lead: ***Adeen Atif (22999)***

Member(s):

M Saad Tariq (22947)

Mir Hamza Ali (22767)

Dania Ahmed (22795)

PROJECTS COMMITTEE (PC)

SUPERVISOR: ***Dr Syed Ali Raza***

INDUSTRY SUPERVISOR: ***Arsalan Rashid (Systems Ltd)***

SUBMITTED TO

PROJECTS Manager – FYP

ON

January 22nd, 2024

IBA SMCS

School of Mathematics and Computer Science

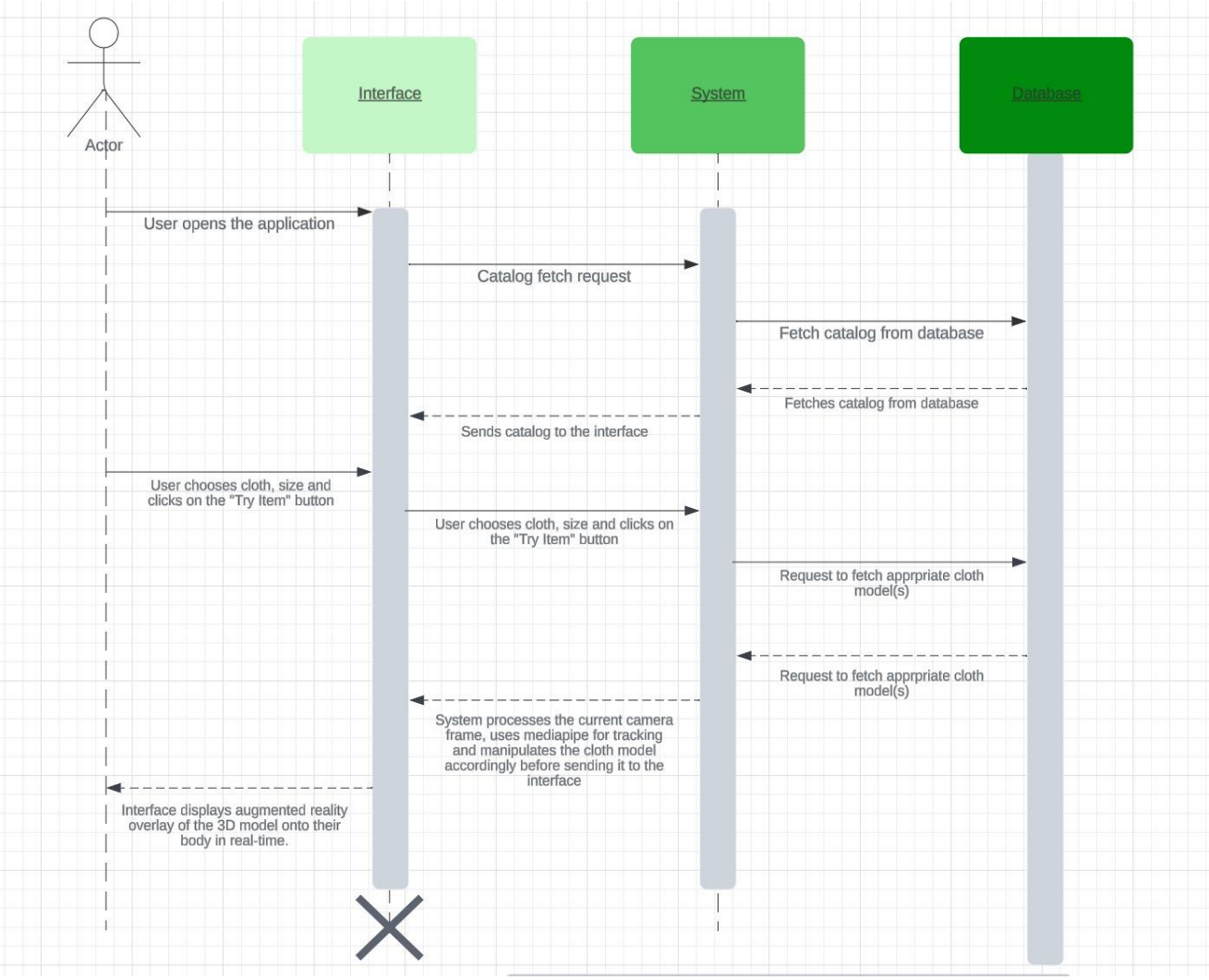
Table of Contents

System Design

1. Interaction Diagrams

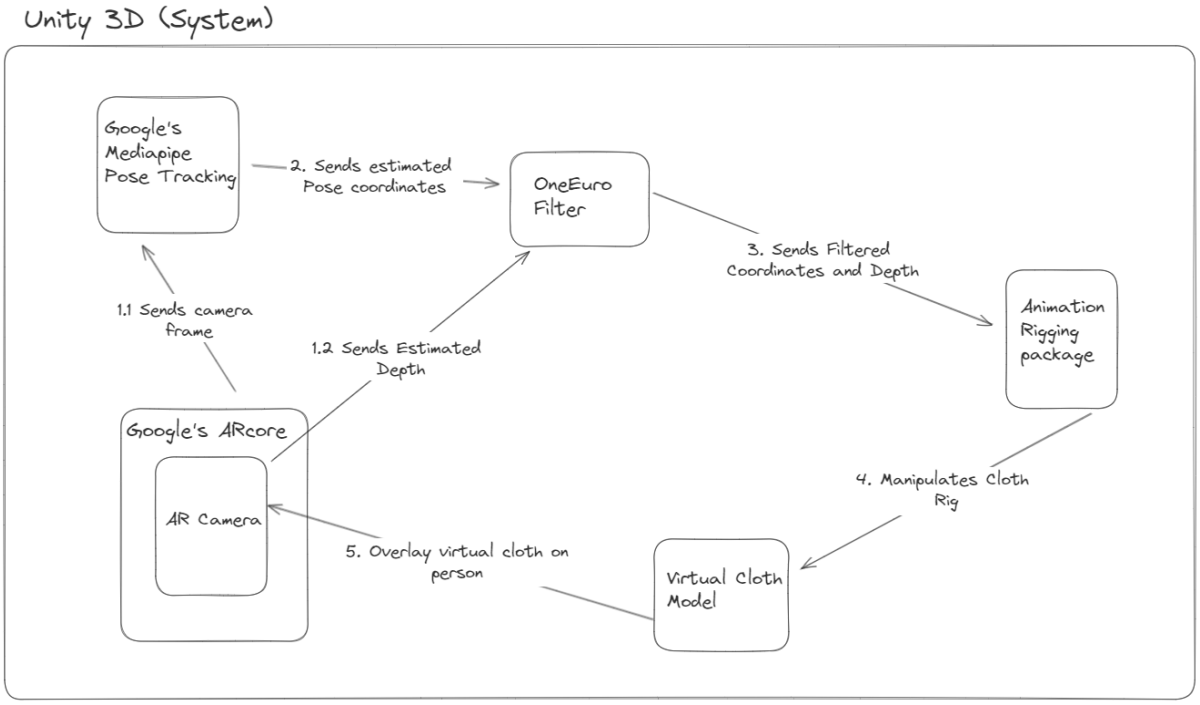
Either sequence diagrams or UML of the fully dressed use cases.

1.1 Sequence Diagram



Sequence diagram for the interaction of objects (interface, system, and database) within the system

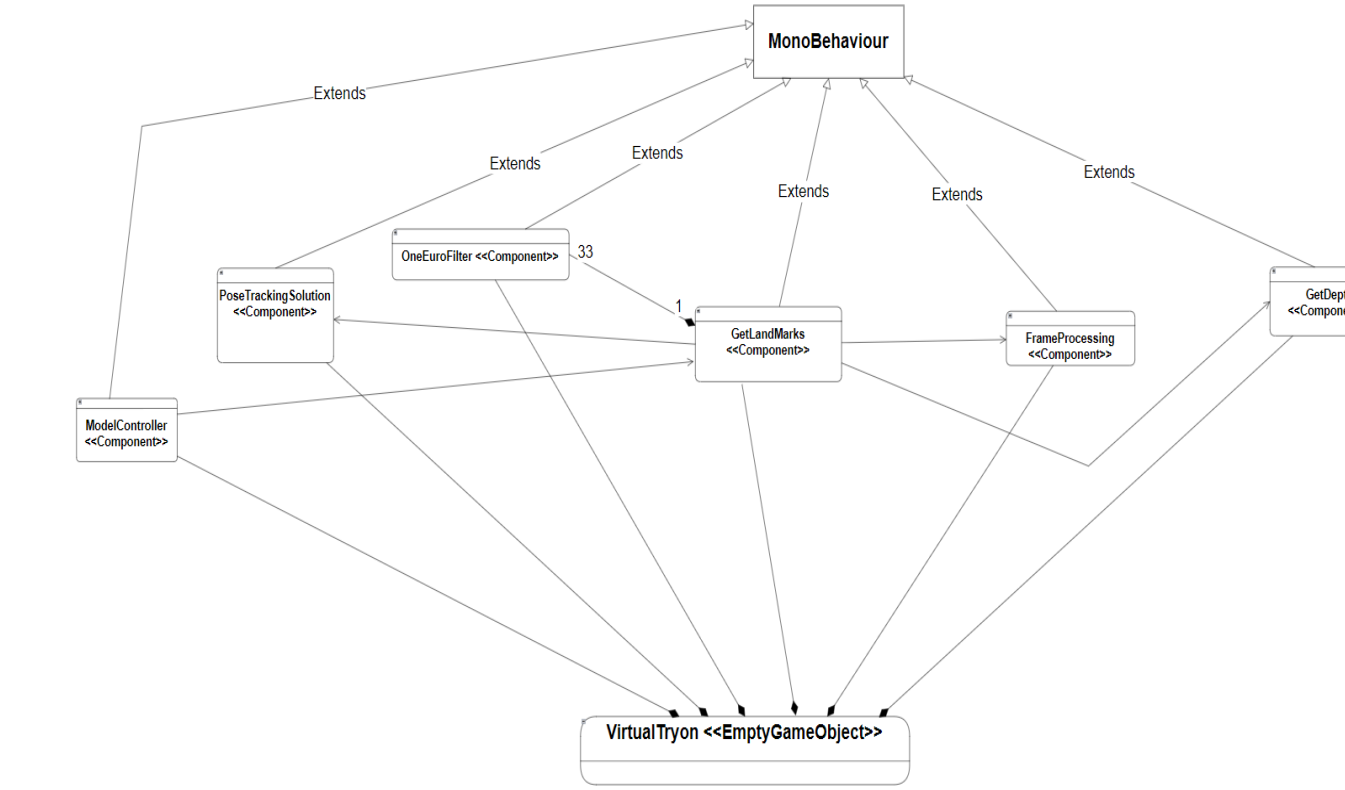
1.2 System Internal Collaboration Diagram



Flow diagram to represent the working of the system sequentially.

2. Class Diagram and Interface Specification

2.1 Overview Class Diagram

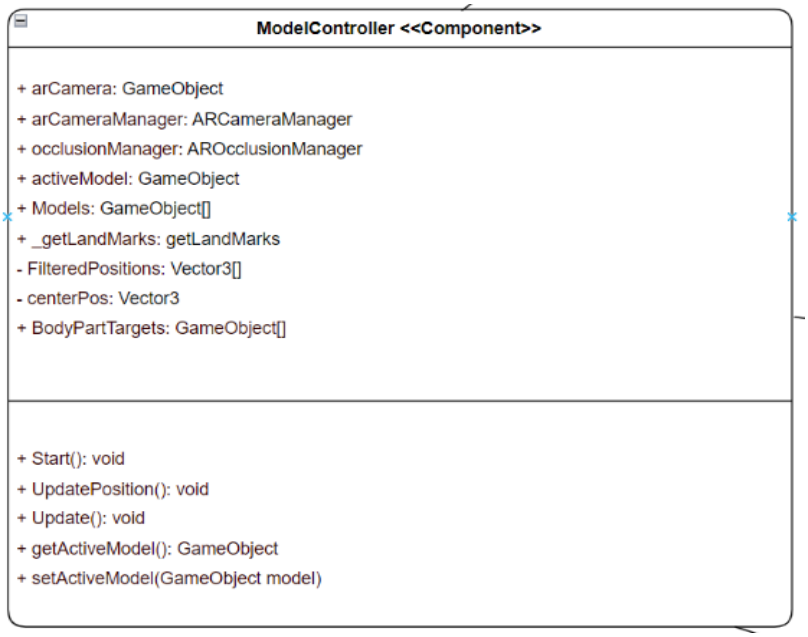


Class diagram consisting of the main components of the system

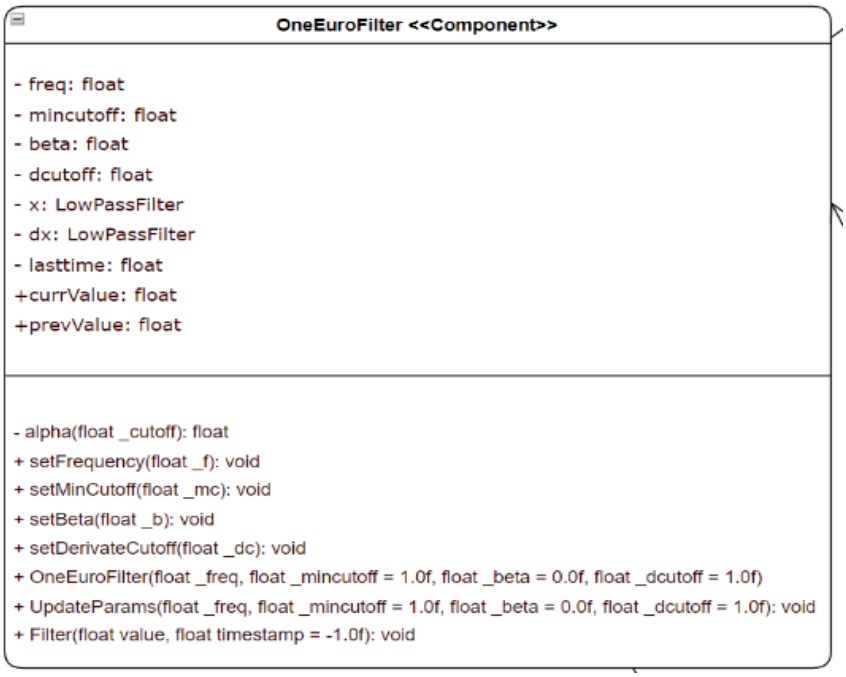
Note: "MonoBehaviour" is base class that many Unity scripts derive from. Details of it can be found here: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

"VirtualTryOn" is a Unity Empty 'GameObject', it is only composed of the scripts in our application, and it does not have any properties of its own. The scripts need to be attached to a GameObject in order to run.

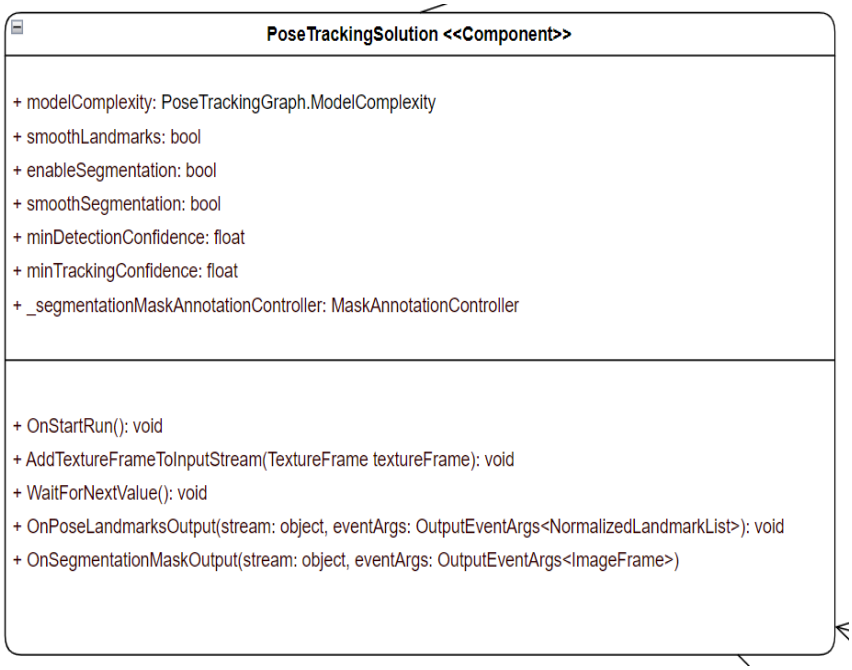
2.2 Partial Class Diagrams



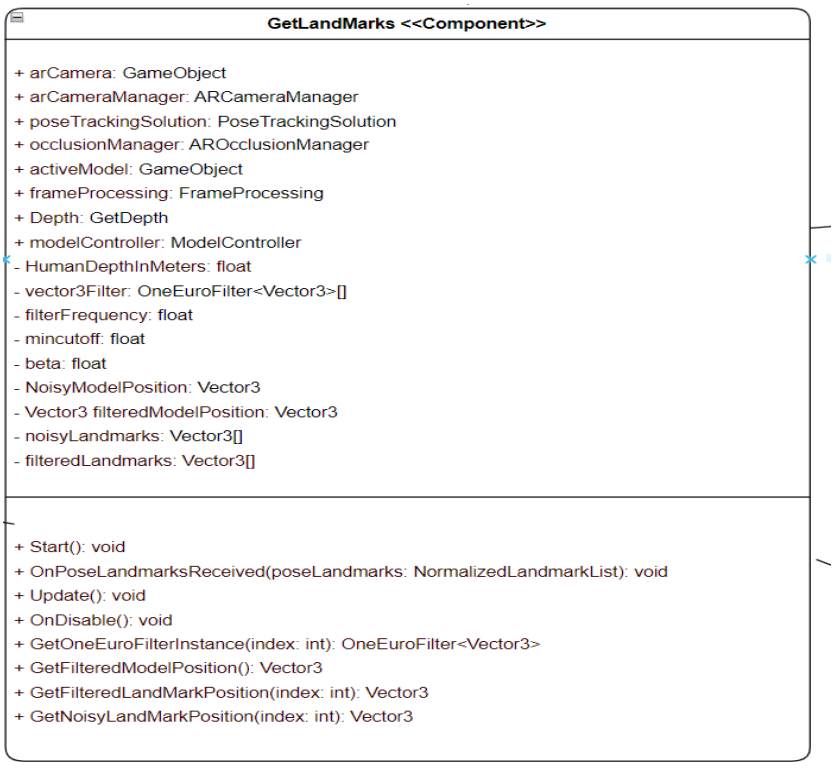
ModelController Class



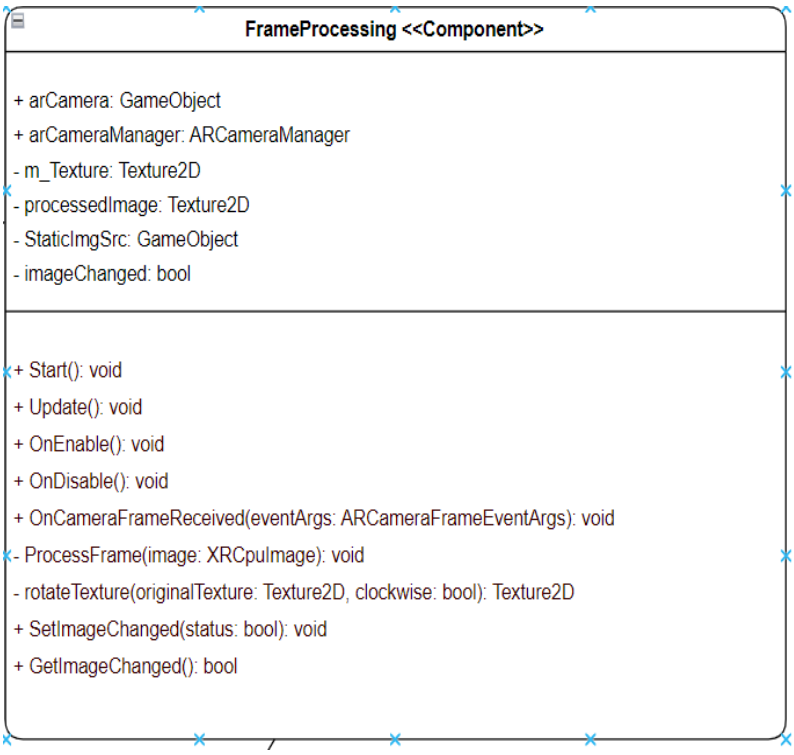
OneEuroFilter Class



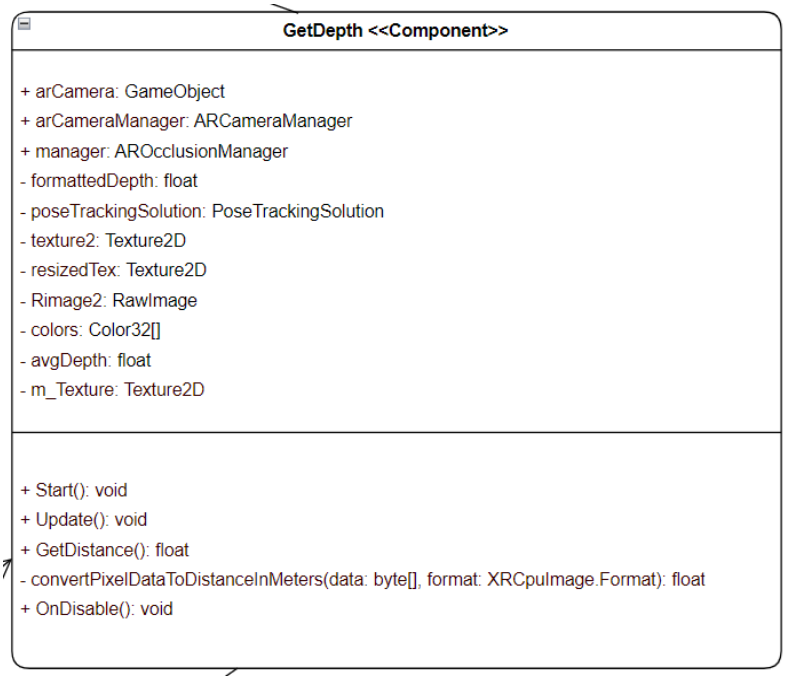
PoseTrackingSolution Class



GetLandMarks Class



FrameProcessing Class



GetDepth Class

3. System Architecture and System Design

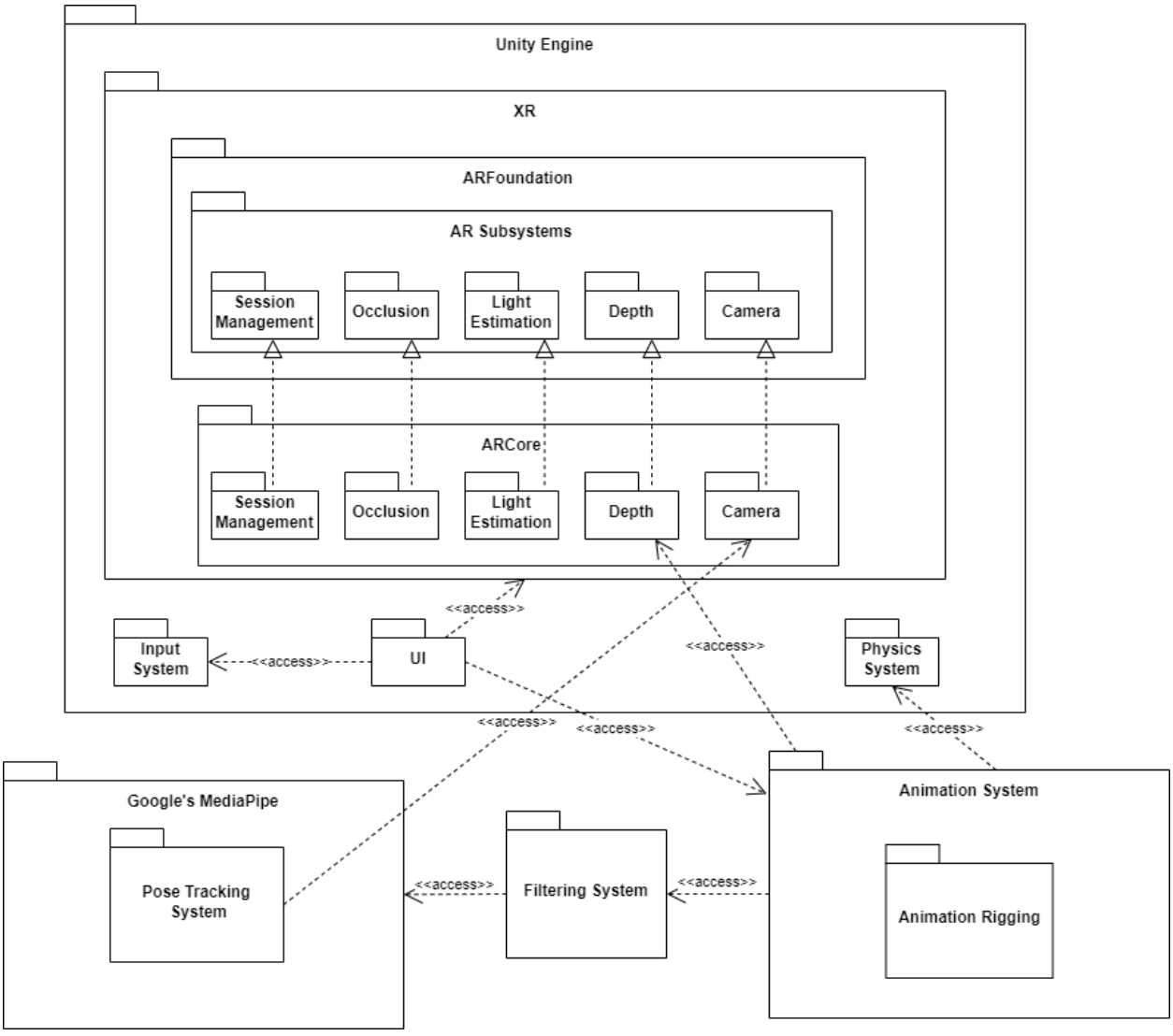
3.1. Architectural Style

Software Architectural Style is a hybrid between component-based architecture and event-driven architecture.

Unity extensively uses a component-based architecture. Our AR app consists of various components (like 3D models, scripts, AR tracking components) that are attached to game objects. These are loosely coupled, modular and reusable components.

Since our applications works in real time, it also heavily relies on events (e.g., user interactions, tracking pose changes and movements). Functions are triggered in response to user interactions/movements or changes in the AR environment.

3.2. Identifying Subsystems



- **UnityEngine**: This is a Unity package which contains multiple packages/subsystems such as Input, User Interface, Physics and Augmented Reality subsystems.
- **XR**: Unity supports XR development on multiple platforms and devices through its XR plug-in framework. This XR package contains the packages required to build XR applications such as Augmented reality, Mixed Reality and Virtual Reality.
- **ARFoundation**: The AR Foundation package contains interfaces for AR features, but doesn't implement any features itself. To use AR Foundation on a target platform, you also need a separate provider plug-in package for that platform like ARCore for android.
- **AR Subsystems**: The AR-related subsystems are defined in this package. This package only provides the interface for various subsystems.
- **Session Management:** Refers to an instance of AR and controls the lifecycle of all AR-related subsystems.
- **Occlusion:** Responsible for achieving realistic-looking blending of augmented and real-world content by making sure that nearby physical objects occlude virtual content that is located behind them in the shared AR space.
- **Light Estimation:** Responsible for dynamically adjusting virtual object lighting in AR scenes to match the real-world lighting conditions.
- **Depth:** The depth subsystem is an interface into depth information detected in the scene.
- **Camera:** The camera subsystem manages a hardware camera on the AR device.
- **ARCore:** The ARCore XR Plug-in package enables ARCore support via Unity's multi-platform XR API. This package implements the AR subsystems present in the ARFoundation.
- **Input System:** Allows users to control your game or app using a device, touch, or gestures.
- **UI:** Responsible for displaying content of the AR world to the user.
- **Physics System:** Responsible for simulating physics (cloth physics in our case) in project.
- **Google's Mediapipe:** This package contains the various solutions for perception-based tasks such as hand tacking, face detection, pose estimation and more.
- **Pose Tracking System:** Responsible for human pose estimation. The pose landmarker model tracks 33 body landmark locations, representing the approximate location of the following body parts. It also provides human segmentation image.
- **Filtering System:** Responsible for reducing jitter and lag in the coordinates detected by the pose tracking system. This contains OneEuro Filter which is a low pass filter and helps in reducing noise from the signals.
- **Animation System:** Responsible for creating and controlling dynamic, interactive animations within Unity applications.

- **Animation Rigging:** Provides advanced rigging tools and features for creating more sophisticated character/cloth animations. This is responsible for manipulating and adding constraints to the rig of the clothes' models.

3.3. Mapping Subsystems to Hardware

Since our Unity project is not a client-server-based project or a multi-tier project but rather has more of a monolithic structure, the system, with all of its subsystems, runs on a single machine which is the client's Android device.

3.4. Persistent Data Storage

3.4.1 Data Storage

Considering our current scope of application, we will not be using the database unless necessary, since a limited amount of rigged 3D cloth models are available. We will be storing our 3D models in the local file system of the application for now. In case of scaling and increasing number of models, we will transfer the models to Firebase cloud file storage because it has good integration support for Unity.

3.4.2 File Format Description

Our 3D models are stored in the FBX file format. FBX files are a type of 3D model file created using the Autodesk FBX software. FBX files typically contain mesh, material, texture, and skeletal animation data. It encapsulates 3D models, animations, textures, and scene information, making it ideal for the diverse needs of our AR application. FBX files provide a comprehensive way to integrate detailed models and animations into the AR environment. They are particularly well-suited for virtual and augmented realities because they support a wide range of data types, including geometry, animation, skinning, and lighting. The FBX can be represented on-disk as either binary or ASCII data; its SDK supports both reading and writing. While neither of the formats is documented, the ASCII format is a tree structured document with clearly named identifiers. Aside from the FBX file, textures for the models will be stored as PNG files.

3.5. Network Protocol

System works offline for now. If we scale and expand, where our 3D models are stored on cloud, then internet will be required, in such case Firebase Storage operations, such as uploading and

downloading files (including 3D models in formats like FBX), are performed over HTTP/HTTPS.

3.6. Global Control Flow

3.6.1 Execution Order

Our application is primarily event-driven, responding to user inputs, AR tracking events and pose landmark and segmentation events. This approach is integral for interactive features like pose detection and AR rendering in real time. When a frame is received, an event is triggered by ARCamera. After the frame has been processed, it is given to the Mediapipe's pose detection graph which raises an event if a human pose is detected. Similarly, ARCore's depth and environment images raise event and then they are processed.

Certain initialization and setup processes follow a linear execution flow, ensuring that essential components (like ARCore initialization, model loading) are ready before user interaction begins. After human pose coordinates are detected along with depth information, instances of OneEuroFilter filters the coordinates, and Animation Rigging package manipulates the rig in linear execution order.

3.6.2 Time Dependency

Certain processing and computations have been made time dependent for optimizing performance and resource usage in AR functionalities. Frame Processing, depth calculation and gradient (for arm movements) calculations are all subject to time dependency. Frame Processing is performed every 0.1 seconds (tentative), depth calculation is performed every 5 seconds and gradient calculation is performed every 0.1 seconds. This helps in reducing noisy signals and saves computational resource. Unity Coroutines have been used for time dependency. A coroutine allows you to spread tasks across several frames. In Unity, a coroutine is a method that can pause execution and return control to Unity but then continue where it left off on the following frame. Unity's coroutine system is employed extensively for managing sequences that span multiple frames. Coroutines allow for creating delays and scheduling tasks without blocking the main thread.

3.6.3 Concurrency

The application operates on a single-threaded model, with all processes running on Unity's main thread so there are no concurrency issues. Given the single-threaded nature, careful attention is paid to optimizing performance and ensuring responsive interactions without overburdening the main thread.

3.7. Hardware/Software Requirements

3.7.1 Hardware requirements:

Mobile Device Specifications:

- Processor: A modern, multi-core processor (e.g., Qualcomm Snapdragon 845 or equivalent).
- RAM: minimum 4 GB
- Camera: A working and good-quality rear-facing camera
- Sensors: Gyroscope and accelerometer
- Storage Space: A minimum of 100 MB free space could be necessary. (Note: this is assuming that the 3D models are not saved on the device but rather on a cloud file system or database. If the models are stored on the device then this storage can be increase significantly, depending upon how many we models we store).

3.7.2 Software requirements:

Operating System: Android minimum API level: Android 7.0 'Nougat' (API Level 24) or higher.

Additional requirement: Device should also support ARcore Depth API. As of May 2023, approximately 89% of active devices support the Depth API and the list of supported devices can be found here: https://developers.google.com/ar/devices

4. Algorithms and Data Structures

4.1 Algorithms:

Algorithm 1: 1€ filter

EXT: First time flag: *firstTime* set to *true*
Data update rate: *rate*
Minimum cutoff frequency: *mincutoff*
Cutoff slope: *beta*
Low-pass filter: *xfilt*
Cutoff frequency for derivate: *dcutoff*
Low-pass filter for derivate: *dxfilt*
IN : Noisy sample value: *x*
OUT: Filtered sample value

```
if firstTime then
    firstTime ← false
    dx ← 0
else
    dx ← (x - xfilt.hatxprev()) * rate
end
edx ← dxfilt.filter(dx, alpha(rate, dcutoff))
cutoff ← mincutoff + beta * |edx|
return xfilt.filter(x, alpha(rate, cutoff))
```

Algorithm 2: *Filter* method of Low-pass filter

EXT: First time flag: *firstTime* set to *true*
IN : Noisy sample value : *x*
Alpha value : *alpha*
OUT: Filtered value

```
if firstTime then
    firstTime ← false
    hatxprev ← x
end
hatx ← alpha * x + (1 - alpha) * hatxprev
hatxprev ← hatx
return hatx
```

Algorithm 3: Alpha computation

IN : Data update rate in Hz: *rate*
Cutoff frequency in Hz: *cutoff*
OUT: Alpha value for low-pass filter

```
tau ← 1.0 / (2*π*cutoff)
te ← 1.0 / rate
return 1.0 / (1.0 + tau/te)
```

OneEuroFilter Algorithm

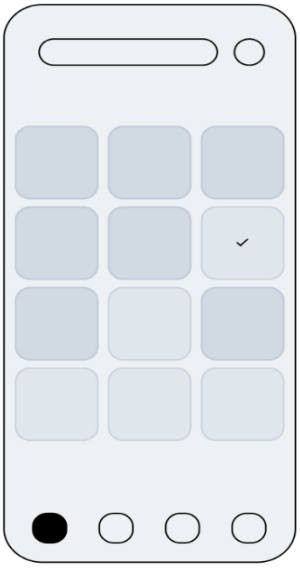
4.2 Data Structures:

1. Arrays: These will be used for holding multiple components such as OneEuroFilter instances, 3d models and colors etc.
2. Vector3 (Unity Engine's data structure): This data structure will contain float values of x, y and z coordinates which are being used for position and landmark coordinates.
3. Color (Unity Engine data structure): This data structure contains float values of R, G, B representing pixel values, used for frame/texture processing and texture manipulation.
4. Queues: This will be used to store pool of frames/textures.

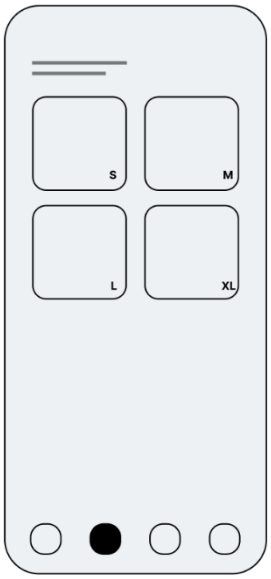
5. User Interface Design and Implementation

Following are the interface designs for the system:

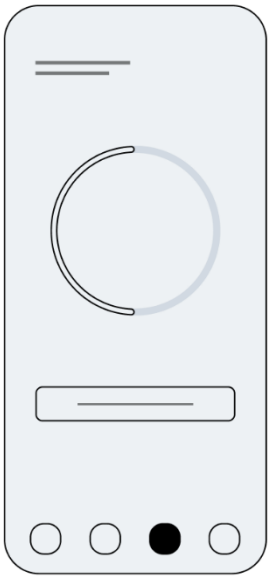
Upon opening the app a landing page appears. Once the user clicks the arrow at the bottom of the page, the home screen appears.



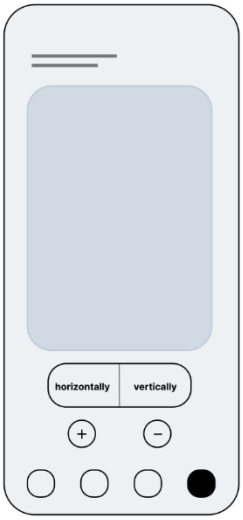
The home page shows a catalogue of all the clothing items available. User can click on items to try them on.



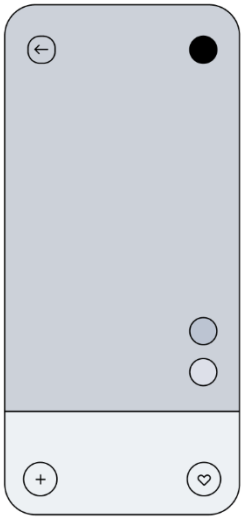
After choosing the item the user is redirected to another screen where they can select the size they want from the given options to try on



Upon choosing next the screen displays the loading page while the system detects the user and overlays the clothing item onto him/her



After a waiting period of 5-10 seconds, the camera adjustment screen appears which gives a preview of how the item looks on the user while allowing them to make minor adjustments (move the clothing model up/down, left/ right) to get an accurate fit



Once the user has finalized the adjustments by clicking on the tick button on the previous screen, they can now see the final fit of the clothing item on themselves

6. Design of Tests

Following are the test case for the functionalities of the user role:

Test Case ID	Description	Test Step	Expected Result	Actual Result	Status Pass/Fail
1	Users will be able to view and scroll through the available items	1. User opens the DASH Fit application on their device.	Available clothing items appear in grid style on the screen		
2	Users will be able to select an item from the catalogue.	1. User scrolls through the catalogue. 2. User chooses the desired item and clicks '+' to add to chosen items.	Item is selected successfully, and size options appear		

3	Users will be able to choose a size from the given options	1. User selects an item from the catalog. 2. User selects one of the available size options. 3. User clicks '+' to add to chosen items.	Size is selected successfully		
4	User will be able to select multiple items to switch between while in the try room	1. User selects an item from the catalog. 2. User chooses the desired item and clicks '+' to select it. 3. User returns to the catalog. 4. User selects a second item and clicks '+' to add to chosen items.	The items chosen will be displayed in the corner of the screen		
5	User will be able to view the chosen clothing item(s) on themselves	1. User will click on the 'try on' button in the chosen items section.	The camera will open with the user in the frame and the clothing item overlayed on top.		
6	User will be able to adjust the position of the clothing item once displayed on them	1. User chooses whether adjustment to be made on horizontal or vertical axis. 2. User clicks the '+' or '-' buttons to move the clothes upward/right or downwards/left.	Clothes shifted to the new position on the person		
7	The clothing item will move as per	1. User will click on the 'try on' button	Clothes stay in the same position		

	the user's movements once the item has been overlayed onto the person	in the chosen items section. 2. User will move around forward/backward & left/right	relative to the person's body		
8	The lag between the cloth and the user's movement will be less than 0.5s	1. User will click on the 'try on' button in the chosen items section. 2. User will move around forward/backward & left/right	The time difference between the user and the cloth's movement is less than 0.5s		
9	User will be able to take a screenshot of the screen once they've 'tried it on'	1. User clicks the button with the camera icon at the bottom of the screen.	Screenshot captured and saved to the photo album		

Following are the test case for the functionalities of the admin role:

Test Case ID	**Description**	**Test Step**	**Expected Result**	**Actual Result**	**Status Pass/Fail**
10	Admin will be able to add new items to the catalogue	1. Admin enters the item's pictures, available sizes, colors and their quantity. 2. Admin clicks 'Enter'.	New clothing item added to the catalogue		

11	Admin will be able to remove items from the catalogue	1. Admin chooses the item to be removed. 2. Admin clicks 'Delete'.	Selected clothing item is no longer in the catalogue		

Note: The 'Actual Result' and 'Status' columns are kept empty since the testing hasn't been conducted yet, the table will be filled completely with these two columns once unity testing has been performed.