Digital Human Model for Postural Assessment - Postural Visualisation Application

Khalid Ali (15005070)

University of South Wales
Cardiff and Vale University Health Board

MSc by Research Computer Science

July 2022

Acknowledgements

I I would like to thank my supervisors Dr Janusz Kulon and Dr Marius Miknis for their guidance and patience throughout this entire research project.

II I would also like to thank Dr Adam Partlow and Prof. Colin Gibson for their support of the project, their feedback and all the insight they were able to provide me.

III Another thank you to members of our research group Sebastian Haigh and Elliot Naylor for their feedback and support.

IV Another big thank you to the staff at Cardiff and Vale University Health Board Rehabilitation Engineering Unit for their enthusiasm and providing me insight into their existing practices and systems.

V I would also like to thank the Knowledge Economy Skills Scholarships (KESS) programme and Cardiff and Vale University Health Board for funding this project. KESS is a pan-Wales higher-level skills initiative led by Bangor University on behalf of the HE sector in Wales. It is partly funded by the Welsh Government's European Social Fund (ESF) programme for East Wales.

VI Finally, a thank you to my mother Hayley, grandmother Wendy and all my friends for their moral support over these past two years.

Abstract

Postural assessment is a vital process that enables clinicians to diagnose abnormalities and suggest treatment to improve a patient's quality of life. However, the current methods of assessment and practices of data recording are non-standardised and do not utilise the opportunity modern mobile technology provides to improve the assessment experience for all parties involved.

This project aimed to enable clinicians to enhance their posture assessment process with the use of cutting-edge augmented reality technology to improve how measurements are taken, implemented as part of a complete system that standardises the data recording and transfer process to improve the efficiency of those processes and enables data to be more easily understood by other people.

An Android-compatible mobile application with ARCore technology was developed that provides features for static and real-time capture, to add annotations onto said captures, and to send data to a server with standardised methods. This was achieved by identifying clinician's needs through survey and developing an application to solve the identified problems, ultimately achieving a measurement accuracy of approximately 1 to 1.5cm.

Contents

List of Figures

List of Acronyms

API Application Programming Interface.

AR Augmented reality.

GDPR General Data Protection Regulation.

MDR Medical Devices Regulations.

NHS National Health Service.

PMC Posture and Mobility Centre.

REU Rehabilitation Engineering Unit.

1 Introduction

1.1 Project Aim

This project aims to design and implement a mobile application that can assist NHS Wales Posture and Mobility Centre clinicians with the postural assessment process by giving staff a standardised but flexible platform to acquire and annotate images during an assessment.

1.2 Objectives

I To investigate the present issues of the postural assessment process in the NHS Posture and Mobility Centre and assess how to best meet their needs to improve the process with a mobile application, along with establishing background research on posture assessment and digital human modelling.

II To develop an application capable of acquiring, annotating and serialising data during a posture assessment with the inclusion of a real-time length measurement system. These processes will need to adhere to industry standards and API and be functionally easy to use by clinicians in the field.

III To evaluate the application during and after development to ensure the system is accurate and reliable enough to be a viable supplement or replacement to manual measuring techniques. This will be achieved by demonstrating a prototype to clinicians with an accompanying survey to gather opinions and suggestions, and finally conducting a summative test of the application's real-time measurement accuracy against expected manual measurement margins.

1.3 Risk Assessment

1.3.1 Resource safety

There is always a potential risk of hardware and/or software failure when working with technology that could result in some or all of the project's material being lost. To counteract these types of issues, various backup and data control methods will be employed throughout this project's duration to ensure much of the material is recoverable in the event of local technological failure. This thesis shall be developed on the LaTeX-based Overleaf document preparation system, an in-browser and platform-independent solution that stores project data in the cloud whilst allowing the source material to be exported into other backup means such as Google Drive. Google Drive will also be used to store notes, graphics and periodic backups of the thesis and source code. Git will be

used as the source control system for the project solution's source code. The code will be mirrored to GitHub's cloud storage via commits made upon each feature addition or bug fix.

1.3.2 Software updates

The development software used during this project is subject to updates at the author's discretion. To control the possibility of introducing new variables to factor in or instability in the tools, stable releases of the tools involved will be decided and used throughout the project. As further explained in Section 3.1.1, the use of Unity 2019.3.5f1 and Microsoft Visual Studio 2019 will be maintained throughout the project unless a specific need to update arises.

1.3.3 GDPR

GDPR has potential implications relating to data security and consent. As per its integrity and confidentiality principle, any personal data such as patient data would require appropriate security measures in place to protect it (Information Comissioner's Office, 2022a). Consent in GDPR is defined as "any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her" (Information Comissioner's Office, 2022b). Whilst not planned for this project; should a clinical trial of this application be conducted, a clear consent form and appropriate consideration into how data is gathered and stored would be needed.

1.3.4 MDR

As per UK MDR 2002, it's possible for software applications to be considered a medical device and thus possibly subject to regulation and required to be UKCA marked. GOV.UK (2021) provides some practical examples of software that could be considered a medical device:

- An application that calculates medicine doses for the user to take/inject
- An application that tells the user they have a medical condition or disease or gives the user an individual percentage risk score of having one

This application's purpose is data gathering for clinicians' use - it does not report a possible condition to the user nor does it administer medicine.

1.4 Deliverables

This has two required outcomes for it to be considered a success. The first outcome is obtaining the knowledge of the needs of NHS Posture and Mobility Centre clinicians that could be used to develop applications like this project's other deliverable to enhance their ability to work. The outcome is the production of an application that can be used to supplement the clinicians' ability to work by using cutting-edge technology such as augmented reality to improve how clinicians record data and provide a platform to standardise that data too.

1.5 Thesis Outline

This thesis consists of six chapters. The first chapter is this introduction that outlines the intention of the project, why it will be useful, and what deliverables will be achieved. The second chapter provides background knowledge on spinal anatomy including what the spine is, an explanation of the five sections the spine is comprised of, and details surrounding the vertebrae, facets, and ligaments. This is followed by an exploration of possible conditions that can affect human posture and an explanation of the standard by which anatomical areas of interest are marked. Various mobile applications that perform some sort of similar function to that intended from the deliverables will also be explored and compared to find what they do well and what gaps in functionality exist. And finally, information obtained from an interview and survey from NHS will be included that contain valuable information about what PMC clinicians want to see from the ideal posture assessment application tailored to their needs.

The third chapter presents the design of the application. This section contains a blueprint of the development cycle and the structures that make up the application. This includes details on the tools and methodology used, how the application will conceptually operate, the core class diagram and code structures that will be used, and the design of user interface elements. The fourth chapter details the implementation of the application, putting the third chapter's content into practice. Included is essentially an outline of what the end application is, including a description of all the code that was developed and how it all links together.

The fifth chapter describes the evaluation of the application's effectiveness. This includes a reflection on the design choices and code implementation, how well the code features of the application work when tested, and the performance and usability of the application itself. The sixth chapter concludes the thesis, describing how well the end product meets the objectives originally set out and what future work could be done to further the project if given the chance.

2 Literature Review

2.1 Human Posture

2.1.1 Spinal anatomy

The spine is an important part of the human body designed to keep yourself upright, standing up and mobile. It protects a series of nerves called the spinal cord that connects the brain with the rest of the body. (University of Maryland, 2003). According to TeachMeAnatomy (2020), the spine or vertebral column is comprised of a series of approximately 33 bones called vertebrae. The vertebral column has five distinct sections called the cervical, thoracic, lumbar, sacrum and coccyx. Typically, a human vertebral column has 7 cervical, 12 thoracic, 5 lumbar, 5 fused sacrum and 4 fused coccyx vertebrae each as shown in Figure 1 (Kayalioglu, 2009). Each vertebra is labelled with a two-character name; a letter indicating its section and a number indicating its position ascending from the one closest to the cranium. For example, the 7 cervical vertebrae are labelled C1 through C7 (TeachMeAnatomy, 2020).

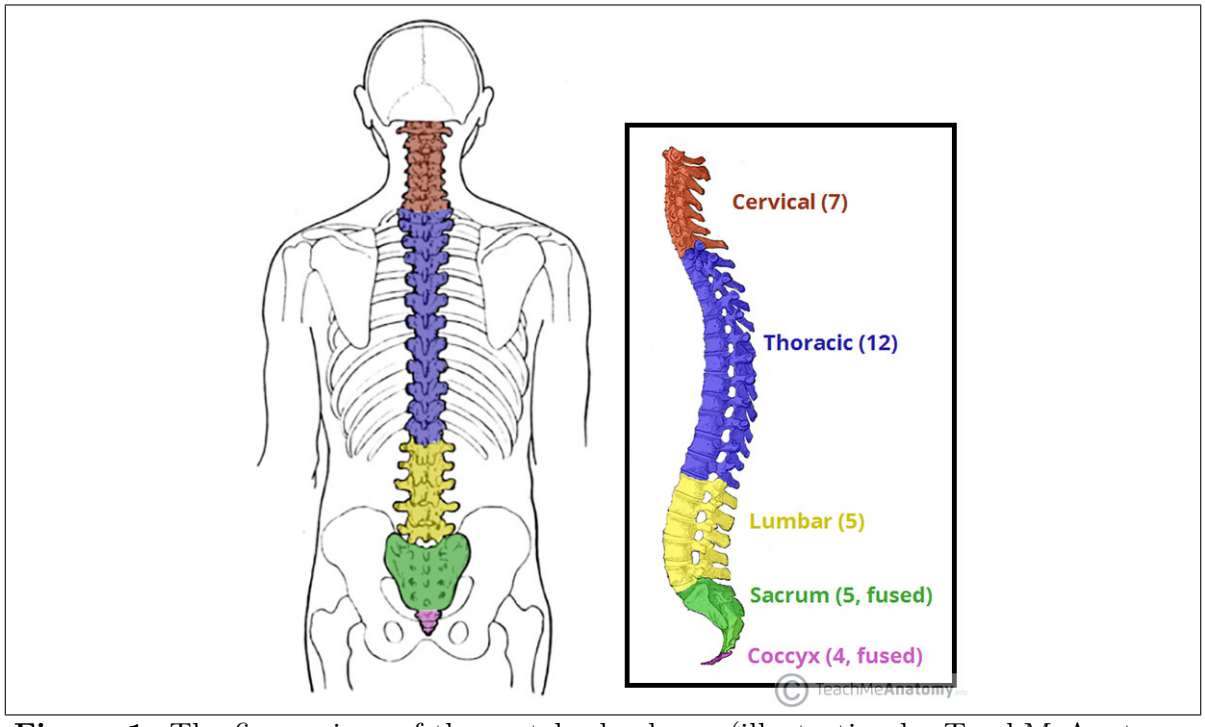

Figure 1: The five regions of the vertebral column (illustration by TeachMeAnatomy (2020))

The vertebral column is known to perform four main functions for the body; protecting the spinal cord, supporting the weight of the body above the pelvis, forming the

central axis of the body, and facilitating posture and movement of the body (TeachMeAnatomy, 2020). The cervical section (craniocervical) at the cranial end of the vertebral column allows for three-dimensional movement of the head and neck and is required for optimal spatial orientation of the special senses. The thoracic section (thoracolumbar) protects your heart and lungs whilst still being mobile to allow for breathing. Muscles in the region also provide core stability to the trunk and body as a whole. The lumbar (lumbosacral) and sacrum (sacroiliac) (together referred to as the inferior or caudal end of the vertebral column) transfer large forces from body weight and activated muscle through the pelvis and to lower extremities (Mansfield and Neumann, 2019). And finally, the coccyx (commonly known as the tailbone) is a vestigial tail that connects to the bottom of the sacrum via the sacrococcygeal joint (Veritas Health, 2021).

The vertebral column does not only consist of these vertebrae. To maintain its intended 'double-S' shape, skeletal support, and nerve routing, the vertebral column relies on several support structures in between and around the vertebrae. Firstly, most of the cervical (excluding C1 and C2), thoracic and lumbar vertebrae are separated by intervertebral discs, which are made of fibrocartilage and are anchored in place vertebral endplates (also made of cartilaginous material). The discs act as spacers and shock absorbers. Secondly, facet joints (also known as zygapophyseal or apophyseal joints) are paired on the left and right sides of the back of each vertebra from C3 to L5. These joints help stabilise the vertebral column whilst allowing for flexion, extension, and articulation movement. Facet joints are connected with connective tissue capable of producing nourishing fluid for self-lubrication. They are also coated in cartilage to ensure smooth movement (Highsmith, 2021). The relation of both structures to the vertebrae is shown in Figure 3.

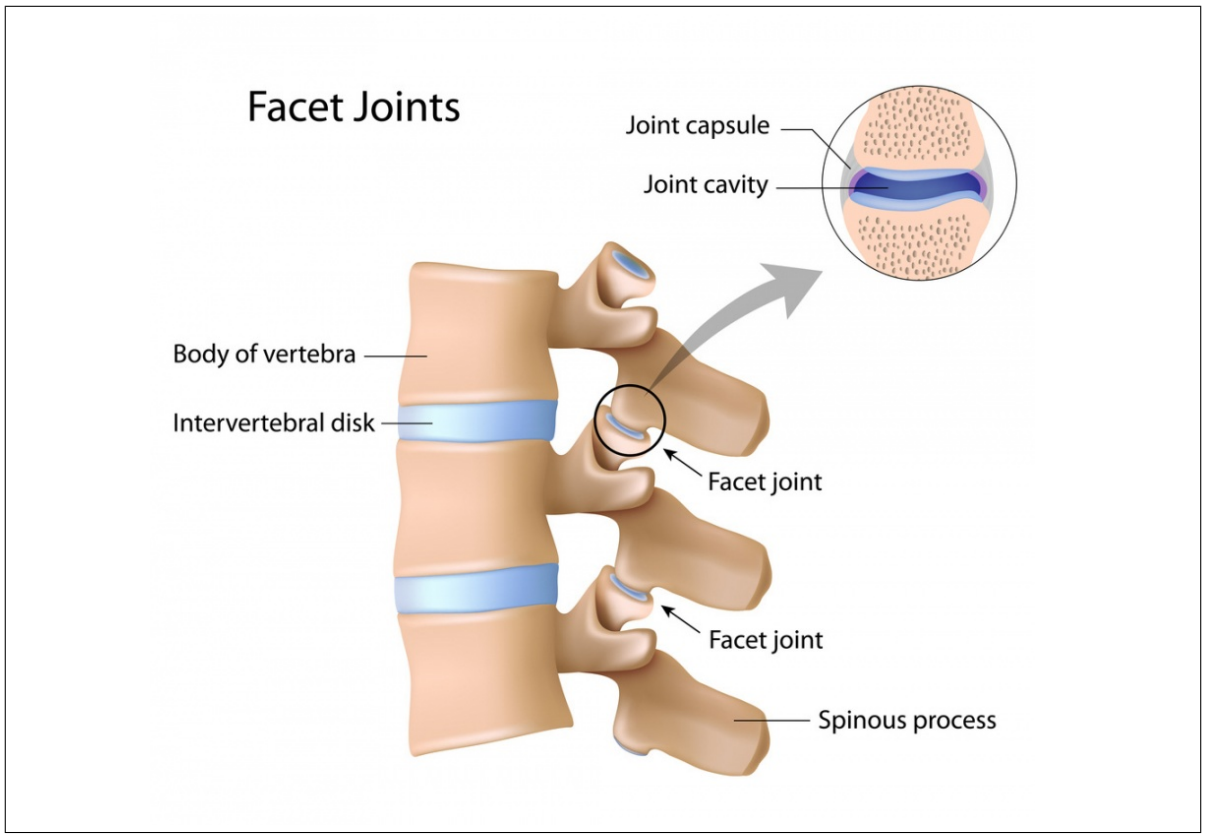

Figure 2: Composition of the facet joints (illustration by Highsmith (2021))

The vertebrae, intervertebral discs and facet joints are all connected via the connective tissue called ligaments. Ligaments in general connect bones together and help stabilise joints, and are vital in the vertebral column since they provide a brace to help protect the spine from injury (Colorado Comprehensive Spine Institute, 2016).

Name	Description
Anterior Longitudinal Ligament	Also known as ALL, this runs from the base of the skull to the sacrum and connects the anterior of the vertebral body to the front of the annulus fibrosis
Posterior Longitudinal Ligament	Also known as PLL, this also runs from the base of the skull to the sacrum and connects the posterior of the vertebral body to the back of the annulus fibrosis
Supraspinous Ligament	Connects the tip of each spinous process to the other
Interspinous Ligament	Connects to the ligamentum flavum
Ligamentum Flavum	The strongest ligament that runs from the base of the skull to the pelvis to protect the spinal cord and nerves

Figure 3: Table of ligaments in the vertebral column (Colorado Comprehensive Spine Institute, 2016)

2.1.2 Spinal deformities

Spinal deformities are abnormalities in the vertebral column's alignment and curvature due to factors such as birth defects, child growth, ageing, injuries, and surgeries. The three main spinal deformities are scoliosis, kyphosis and lordosis (Mayfield Brain Spine, 2018).

2.1.2.1 Scoliosis

Scoliosis is a lateral curvature of the vertebral column that typically occurs during a child's growth spurt near puberty (Mayo Clinic, 2021b), however, there are several types of structural scoliosis, including idiopathic, degenerative, neuromuscular, and congenital scoliosis. Idiopathic scoliosis accounts for about 8 out of 10 cases of scoliosis and typically forms during adolescence. The cause of idiopathic scoliosis is presently unknown. Degenerative scoliosis occurs as the joints in the vertebral column degenerate with age. Neuromuscular scoliosis can sometimes occur with neuromuscular conditions such as dystrophy or cerebral palsy. And finally, congenital scoliosis develops whilst a person is in the uterus, is present in infancy and needs to be corrected surgically (Veritas Health, 2017). Common symptoms associated with scoliosis, in general, include back pain and trouble breathing due to reduced chest area for lungs to expand (Healthline, 2021).

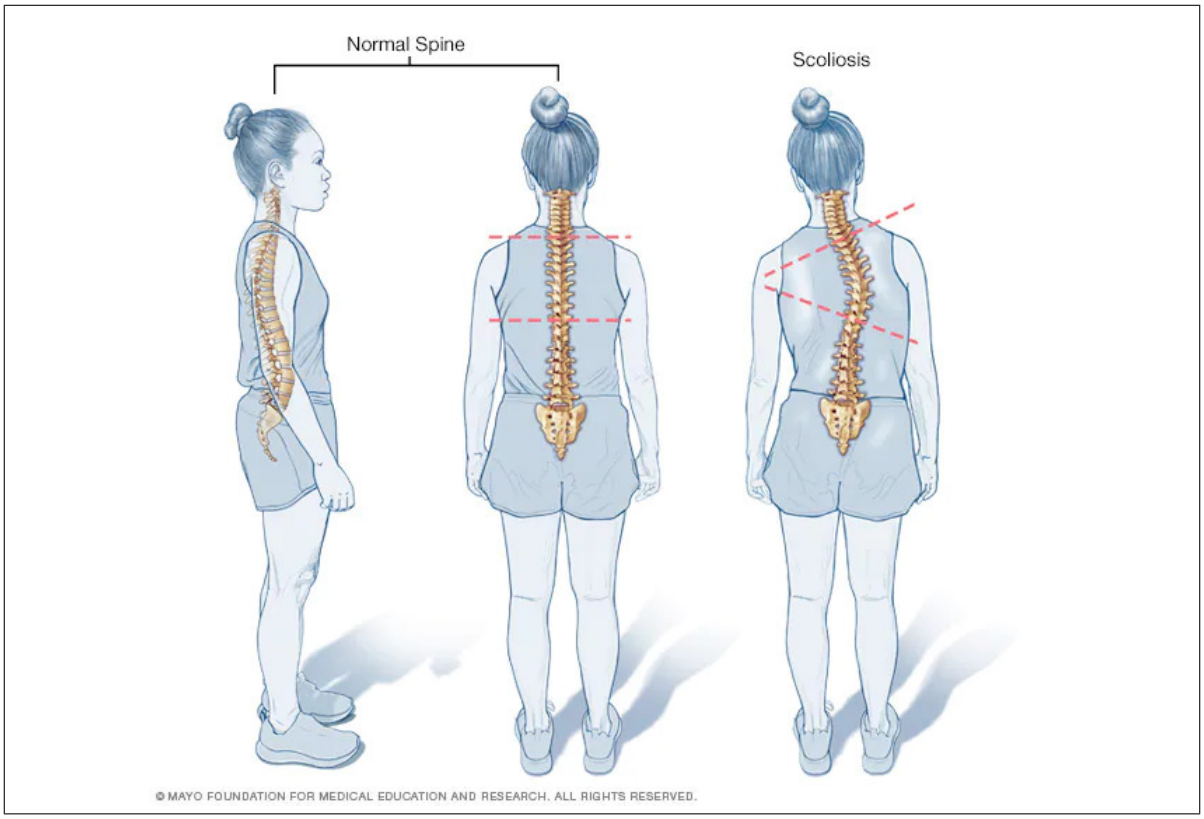

Figure 4: Comparison of a normal spine and one affected by scoliosis (illustration by Mayo Clinic (2021b))

Cruickshank, Koike and Dickson (1989) notes that scoliosis can also occur in localised areas of the vertebral column. For idiopathic scoliosis, six variations of curve patterns can be described:

- **Single thoracic**: when the apex of the curve lies around T2 to T11.
- **Single thoracolumbar**: when the apex of the curve lies at T12 or L1 and extends to T8 to T11 and L2 or L3.
- **Single lumbar**: when the apex of the curve lies at L2.
- **Double thoracic/thoracic**: when two curves are present and the upper apex lies around T3 or T4 and the second apex lies from T7 to T11.
- **Double thoracic/lumbar or thoracic/thoracolumbar**: when a thoracolumbar or lumbar curve occurs with a contralateral thoracic one.
- **Triple thoracic/thoracic/lumbar or thoracic/thoracolumbar**: a small upper thoracic curve with larger curves in the lower thoracic and lumbar regions.

2.1.2.2 Kyphosis

Kyphosis is a forward curvature of the vertebral column that can occur at any age. The kyphosis common in that aforementioned demographic occurs due to weakness in the spinal bones that causes them to crack or compress (Mayo Clinic, 2021a). According to Cleveland Clinic (2020), there are three common types of kyphosis; postural, Scheuermann's and congenital. Postural kyphosis is caused by actions such as slouching that stretch the ligaments and muscles holding the vertebrae in place and is typically found to affect teenage women. Scheuermann's kyphosis is caused by vertebrae having a wedged appearance instead of rectangular and typically affects teenage boys. Congenital kyphosis is a born-with condition that can get worse as an affected child grows, typically due to the spine not developing properly before birth. Typical symptoms include the appearance of an affected person's back being abnormally curved, however, back pain, stiffness, tiredness, and tenderness of the spine can also occur (NHS, 2018).

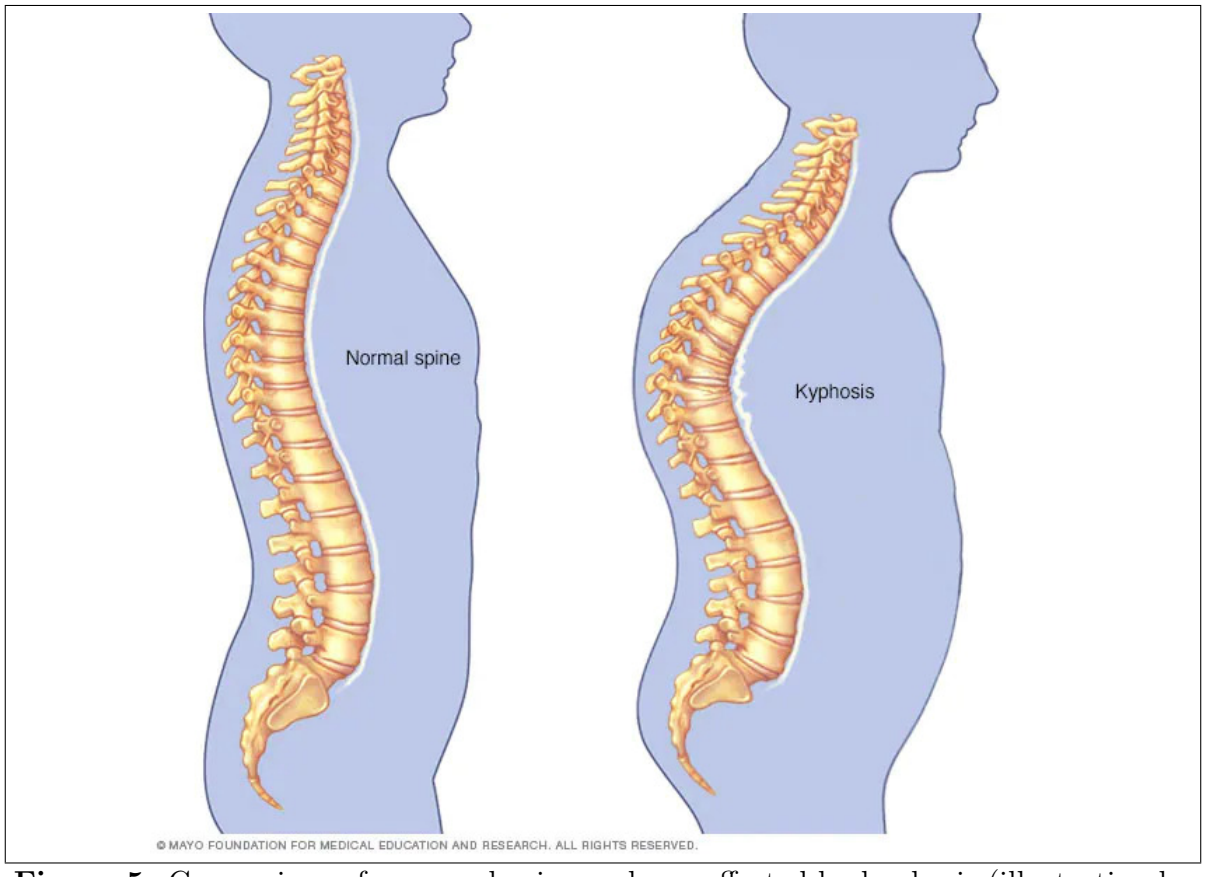

Figure 5: Comparison of a normal spine and one affected by kyphosis (illustration by Mayo Clinic (2021a))

2.1.2.3 Lordosis

Lordosis is an inward curvature of the vertebral column at the lumbar section. Whilst a small degree of such a curve is considered normal, having too much curving is called swayback. Typically found in childhood, it can often fix itself as the child grow and in such cases is referred to as benign juvenile lordosis (Mount Sinai, 2021). According to Boston Children's Hospital (2021), the main symptom is the 'swayback' appearance, which can manifest itself as protruded buttocks from the back and protruded stomach from the front. In some rare cases, severe lordosis can result in pain and weakness in one or both legs, as well as potential loss of bladder control.

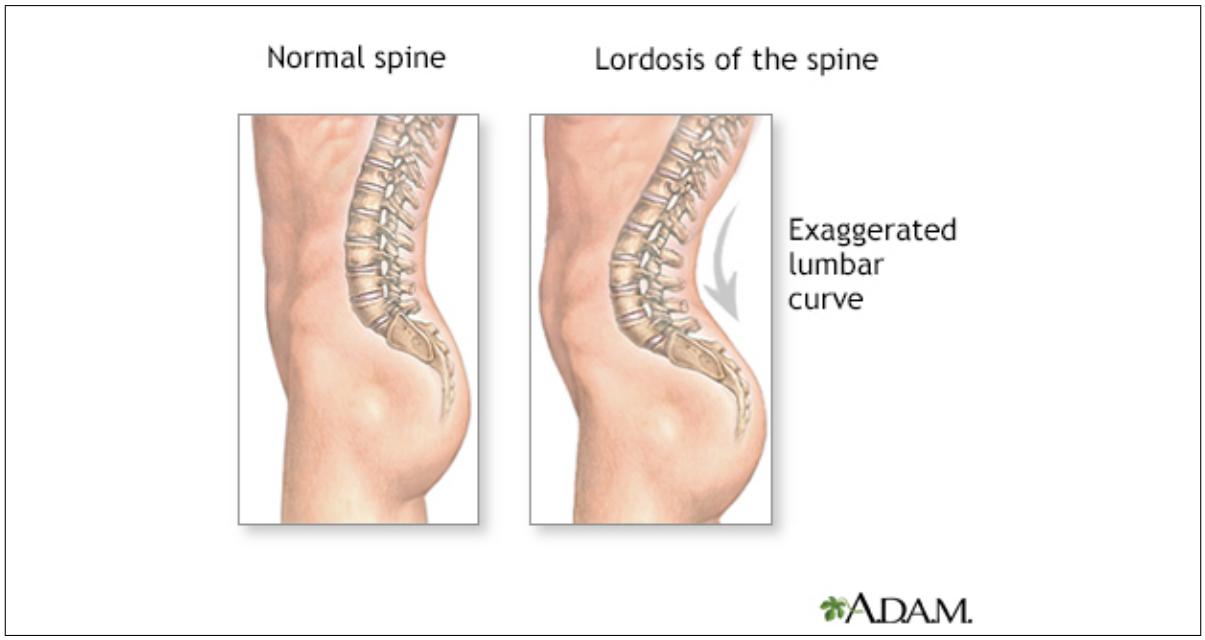

Figure 6: Comparison of a normal spine and one affected by lordosis (illustration by Mount Sinai (2021))

2.1.3 Pelvis anatomy

The [bony] pelvis is a basin-shaped bone structure that constitutes the framework of the pelvic region and houses organs inside it. It is typically divided into the pelvic girdle and pelvic spine regions. The pelvic spine is the posterior portion of the pelvis, composed of the two lower regions of the vertebral column covered; the sacrum and coccyx (Kenhub, 2021).

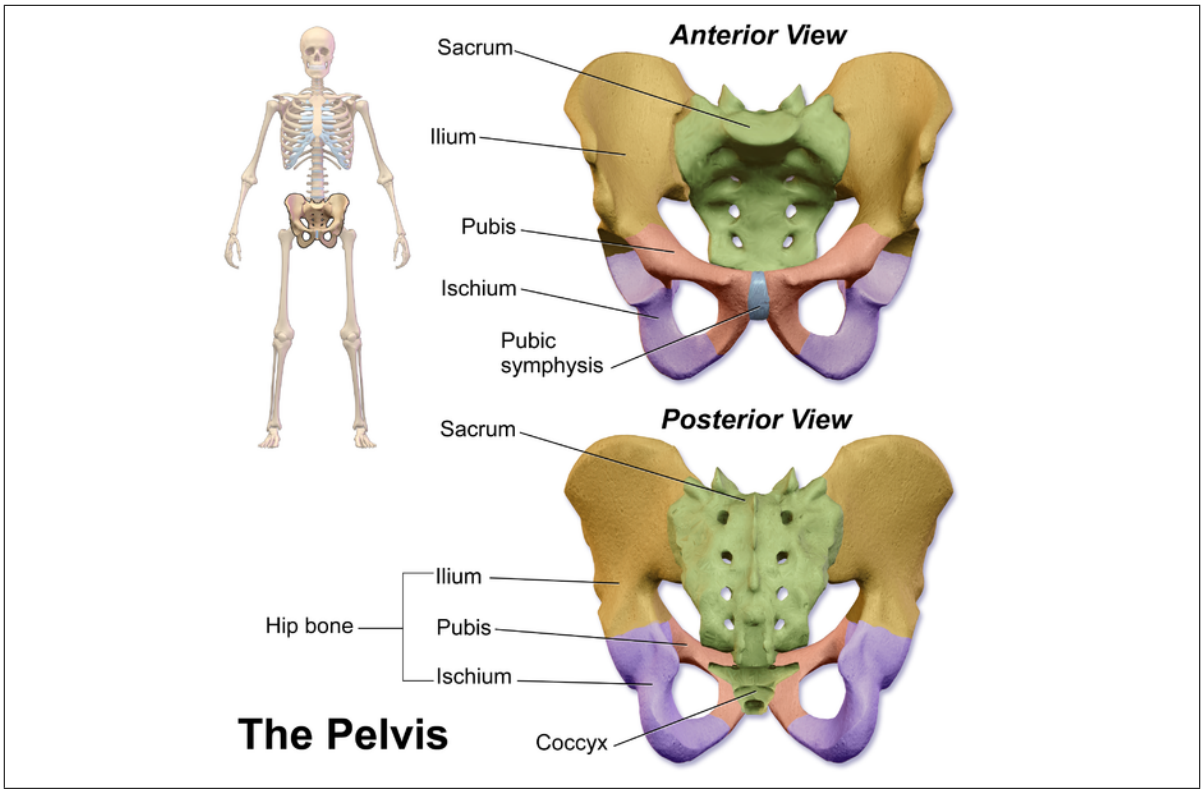

Figure 7: Anatomy of the pelvis (illustration by Physiopedia (2021))

The pelvic girdle is formed by the hip bone (also known as the coxal bone). It serves to attach each lower limb to the upper half of the body. Each hip bone is then connected to the pelvic spine. The hip bones are comprised of three separate bones called the ilium, ischium, and pubis as shown in Figure 7. The ilium is the superior region that forms the largest part of the hip bone and is firmly attached to the sacrum. The ischium is the posteroinferior region of the hip bone that supports the body whilst sitting down. And finally, the pubis is the anterior portion of the hip bone that joins the two hip bones together at the pubic symphysis joint (Lumen Learning, 2021).

2.1.4 Pelvic orientation

2.1.4.1 Pelvic rotation

Pelvic rotation occurs when one hip is further forward than the other due to one of the anterior superior iliac spines being more forward. This creates the appearance of one leg appearing more forward than the other whilst the person affected is seated. Pelvic rotation can be caused by unsuitable seat dimensions for a given person, spinal rotation, asymmetrical hip movement or stance, and trauma to the hip (CareFlex, 2021b).

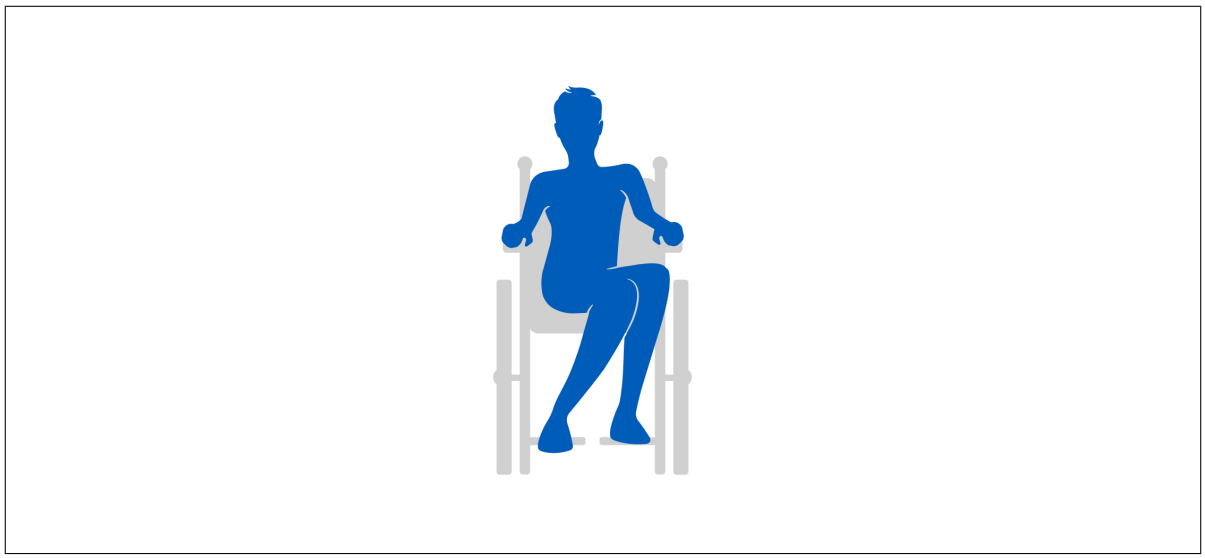

Figure 8: Demonstration of pelvic rotation (illustration by CareFlex (2021b))

2.1.4.2 Pelvic obliquity

Pelvic obliquity occurs when the pelvis is asymmetrical, tilting to one side. This can happen on either side of the pelvis, occurring when one anterior superior iliac spine is higher than the other. This creates the appearance of a lean or fall to one side as the pelvis affects the position of the vertebral column. This results in unequal weight distribution and the affected person's attempt to correct their posture cause a curve in the thoracic region of the vertebral column. This leads to pelvic obliquity being associated with scoliosis (CareFlex, 2021d). Pelvic obliquity is known to be also caused by another underlying medical condition, or due to reduced physical activity resulting in low or asymmetrical muscle tone paired with poor seating. This makes pelvic obliquity common in old people (Yorkshire Care Equipment, 2020).

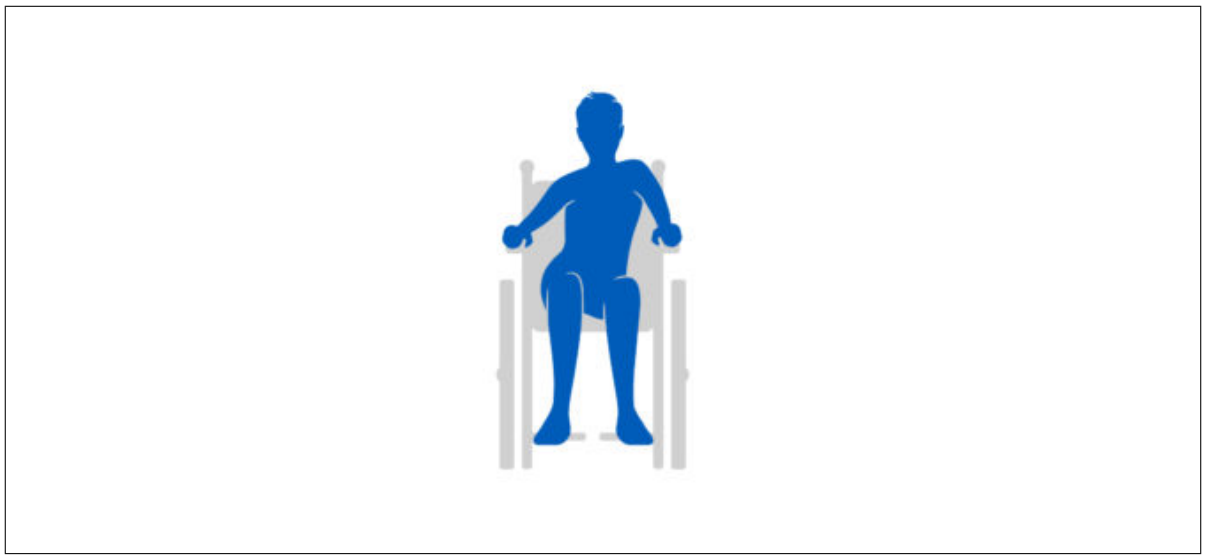

Figure 9: Demonstration of pelvic obliquity (illustration by CareFlex (2021d))

2.1.4.3 Pelvic tilt

Pelvic tilt occurs when the pelvis is tilted forward or backwards from the superior end of the pelvis. A forward tilt is known as an anterior tilt and a backward tilt is known as a posterior tilt. An anterior tilt is caused when the anterior superior iliac spines are lower than the posterior super iliac spines. It can be associated with lumbar lordosis and can be due to said lordosis, poor back support, an anterior slope on seating, tight hip flexor or paraspinal muscles, or obesity (CareFlex, 2021a). In turn, a posterior tilt is caused when the anterior superior iliac spines are higher than the posterior super iliac spines. It can be associated with thoracic kyphosis and can be due to unsuitable seat dimensions, fatigue, abnormal muscle tone and poor foot support (CareFlex, 2021c).

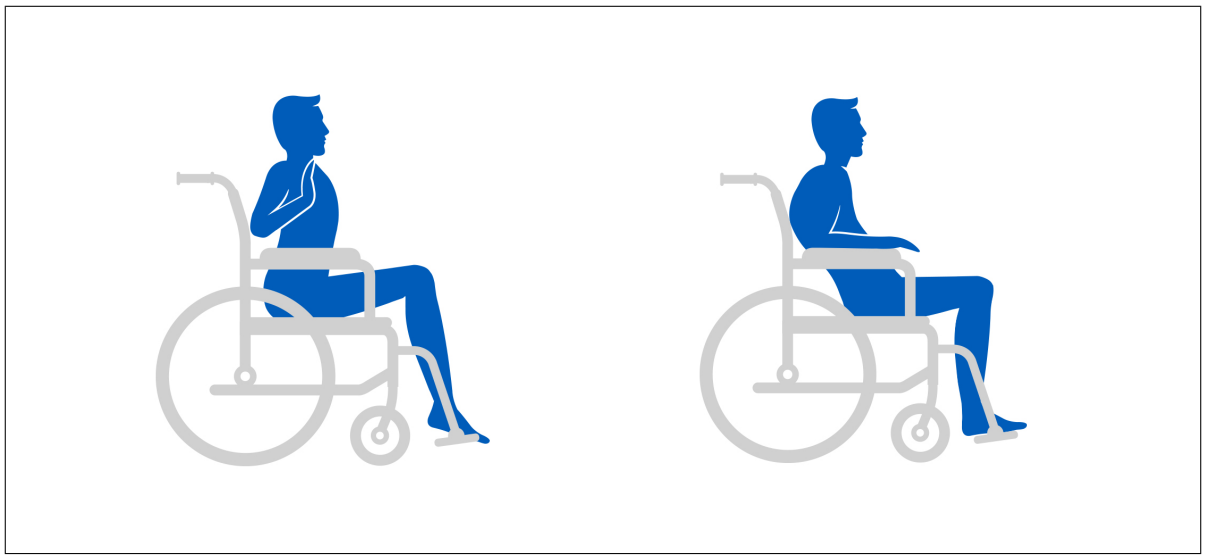

Figure 10: Demonstration of anterior (left) and posterior (right) pelvic tilt (illustration by CareFlex (2021a) and CareFlex (2021c) respectively)

2.2 Postural Assessment

Also called posture analysis or postural alignment assessment, postural assessment is the observation of the body's posture and is a part of kinesiology - the study of the body's anatomy and how the body moves. The body's static posture and alignment are typically measured, which are used to look for abnormalities and imbalances (OriGym, 2020) as described in previous sections. According to Seffinger and Hruby (2007), postural assessment is also used to observe the symmetry of paired anatomic landmarks for symmetry. A typical postural assessment would entail instructing the patient to stand still with feet shoulder-width apart, facing forward and arms relaxed to their sides, and then evaluating the patient's posture from anterior, posterior and lateral views.

2.2.1 Anatomical Landmarks

Anatomical landmarks are markers that identify the exact location of a measurement site, generally made close to the body's surface to mark an identifiable skeletal point (Norton and Olds, 1996).

Symbol(s)	Name	Description
V	Vertex	The most superior part of calvaria whilst standing up, thus highest landmark when standing up (Radiopaedia, 2021b).
A_L, A_R	Acromiale	The most lateral point on the inferior border of the acromion process (Oxford Reference, 2021a).
R_L, R_R	Radiale	The anterior, upper, lateral part of the radius' head (Oxford Reference, 2021b)
S_L, S_R	Stylion	The anterior, lower, lateral part of the radius.
$ASIS_L$, $ASIS_R$	Anterior Superior Iliac Spine	The most anterior part of the ilium and attaches the inguinal ligament and sartorius muscle (Radiopaedia, 2021a).
$PSIS_L$, $PSIS_R$	Posterior Superior Iliac Spine	The most posterior part of the ilium and attaches the sacroiliac ligament and multifidus muscle (GetBodySmart, 2021).
AP_L, AP_R	Anterior Patella	The anterior point on the knees.
ST_L, ST_R	Sphyrion Tibiale	The lowest point of the medial malleolus.

2.2.2 Measuring Posture

Singla and Veqar (2014) has described in detail the different ways posture is typically measured by practitioners, with the use of the technology ranging from non-existent to completely technology-dependent. These methods include visual observation, plumb line, goniometry, radiographic and photogrammetric. Visual observation is the simplest method of assessing posture, requiring no equipment or technology. Precise data cannot be gathered through this method, thus its use is discouraged for scientific research purposes. The plumb line method and combined with a postural grid is another simple method, with measurements of side and back plumb line alignment compared against guidelines. However, this method also cannot produce precise data.

Goniometry uses an instrument called a goniometer to measure the range of motion for a given joint. However, this method can be applied to posture assessment since a goniometer can be used to measure postural angles such as neck inclination and cranial rotation too. This produces quantitative data, unlike the previous two methods (Singla and Veqar, 2014). Radiographic assessment with the use of X-rays is considered a 'gold standard' in the evaluation of scoliosis (Negrini et al., 2009). Despite this, it can be costly and risks exposure to harmful radiation (Singla and Veqar, 2014). Photogrammetric assessment is the most widely used non-invasive measurement technique. It quantifies linear distances and angles on digital images through software (Singla, Veqar and Hussain, 2017). For seated individuals, novel approaches such as using an articulated arm coordinate measuring machine such as a MicroScribe can be used to

gather quantitive data (Hillman and Hollington, 2016).

2.2.3 Posture Management

Posture management is a multi-disciplinary approach to minimising posture-related abnormality and enhancing function and is considered on an individual basis. It should be considered when an individual is at risk of posture challenges discovered through assessment (CareFlex, 2022). A twenty-four hour approach to posture care is usually employed to help people adopt positions that are as comfortable as possible throughout the day and night, especially considering how the amount of time people are in a single position due to sleeping can cause a body to resist changing into alternative positions (NHS Education for Scotland, 2017).

The main benefits of posture management include improving functional ability, facilitating the development of motor functions, controlling abnormal movement patterns, encouraging activity, preventing secondary complications, improving quality of life, and enhancing breathing, swallowing and digestion (Moen, 2020). Identifying and managing bad posture is particularly important during child development as the stresses from gravity when an individual has bad posture can cause damage. Managing posture also allows the child to be as independent as possible and reduce energy expenditure. Typical ways to manage posture include adaptive chairs, gait trainers, standers and tricycles (Harvey, 2020). A specific example of assessment and management coming together to find and provide a solution that can improve comfortable is assessing internal musculoskeletal configuration to provide information necessary for constructing tailored devices to support a person such as custom seating systems (Partlow, Gibson and Kulon, 2021).

2.3 Existing Posture Assessment Software

Before proceeding to design an application to fulfil a task, it is important to investigate what similar applications already exist. The objective of this research section is to identify common features, pros and cons of existing posture assessment aiding applications to help inform feature development during a later stage of the project. To control the number of applications to explore, the applications chosen were at least two years old by the project start date (September 2019) and ideally appear within the top results of the Apple App Store and/or Google Play Store.

2.3.1 iGonio

iGonio is a goniometer application exclusively for iOS that is designed to accurately measure angles and distances on photographed radiographs. The application has a six-step user experience (Ahmed Zemirline, 2019):

I Photo acquisition: capturing a new photo or selecting an existing photo of a radiograph.

II Cropping: applying any needed cropping to ensure only the needed subject is in shot.

III Perspective adjustment: ensuring the radiograph is effectively placed.

IV Format selection: selecting the desired radiograph format resolution.

V Measure: dragging a ruler or protractor tool to desired positions to get obtain an angle or length measurement.

VI Export: saving or emailing the straightened radiograph and measurements.

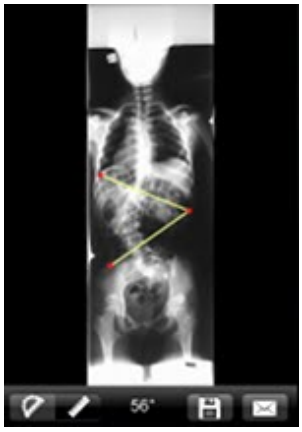

Figure 11: Screenshot of iGonio (Ahmed Zemirline, 2019)

The application's visual appearance and structure are simplistic. It appears that the application was developed and has not been updated since several years ago, thus paired with the fact the application is highly specialised for a specific task means the application does not need a highly sophisticated layout since it only needs to perform a single task. Besides the initial menu, most user elements do not possess a text label, instead of having just an icon to represent a function. The lack of other features allows for this since there is less scope for confusion. Due to the highly specialised nature, the application is suited for the aforementioned six-step process for operation.

2.3.2 PostureScreen Mobile

PostureScreen Mobile is a postural assessment mobile application developed by PostureCo for Apple iOS and Google Android devices. Its features include 1, 2 or 4-view

postural analysis, comparative reports, augmented reality assistance, seated desk analysis, remote screenings to clients via a separate RemoteScreen application, HIPAA-compliant cloud data backup, and report customisation (PostureCo, Inc., 2018).

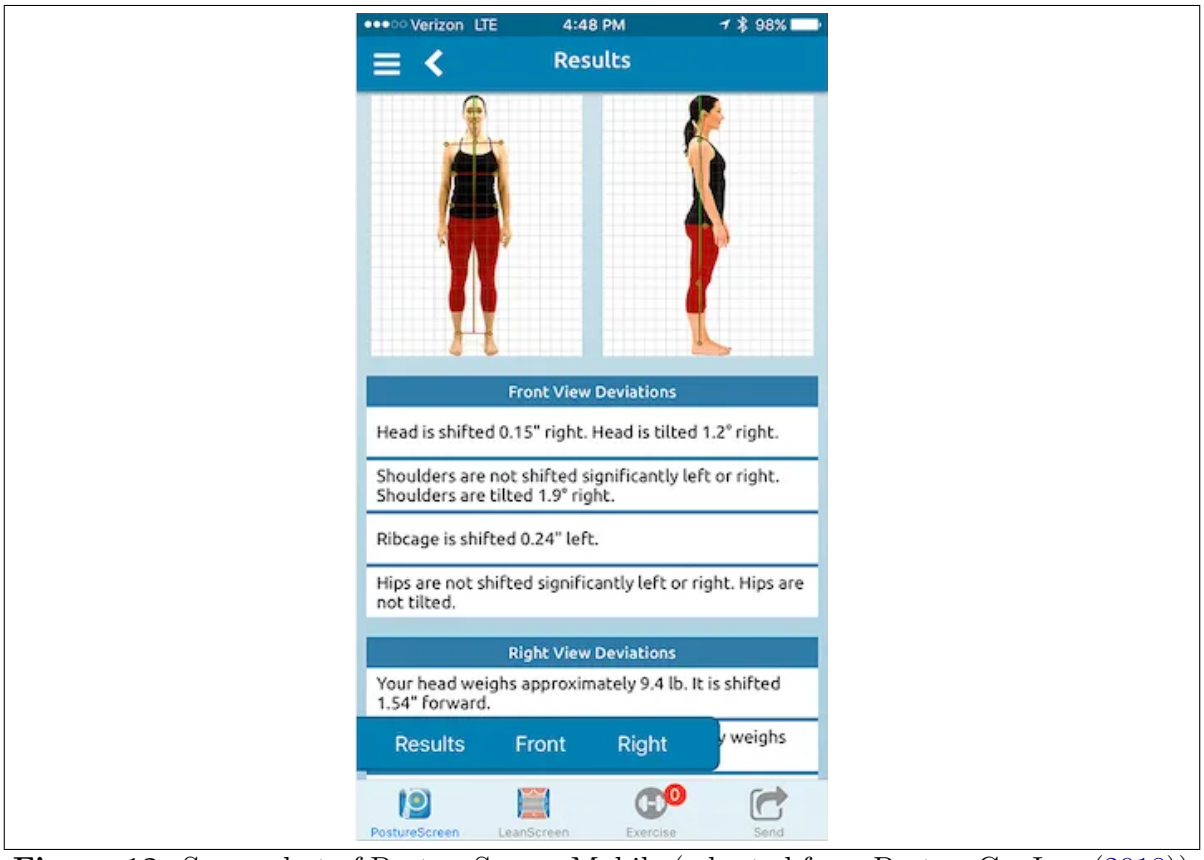

Figure 12: Screenshot of PostureScreen Mobile (adapted from PostureCo, Inc. (2018))

As shown by Ferrantelli (2017), using PostureScreen Mobile's base posture assessment function is largely a guided process. Data collection begins by entering the client's (ie, the patient) information such as name, sex, height, weight, and contact details. Next, photos of the client from the front, [right] side, back and left side are requested. The photos can be taken directly in the application or chosen from the host device's photo library. The back and left side photos are considered optional for the assessment. When taking each photo, several yellow guidelines need to be dragged to the edge of the client's body on the axis requested. Once the operator is happy with the photos, they tap 'Begin Posture Analysis' to start evaluating the client's posture. The application will start by requesting the operator to place markers for several anatomical landmarks on each of the photos previously taken. A photo preview of where the markers need to be is given in the bottom-left corner of the application. When the markers are placed, the application will produce a 'Results' page explaining all the deviations found. The results can then be sent to the client through email or cloud sharing, or a generated PDF version of the results report as shown in Figure 13 can be saved locally.

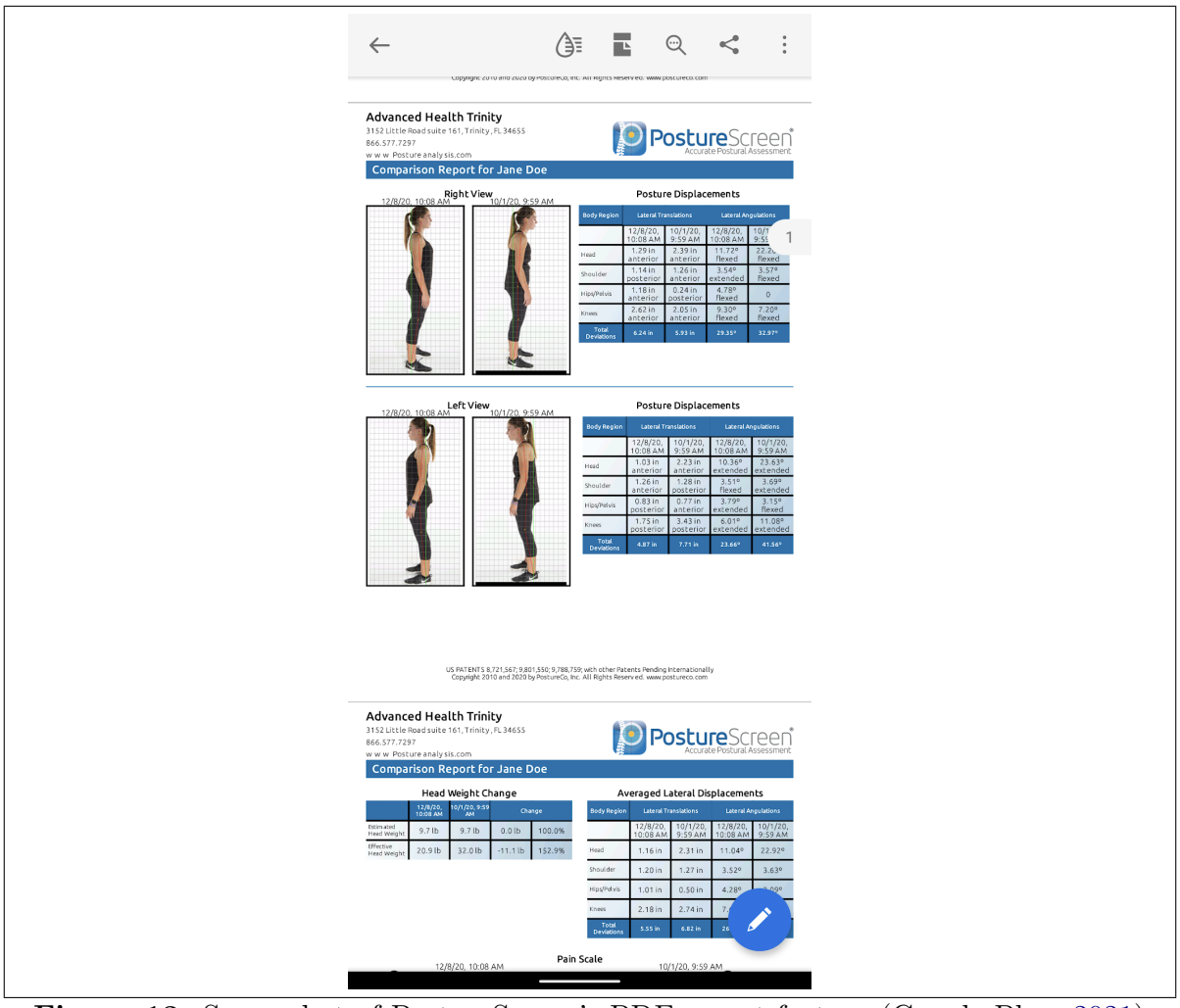

Figure 13: Screenshot of PostureScreen's PDF report feature (Google Play, 2021)

This application possesses several features besides basic anthropometric data entry. The most notable is the use of AR technology, something unique to this application out of all the applications discussed. According to a video by PostureCo, Inc. (2019b) demonstrating this feature, AR is employed to provide markerless tracking of a subject to speed up the assessment process. However, the video makes it clear that this is a limited implementation of the technology as manual calibration (shown as ruler guides being moved into place) is required. Another limiting factor for its usefulness is that this feature is strictly limited to the Apple platform and only devices using the Apple A12 or newer CPU and Apple iOS 14 or newer operating system (PostureCo, Inc., 2019a) - on older Apple devices or all Android devices, the process of using the application is manual in nature. Typed-in context (such as anthropometric data on the subject) is required and actions such as mapping out posture require using a generated snap-on grid. There are also several features that require an in-app purchase or external subscription, such as the LeanScreen (photographic analysis tools) and SquatScreen (objective functional

movement assessment) upgrades and the WebExercises.com integration.

The user interface and experience are significantly more developed than most of the other applications discussed. As shown in Figure 12, the application employs two styles of menus; a bottom bar and a hamburger-style menu. The former is used for selecting the main functionality, which includes PostureScreen (the main features of the application), LeanScreen, Exercise (for the aforementioned WebExercise.com integrated), and Send. The two middle buttons seem to be absent if the upgrade for LeanScreen is not purchased and WebExercise.com integration is not set up. The latter menu is used for secondary options.

2.3.3 PostureZone

PostureZone was originally a postural analysis program developed by BodyZone for Microsoft Windows, claiming to be an 'effective tool for assessing posture distortions, demonstrating postural improvements with clinical treatments, and wellness assessments'. PostureZone focuses on making the image importation easy, providing the ability to apply basic modifications such as zoom, cropping, brightness and contrast adjustments to an image being imported, and overlaying a digital posture grid to use as a reference when marking anatomical landmarks (BodyZone LLC, 2019). A mobile variant of PostureZone is available on Apple iOS and Google Android devices. It is available as a free-to-use and 'PRO' version. The free version allows anyone to analyse posture to see how it could be improved to help 'nature and tech-induced postural pain, sports performance, post-injury rehabilitation, and self-confidence'. The PRO version is designed for Certified Posture Exercise Professionals and health care providers to provide 'compelling visual evidence of improvement or decline'.

Figure 14: Screenshot of PostureZone Mobile (Google Play, 2019)

As shown by Weiniger (2014), PostureZone Mobile is also a guided process like PostureScreen. When the application is opened, a rough photo capturing guide is given that explains the person being analysed must be captured between two red guide bars. The operator then clicks the large blue camera button to begin taking photos. Each photo taken can only be confirmed once a rotatable guide bar for levelling in the middle of the screen turns green. Once confirmed, the operator then needs to move several 'posture zone' brackets into place around the subject, namely for the head, torso left, torso right, pelvis left, pelvis right and feet. The large blue camera button now turns into a check button that once tapped brings up the identified deviations as shown in Figure 14.

As aforementioned, the application has a free and paid-for 'Pro' version. The free application provides the functionality described above; measurement of changes in head, torso and pelvis alignment. This is coupled with social media features such as sharing results on Facebook and Twitter. The Pro version adds multiple case management, 'tap-to-zoom' placement of markets and additional export options (BodyZone LLC, 2021). The application is rated 3.1 out of 5 on the Apple App Store (2019) and 3.0 out of 5 on Google Play (2019).

2.3.4 Scoligauge

Scoligauge is an iPhone goniometer application developed by Ockendon.net that is designed to mimic the functionality of a scoliometer using the host device's accelerometer. To help the user, it gives a realistic-looking scoliometer model on screen and can also measure an object from a camera photo. The Franko (2011) review article states that its accuracy is dependent on the iPhone's internal hardware, but also states that the results that the application outputs were sufficient.

Figure 15: Screenshot of Scoligauge (Franko, 2011)

Like iGonio, this is a highly specialised application used solely for measuring trunk asymmetry and helping to identify scoliosis. Similar functionality could be identified as a requirement for the application, however. The fact this application exists and its accuracy was rated highly suggests an accelerometer could be used for implementing this functionality if needed.

2.4 User Interface

A user interface (UI) is a set of commands or menus in a program that the user communicates through. It is one of the main challenges of mobile application development as the quality of the UI design affects how easily a user can complete tasks with a given program (Fu et al., 2019). As listed by usability.gov (2021), there are four categories of UI elements:

- Navigational components: breadcrumbs, search boxes, sliders
- Input controls: for example, checkboxes, radio buttons, drop-down lists

- Informational components: tooltips, icons, program bars
- Containers: accordions

UX Planet (2018) explains that mobile menu navigation design is the most important between user and platform. Navigation is the 'skeleton of the app' as it supports the overall content that is to be displayed to the user, thus the development of an intuitive menu is a must. In the interest of ensuring this project's output would result in an easy-to-use experience for the clinicians that use it, the menu design will be heavily discussed in this research section.

2.4.1 Menu

As a practical definition, a menu is a graphical control that presents possible navigation options within its interface. Typically, it takes the form of a list of commands with verbs indicating possible actions like 'save' or 'buy', etc (UX Planet, 2017). Menus can be manifested in several forms, however, with the two most common types being the hamburger (or side drawer) and tab (or bottom) bar menu. The hamburger menu is very popular since it is designed to minimise the amount of screen space used in normal operation and the menu gets hidden in a single corner of the screen. By contrast, the tab/bottom bar menu is a fixed group of buttons usually residing at the bottom of the screen (Babich, 2020).

A paper by Tsiodoulos (2016) provides a comprehensive comparison of the hamburger and bottom bar menus on mobile devices. Tsiodoulos evaluated user performance and preference between the two types of menus with a survey, asking participants to complete three tasks on an application that had a hamburger and bottom bar menu variants. All tasks were timed and had the number of taps recorded. The control for the experience was using a Samsung Galaxy J5 smartphone with a 5-inch screen and 1280x720 resolution. The application was web-based and viewed on Android's Chrome web browser in full-screen mode.

The study showed that using the bottom bar menu was overall faster by 28.6% (rounded to 1 decimal place) in its sixth figure, performing better than the hamburger menu in two of the three tasks. The task where the hamburger menu was more performant was noted to be the more complicated task that involved going into multiple levels of navigation without aids such as the host device's back button or a breadcrumbs navigation bar. In the paper's seventh figure, Likert scale questions were used to assess perceived ease of use. The review states there is no significant difference in this regard, which could be explained by the fact all participants were able to complete all tasks with both versions of the test application.

Table 2 shows comparative questionnaire responses, asking participants to choose between the two versions of the application or state 'not sure/no preference' for a series of questions. Notable takeaways from the analysis include that 60% preferred the

bottom bar version and 70% felt that the bottom bar was faster/more efficient. The qualitative responses given seem to tell a more mixed story, with three responses noting that hamburger menus were what they were most used to but two of those noted that the bottom bar (that they were not used to the bottom bar) was not harder/outright easier to use. The former statement is backed up by Table 1, which shows the top 12 news websites and all but two of them use a recognisable form of the hamburger menu, according to Alexa Traffic Rank.

There are several blog-type posts from persons involved in web design who have voiced opposition to the use of hamburger menus. A TNW (2021) article written by Anthony Rose, co-founder and CTO of a social media company zeebox (now rebranded as Beamly) and former developer for BBC iPlayer, details the situation zeebox suffered when its Android application migrated from a top-aligned variant of a bar menu to a hamburger-based side drawer menu. After the change, user reviews on Google Play Store showed an improvement but zeebox's analytics showed that user engagement time was in fact halved. An update for the application was made within two weeks to restore the previous navigation pattern, whilst also providing an option for users to use the hamburger menu should they desire. His team then used A/B testing to reveal the true impact of using a side menu for their application, which showed that users served with the hamburger menu spent less time on the application than those served with the original menu. Another article that is critical of hamburger menus is Abreu, 2014's article. The former product designer who worked for KFC noted that hamburger menus have 'lower discoverability, are less efficient, clash with navigation patterns provided by the host operating system/platform, and the options within it are not 'glanceable".

2.5 2019-11 interview of PMC staff

During the early stages of the project, an interview was conducted with several members of NHS Wales Posture and Mobility Centre staff on the 12th of November 2019 to assess their views on the current process of posture assessment and whether a mobile application could be used to supplement their processes. A five-question script was written and is shown in Appendix A.1.

2.5.1 Description of data recording

PMC staff reported that during a posture assessment, most data is recorded on forms or freehand on notepaper. Any photos are taken on a phone or tablet on its native camera application. Once the assessment is complete, the data is fed into the BEST patient management system, which was specifically noted to be an old software solution.

2.5.2 Description of current issues

PMC staff were vocal regarding issues they perceived with their current process. Due to the use of free-hand note taking and simple photo capturing without specialised software, time can be wasted when interpreting recorded data and photos for input into the BEST system. It was especially noted that there were struggles with quickly understanding free-hand notes recorded by others.

It was also noted that there was a lack of standardisation with the data recorded process, within departments and between departments. As aforementioned, notes could be recorded free-hand and thus be in the style of a specific clinician. Photos are stored without meaningful metadata or naming convention to help interpret photos easily. It was mentioned that there was no equivalent to a data overlay on photos for description as usually found with X-Ray images.

2.5.3 Description of previously attempted solutions

PMC staff have attempted two ways to enhance the posture assessment and rectify the data standardisation issue. Firstly, one of the teams received tablets a few months before the interview to help supplement their workflow with technology. However, the staff noted they were unsure how to use them effectively and they were ultimately used for free-hand photo taking.

Secondly, a team attempted a protocol of naming and storing data a certain way. It was tested and implemented in the past, but it never achieved consistency between other departments.

2.5.4 Their questions regarding a mobile application

There was a positive reception to the idea of using a mobile application to improve their workflow experience. At this point in the interview, screenshots of the PostureScreen tablet application and the PostureZone smartphone application were shown to staff who unanimously agreed that the user experience and user interface seemed or looks too complicated respectively. They would like to avoid using a physical grid that PostureZone utilised since it would be difficult to take and set them up when staff are operated externally.

In return, the staff proposed three hypothetical questions to be considered when developing an application that would be suitable for their needs.

I What platform will this mobile application support?

II Is it simple to use?

III How would this handle someone who could only sit down?

2.5.5 The features they would like to see

PMC staff outlined four of the most useful features they would like to see from an application tailored to their needs.

I 'Touch-to-mark' anatomical marking system that would allow them to make free-hand notes on a photo subject but the application would handle standardising and interpreting the data.

II Length and angle measurements between two or more markers.

III Birds-eye photo-taking support - for example, taking a photo of someone lying on a plinth.

IV Automatic perspective adjustment; ensuring the photo environment is not warped from any lens effects.

2.6 2021-01 survey of PMC staff opinions

Beginning on 4th January 2021, an online survey designed to gauge staff opinions on the current progress of the application was run for two weeks. The survey was a seven-page, 25-question Microsoft Forms questionnaire paired with a video introduction to the application and a wireframe PDF tutorial explaining each stage of using the application. The description and outline of the questions can be found in Appendix A.2.

2.6.1 Response for questions on Page 1: Start

This section describes the first stage in the usage of the application - collecting relevant patient data. When respondents were asked if they believed this step was easy and intuitive to use, 80% agreed. A suggestion given was the addition of a patient record verification to ensure data integrity and validity. 60% of respondents felt that relevant patient info is represented. The qualitative responses requested data points for height, weight and the type of posture being assessed. It was also suggested that the buttons on the menu bar could change text based on the situation; an example given was 'Start' could change to 'Store' (or similar term) after patient data is entered, to help the user confirm the entry of the information.

2.6.2 Response for questions on Page 2: Take & Annotation Part 1 (touch-to-mark)

This section describes capturing a still frame image and annotating notes onto the screen. 100% of respondents believed this step is easy and intuitive to use and that the

touch-to-annotate feature is useful as-is. The addition of standardised measurements (with 'hip flexion contracture' given as an example) could be included. The need for a measurement function was also reiterated as its seen as a key part of the assessment process.

2.6.3 Response for questions on Page 3: Annotation Part 2 (templates)

This section expands from the last one, emphasising the inclusion of several predefined annotation templates that would enable quicker note-taking. 75% of the respondents feel this would be a useful feature, with one comment stating that they do not see a need for this feature as they can do the same with making regular annotations. Another comment highlights that the usefulness will depend on what example templates are available. When asked what templates should be included, several suggestions were given.

- Pelvis - anterior, neutral or posterior tilt
- Spine - Scoliosis concave left/right, Drop down options for level i.e. lumbar, thoracic, cervical
- Hips - abduction/adduction, flexion/extension
- Knees - flexion/extension
- Ankles - plantarflexion/dorsiflexion
- For all joints - 'limited', 'full range', 'spasticity'

2.6.4 Response for questions on Page 5: Send

This section explains the photo selection and upload-to-server feature. 100% of respondents believe this step is easy and intuitive to use. An improvement suggested is a way of tracking where sent items have gone. For example, being able to recover data in the event a wrong patient number is entered. 80% of the respondents felt the image selection checklist feature is appropriate as-is. It was suggested a message stating that the messages have or have not been sent would be helpful and that the checklist items should state what standardised measurement/template was used if applicable.

2.6.5 Response for questions on Page 5: User Interface

This section focuses on user interface elements, an area deemed important to the project as user accessibility was identified as a target during the earlier interview. 100% of respondents feel that the user interface elements are easy to understand and navigate,

that the colours and font used were legible and easy on the eyes, and were agreeable towards the way templated and custom annotations are separated. Despite this, it was suggested that button titles could be more intuitive and verbose. It was also suggested that the patient's name, initials or number could be displayed at the top-right of the screen to help make what patient is being evaluated more clearly. And, that a more clear colour contrast between the annotations and the image body would be beneficial for clarity.

2.6.6 Response for questions on Page 6: Summary

This final section was a Likert scale asking respondents to rate how they feel about various sections of the application to inform priority for which features are developed. The given rating options were as follows:

- I don't like this feature or believe it to be unnecessary
- Currently problematic but should still be needed with changes
- Currently fine but needs some improvements
- Current good as is and needs no improvements

Most responses were of the latter two, resulting in positive responses ranging from 60% to 100%. The only notable exception was the ratings for templates, which were rated good as-is and fine with some with 40% share each, while the remaining 20% rating it disliked or unnecessary.

2.6.7 Response for questions on Page 7: All done

A final page was given as an opportunity for any other thoughts or remarks not covered elsewhere in the survey. Two questions were given on details not explained or covered elsewhere in the survey; what happens if no internet connection is present and is there a place for clinicians to enter their details.

3 Design

3.1 Development Plan

3.1.1 Software

3.1.1.1 Unity3D engine

According to Lavieri (2015), the Unity engine is a popular C#-based game engine that is designed to be a sweet spot between capability and difficulty to learn. Unity comes out the of box with data structures, implemented physics, and user interface design capabilities. Unity is also compatible with several augmented reality technologies such as AR Foundation, Unity XR, and ARCore (Unity Technologies, 2021a). Whilst engines such as the Unreal Engine could be used for this project, Unity is considered to be the easier engine to work in and has a larger support community should it be needed. Unreal's strength is in graphics and rendering fidelity (Eldad, 2021), which would have no practical benefit for this project.

Unity 2019.3.5f1 is the version of Unity that this institution's computer game development laboratories have installed at the time of the project's start, thus, it will be the target version for this project. It is also a very recent release of Unity that will support all the required packages and latest technologies to be used.

3.1.1.2 Integrated development environment

Microsoft Visual Studio 2019 was chosen as the IDE for this project due to its widespread availability among this institution's computer laboratories and libraries. It is also bundled with Unity installations by default, making it a choice complement for Unity development.

3.1.2 Phases

This project will be developed in phases that group relevant tasks together, incrementally building up to the final output of a mobile application capable of camera capture, annotation and sending output, and real-time measurement. Each of those goals will be a phase for this project, along with a fourth one that will wrap up the project.

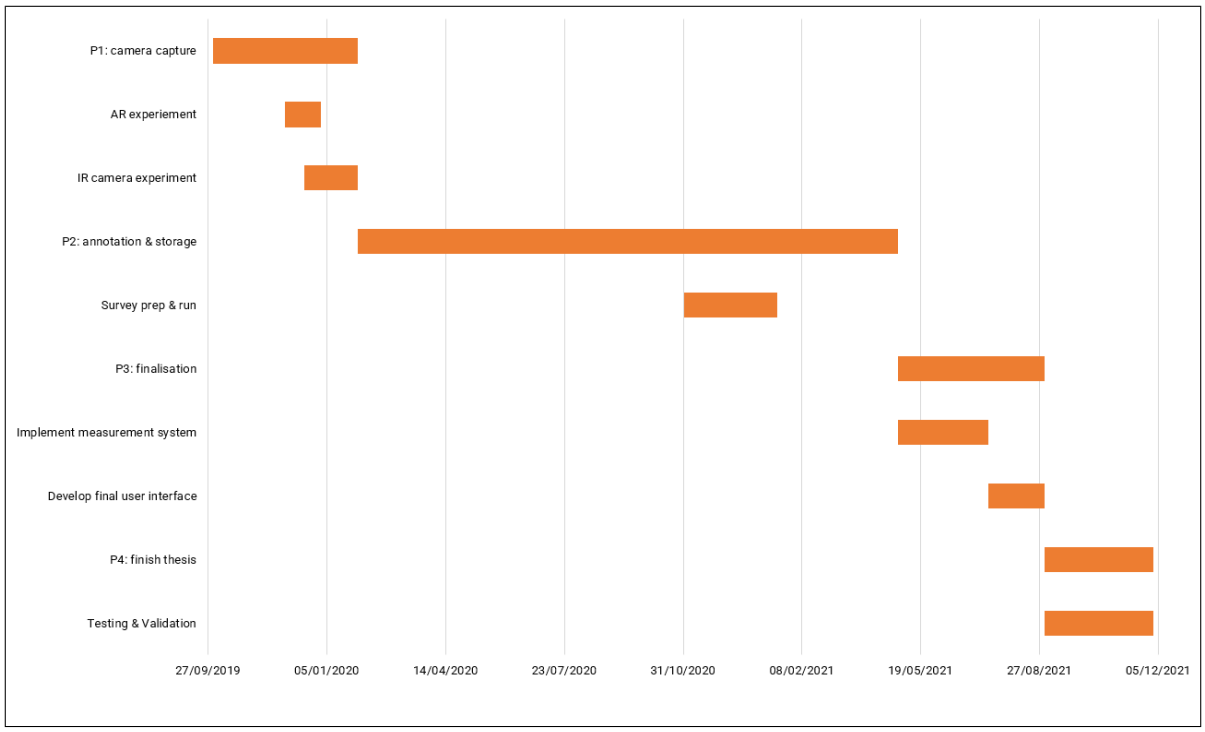

Figure 16: The final completed plan for the project (Gantt chart)

3.1.2.1 Phase 1

Phase 1 is entitled 'camera capture' which aims to produce a suitable codebase for future phases. The functionality will be basic - the application will only capture photos at this point - but the time during this phase will also be used for background research and experiments with augmented reality technologies.

3.1.2.2 Phase 2

Phase 2 is entitled 'annotation and storage' and will expand on phase 1 by adding the ability for the user to make freehand style annotations onto the subject photo and serialise the data for transmission to a given server.

3.1.2.3 Phase 3

Phase 3 is entitled 'finalisation' and will add an augmented reality-powered real-time measurement system to the application to allow the user to make measurements between two points and annotate them accordingly.

3.1.2.4 Phase 4

Phase 4 is entitled 'complete thesis' and will conclude application development, allow for testing and validation of the application, and as titled see the completion of this thesis.

3.1.3 Methodology

The project will be developed around Agile, which is a collection of methodologies that are iterative in design, allowing changes in how a given system is developed to evolve throughout the development process (cPrime, 2021). From the Agile framework, principles of SCRUM and Kanban will be used to enhance the development process and its productivity.

SCRUM is designed to break down the functions of a given system and place them into a product backlog, which is subsequently developed in 'sprints' that usually last a month. Kanban is designed to help facilitate just-in-time production, with the useful tenants of the methodology including incremental and evolutionary change and visualising what needs to be done. The most fundamental visualisation is the Kanban board, which breaks down tasks into at least three states; To Do, In Progress, and Done (kanbanize, 2019).

In practice, this project will make some minor alterations to the established systems; sprints will be conducted weekly instead of monthly and 'Next sprint' will be added as a fourth state of the project's Kanban board so a succession of tasks can be established. The method of tracking progress will be GitHub's built-in Projects system, which implements a highly customisable Kanban board system. A new Kanban board will be made for each phase of the project.

3.2 System Structure

3.2.1 Three-stage process

To promote ease of use and understanding of the process of using the application, a three-stage process will be established and maintained. The processes are given the simple names of Start, Take and Send.

3.2.1.1 Start

The Start process is the data collection step that will usually be completed just once per session, so there will be no need to return to this step after every image capture.

The NHS ID and several data points for the patient will be required before proceeding to another stage.

3.2.1.2 Take

The Take process is the image capture and annotation step. This step has two variants.

- If ARCore is supported by the host device, they can immediately start making measurements and annotations on the shown environment immediately after completing the Start stage. Pressing the 'Take' button will then capture and save the image with completed measurements and annotations intact.
- If ARCore is not supported by the host device, pressing the 'Take' button will freeze the frame and allow you to make as many freehand or templated annotations as required. Without pressing the Take button, a continuously live camera feed will be shown to allow the user to adjust the view to their needs.

In either variant of this stage, the Take step will be repeatable which allows the user to take and annotate as many photos as they wish.

3.2.1.3 Send

The Send process involves selecting what recorded data to send to a remote server and completing that said transfer. After completing as many image captures as they wish, the user can press the 'Send' button to be presented with a list of previously captured items that they can then check to confirm which ones should be sent to the remote server. After they have finished checking, the user can press 'Send' again to start the data transfer.

3.2.2 Flowchart

The diagram shown in Figure 17 shows how this application could be used. It demonstrates how a prospective user could navigate the three-step process described in Section 3.2.1. The flowchart diagram also describes how there will be two ways of using the application; with ARCore support and without ARCore support. Whilst the overall three-step process is similar, some processes are slightly different to reflect the different levels of technology being used.

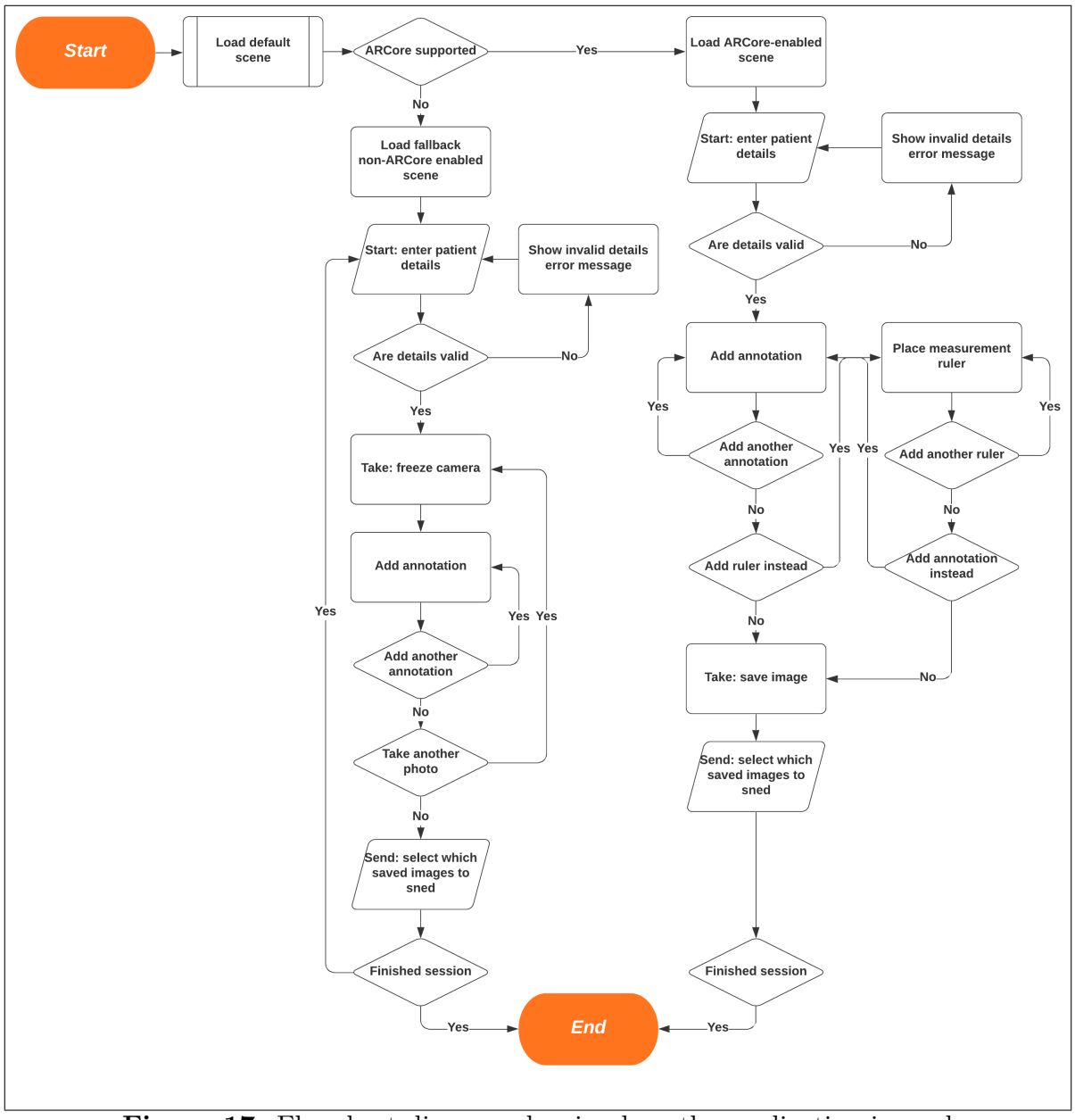

Figure 17: Flowchart diagram showing how the application is used

When the application starts, it will load a default scene that hosts the code needed to determine if ARCore is supported by the host device. The relevant scene is then loaded, and in both cases, the first step of recording patient details is required. This operation is the same for both scenes, where the user presses the 'Start' button to bring up the text input panel. Basic input validation will ensure the correct data types are being provided.

The two scenes diverge for the next operation - taking and annotating photos. For the ARCore-enabled scene, a continuous camera feed is shown where the user can

place annotations or measurement rulers on the subject. The user can be advised of the scanning motion needed to improve accuracy during this stage too. For the non-ARCore scene, the user will be required to freeze the camera by pressing the 'Take' button when they are happy with the subject's position to be able to annotate them. For both scenes, pressing the middle button after completing these tasks will save the result and the user can repeat the previous operations as many times as they desire.

The final operation is also the same between both scenes - selecting and sending results. Clicking the third 'Send' button will bring up a selection list UI element that allows you to preview and select what previously saved screens you want to send to the NHS' server.

3.3 Core Architectures

3.3.1 GameObject-Component architectural relationship

A GameObject is the most fundamental data type in Unity as one represents all objects displayed within the engine. Each GameObject is a collection of components that grant the parent GameObject functionality and properties (Unity Technologies, 2017). Components are based on the MonoBehaviour class, in which you can either use many of the predefined components or you can implement your own by inheriting that class (Unity Technologies, 2021c). Unity Technologies (2015) refers to the relationship between GameObject and components as a 'GameObject-Component Relationship'.

As confirmed by Baron (2019), the relationship is a component-based one. Panunzio and Vardanega (2014) explains that this means the general idea of the system is the separation of concerns into their own objects (in this case, Unity's MonoBehaviours) that contain only functional code inside and are highly reusable. These components are then composited onto a host (Unity's GameObjects). The approximate relationship between GameObject and component is shown in Figure 18.

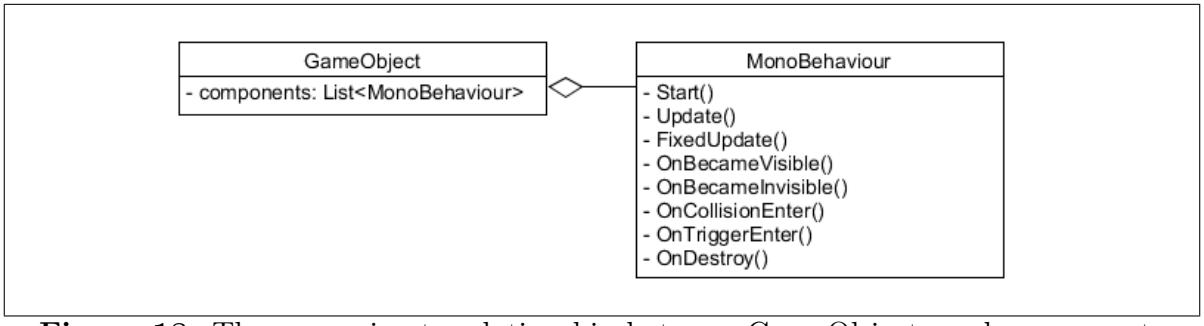

Figure 18: The approximate relationship between GameObjects and components

As shown in Figure 18, GameObjects can contain multiple initialised MonoBehaviour derivative objects. MonoBehaviour supports several built-in event messages that

are used in code as private methods with reserved names. As outlined in Unity Technologies (2021b)'s documentation, these are the most common methods and their respective purpose:

- Start: called when the GameObject begins to exist
- Update: called on every frame update
- FixedUpdate: called on a fixed framerate independent interval, useful for physics calculations
- OnBecameVisible: called when the GameObject leaves a camera's view
- OnBecameInvisible: called when the GameObject leaves a camera's view
- OnCollisionEnter: called when two GameObject's colliders begin overlapping
- OnTriggerEnter: called when two GameObjects physically collide
- OnDestroy: called when the host GameObject is destroyed

3.3.2 Command-based design pattern

The input system for the application will be implemented with a command pattern. At its core, the command pattern demands that a request for some action is represented in an object that encapsulates the necessary information to perform it at any given time. This allows greater control over how these actions can be handled (Akhtar, 2020). Figure 19 shows an example of how the command pattern will be implemented for a given command in the application.

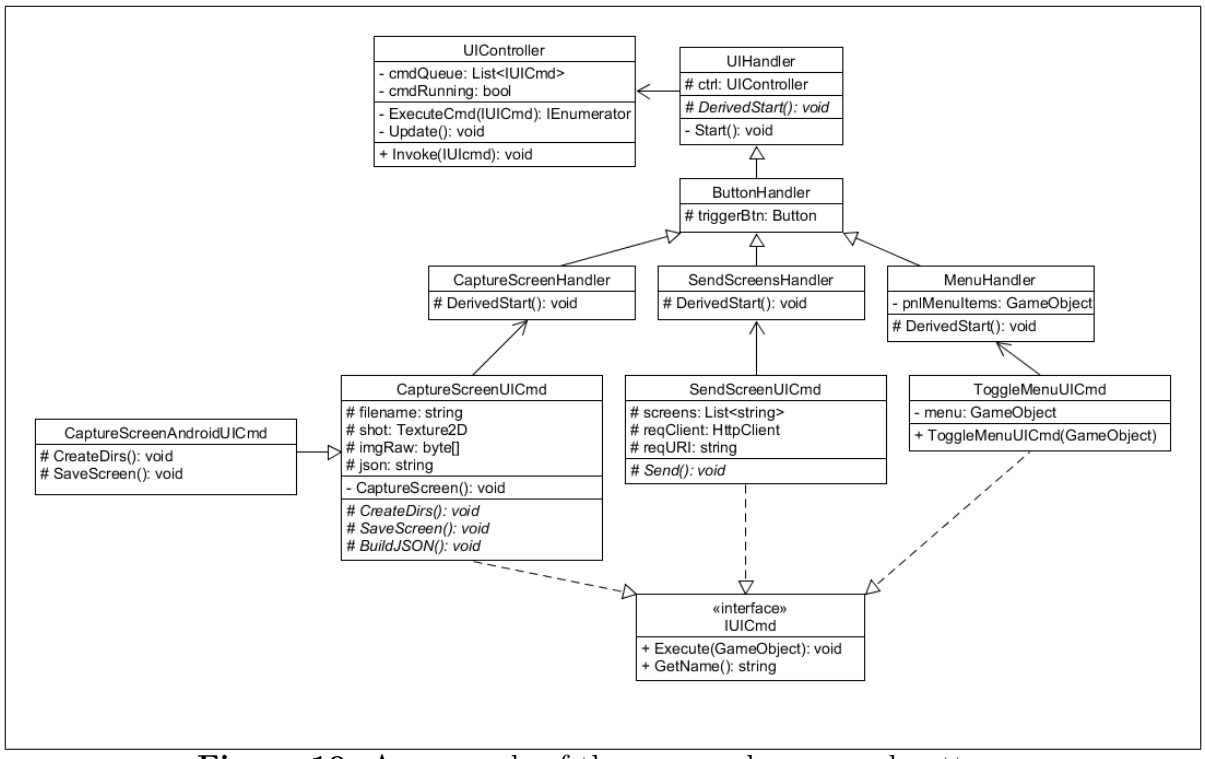

Figure 19: An example of the proposed command pattern

User interface commands are handled through three layers; a controller (UIController) that manages invoked commands' lifetimes, handlers (UIHandler derivatives) that create relationships like a button's callback that triggers commands, and the commands themselves (IUICmd implementations). The most important aspect of this system is both ensuring that a given currently-invoked command's payload is run concurrently to the rest of the application's functionality but also not at the same time as another command to prevent the possibility of race conditions becoming a problem. This will be solved by using the Unity Engine's coroutine functionality to allow the current command to be executed with the rest of the application; other invoked commands will be placed in a queue inside UIController.

It is anticipated that the concise nature of most commands will mean the use of the queue should not perceivable impact performance. If performance becomes an issue, a scheduler could also be implemented if needed.

3.4 Harnessing ARCore technology

3.4.1 ARCore environment understanding

Providing the real-time measurement component of the project relies on understanding and mastering the API Google exposes with ARCore and the ARCore SDK for Unity.

ARCore's understanding of its environments centres around three factors that are each represented with their own struct; CameraConfig, CameraIntrinsics and LightEstimate. All four will likely be used to inform the code to be written how ARCore is currently perceiving the environment.

CameraConfig displays how ARCore is using the device's camera, providing information such as the framerate, image size, Unity texture size, and how the device's possible depth and stereo sensors are used (Google, 2021b). CameraIntrinsics provides details on the camera itself such as focal length, image dimension, and the principal point (Google, 2020a. The LightEstimate struct provides an insight into the lighting conditions experienced when computing an augmented reality frame. This includes details on ambient light, colour correction, pixel intensity, and the reflection cube map (Google, 2021c).

3.4.2 Retrieving planes and feature points

In ARCore, a plane is essentially the system's best attempt at recreating surfaces of real objects that ARCore is looking at a Unity scene. ARCore constructs these planes using clusters of feature points that lie within a common orientation (Google, 2021a). Unfortunately, this process is not exposed in the SDK, meaning what constitutes a plane within the depths of ARCore itself cannot be controlled. Applications using ARCore are only given the end result of this buried computation.

When access to frames is needed, they can be retrieved from the ARCore session using the GetTrackables method and ensuring the first argument given is a list of type DetectedPlane to filter out other types of trackable objects that also exist. The DetectedPlane objects within this list contain information on the plane, including the type (horizontal or vertical), its centre position, the extent of the plane on the X and Z axis, and if the plane has been subsumed by another. When access to the raw feature points is needed, they can be retrieved via the PointCloud class, which is a container of PointCloudPoint structs that are generated for each feature point identified in a given augmented reality frame (Google, 2020b). The PointCloudPoint contains the feature point's identifier, position in Unity's world space, and the normalised confidence value for this point.

3.4.3 Building user input to touch planes

Interaction between Unity and the scene ARCore is handled through raycasting. In the context of 3D interactivity, raycasting is the process of detecting an object or a specific point of an object by shooting an invisible ray from a given or set point of origin (Glover, 2017). In this context and desired usage, user interaction with ARCore-managed objects (in this case, planes) will be achieved by using raycasts that when generated by the user touching their device's screen will translate the two-dimensional position on the screen

they touched to a three-dimensional ray to strike an ARCore-managed object with.

```
1  private void Update()
2  {
3      // Object to use for sampling
4      Touch touch;
5
6      // Attempt sampling of screen touch, returning if no touch was detected
7      if (Input.touchCount < 1 || (touch = Input.GetTouch(0)).phase !=
           ↪ TouchPhase.Began)
8              return;
9
10     // List for storing possible object hits
11     List<TrackableHit> hits = new List<TrackableHit>();
12
13     // Filter only planes as valid objects to hit
14     TrackableHitFlags raycastFilter = TrackableHitFlags.PlaneWithinPolygon;
15
16     // Attempt raycast at touch position and with plane/point filter
17     Frame.RaycastAll(touch.position.x, touch.position.y, raycastFilter, hits)
           ↪ ;
18 }
```

Figure 20: Code snippet for touching ARCore-managed objects within Unity via raycasting

Figure 20 demonstrates how this connection between touching the screen and raycasting to get ARCore-managed objects can be achieved. Of particular note is the TrackableHit type used. TrackableHit is a struct that contains the field Trackable for returning the actual struck object itself whilst also giving details about the raycast itself; such as the distance between the point of impact and the user and the exact position in Unity's world coordinates the raycast hit the object (Google, 2020c).

3.5 Use of HL7 FHIR

The Fast Healthcare Interoperability Resources (FHIR) is a framework created by HL7 to assist with transmitting clinical and administrative healthcare data for contexts such as mobile phones, cloud, electronic healthcare records data sharing, and server communication. It is designed to be fast and easy to implement with human-readable serialised data output and support for XML and JSON standards (HL7, 2019c). Due to its healthcare background and support of JSON serialisation that is supported by C# and the Unity engine, FHIR-based JSON data was chosen as the method of data transmission.

FHIR can be implemented in this project by creating a C# public class for each needed FHIR structure with the C# Serializable attribute attached and all needed properties represented by public variables of a suitable type. As explained by Microsoft Corporation (2021), the Serializable attribute indicates that a given class can be serialised by other mechanics within C# such as the BinaryFormatter or SoapFormatter object serialisers/deserialisers. For Unity, JsonUtility's ToJson method can be used to generate a JSON representation of an object and its public fields (Unity Technologies, 2021d). To allow arrays to be serialised, they will need to be declared as used as a C# list instead.

3.6 User interface

3.6.1 Bottom bar-style main menu

As concluded through research of the two principal types of mobile application menus, the main menu for the application will be a bottom bar menu as it is most suited to represent the core functionality of this application at all times. The menu will contain three buttons representing the three-stage process outlined in Section 3.2.1 and shown in Figure 21; Start, Take and Send. When the application is in the Take stage, the 'Take' button will change to an 'Accept' button for use with confirming the completion of image capture and annotation.

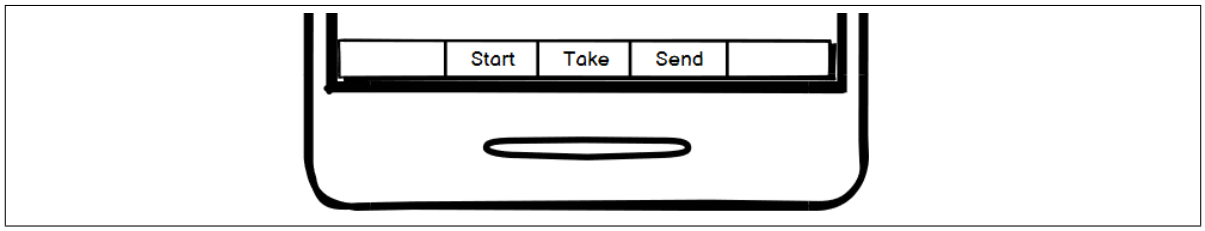

Figure 21: Wireframe mock-up of the bottom bar menu

3.6.1.1 Hamburger-style options menu

To get around the bottom bar's weakness of having limited density to put functions on within compromising usability, a side drawer menu will also be available for displaying secondary options. As an example shown in Figure 22, the sidebar could be used for displaying template options during the Take stage.

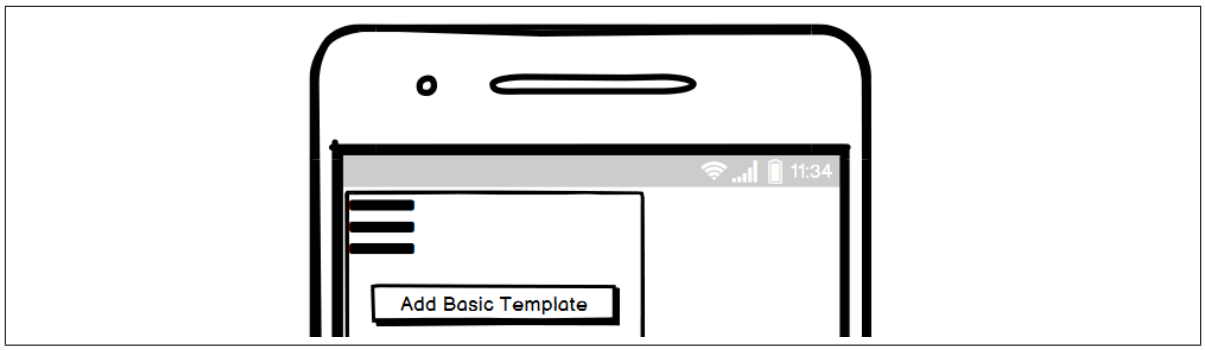

Figure 22: Wireframe mock-up of the sidebar menu

3.6.2 Initial data input panel

The Start stage's data input panel will be a scrollable box that appears at the bottom of the screen. As shown in Figure 23, it will be a list of text boxes with appropriate labels to define what data is expected in each field.

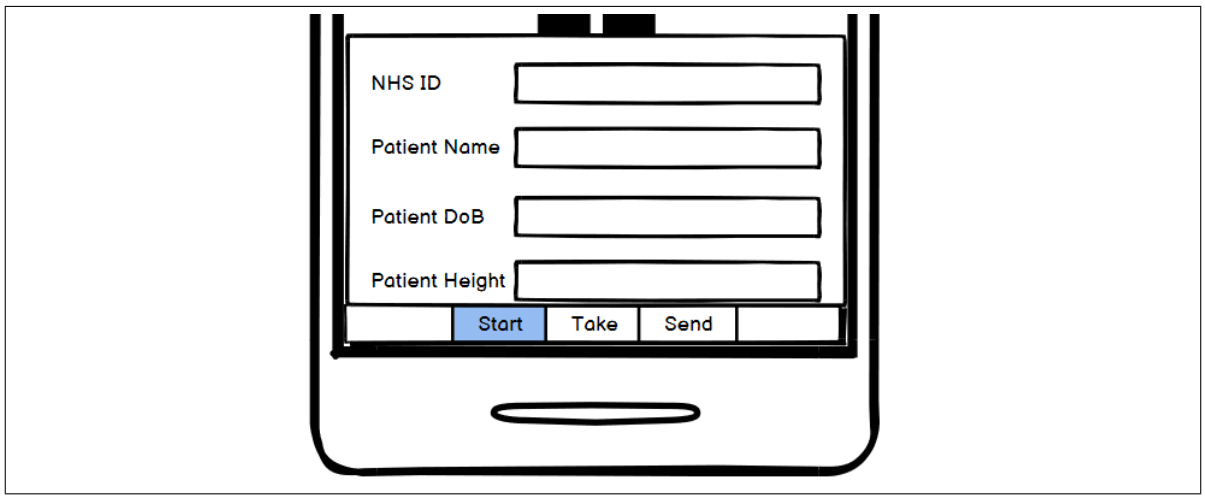

Figure 23: Wireframe mock-up of the Start stage's input panel

3.6.3 End result checklist panel

The End stage's checklist panel will also be a scrollable box that appears at the bottom of the screen. As shown in Figure 24, it will be a checkable list of previously captured photos, giving a brief preview of the subject in question.

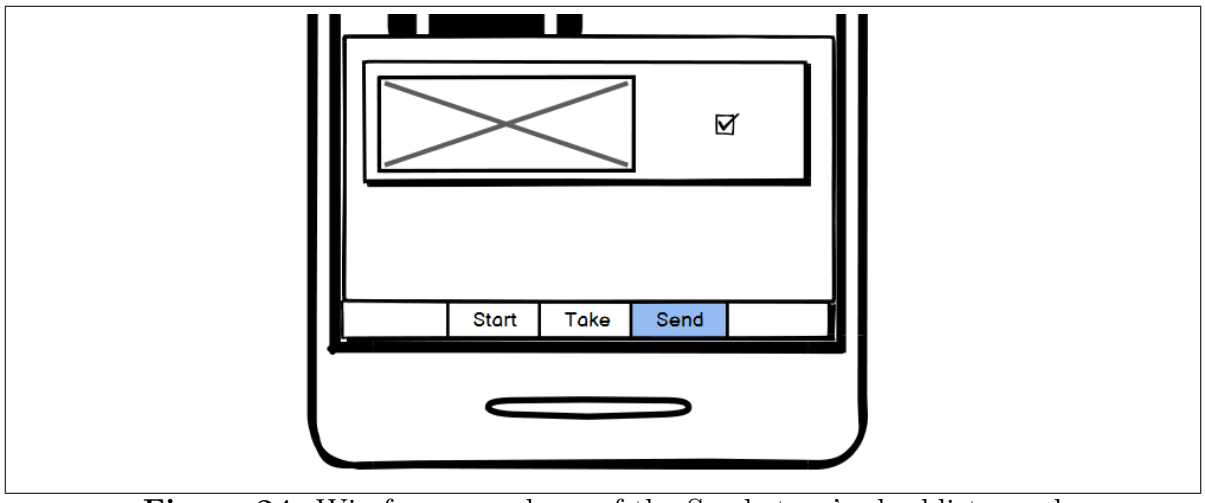

Figure 24: Wireframe mock-up of the Send stage's checklist panel

3.6.4 Point marker element

The point marker is used for mapping the placement of anatomical landmarks during the touch-to-annotate process. As shown in Figure 25, it is a simple box design and is reasonably large to allow it to be easily touched with a touchscreen device.

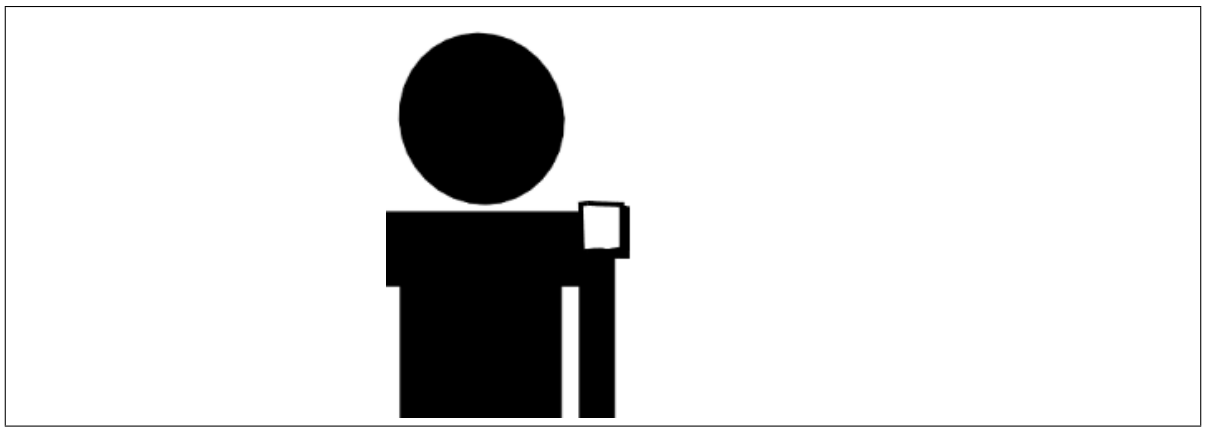

Figure 25: Wireframe mock-up of the point marker element

3.6.5 Annotation box element

The annotation box is used for writing a comment that describes an anatomical landmark or point of interest marked with a point marker. As shown in Figure 26, it takes the form of a text box with an 'X' button to allow it to be deleted if needed.

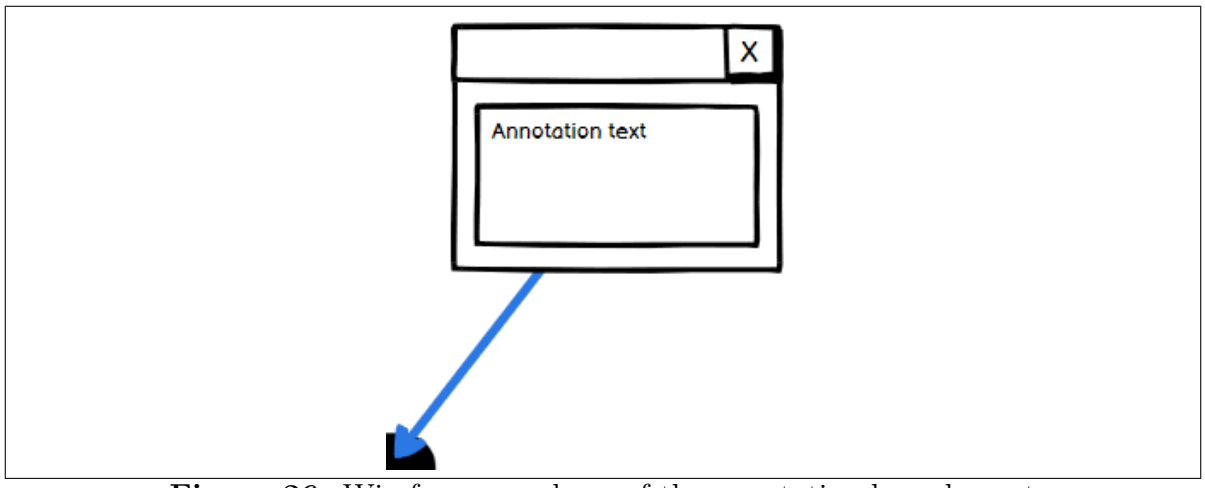

Figure 26: Wireframe mock-up of the annotation box element

4 Implementation

4.1 Scenes

The application is comprised of two Unity scenes that host all the GameObjects that implement the functionality of the application. The default scene is the ARCore scene, which features the standard real-time measurement functionality to be used by clinicians. The other seen is the fallback point-and-shoot scene, which is a simplified version of the application with the measurement functionality removed in case ARCore is not supported by the host device. On startup, the application will test for ARCore compatibility and show the appropriate scene for a given host device.

4.1.1 ARCore scene

As described above, the ARCore scene is the default scene for the application. It features the complete set of functions offered by the application: the recording of patient data, point and shoot photo capturing and annotation, real-time ARCore-powered measurement, and results to a server. The scene is comprised of the following elements:

- Canvas: the UI layer object featuring all user interface elements associated with the scene, including the menus, display readouts and annotations (when they're placed).
- EventSystem: a core Unity system used for facilitating human input into the application such as touch screen taps.
- ARCoreDevice: used for attaching ARCore session data and the ARCore SDK's plane discovery guide component for helping the user with improving the recognition of planes.
- PointCloud: used for attaching the ARCore SDK's point cloud visualizer component that periodically displays the feature points identified by ARCore to give the user an indication the application is working.
- PlaneVisualiser: used for attaching the ARCore SDK's detected plane visualizer component that maintains the visual representations of identified planes.
- AppController: used for attaching the UIController, CameraController, ScreensHolderHandler, CanvasClickHandler, AnnotationStorage and ExtrasMenu components.
- PlaneManager: used for attaching the PlaneGenerator and PlaneRanker components for rating the 'quality' of detected planes.

4.1.2 Fallback point-and-shoot scene

The fallback scene is a cut-down version of the default ARCore scene. Its feature set lacks real-time ARCore-powered measurement, however, the rest of the functions remain unchanged. The scene is comprised of the following elements:

- Canvas: the UI layer object featuring all user interface elements associated with the scene, including the menus, display readouts and annotations (when they're placed).
- EventSystem: a core Unity system used for facilitating human input into the application such as touch screen taps.
- AppController: used for attaching the UIController, CameraController, ScreensHolderHandler, CanvasClickHandler, AnnotationStorage and ExtrasMenu components.

4.2 FHIR implementation

4.2.1 Implementing the API

The implementation of the FHIR API was done as described in theory in Section 3.5. Each structure needed for compliant implementation is represented with its own serialisable class and named with the nomenclature of FHIR with the name of the structure - for example, the implementation of the bundle resource is called FHIRBundle.

Class	Description	Appendix
FHIRBundle	Container for a collection of resources, thus it is the root for the tree of data that will be serialised (HL7, 2021).	A.3.1
FHIRMedia	All the data relating to the captured image itself (HL7, 2019h).	A.3.2
FHIRAnnotation	All the data needed to reproduce an annotation that was made using the touch-to-annotate feature (HL7, 2019a).	A.3.3
FHIRMeta	Child of the bundle resource that indicates when the bundle was last updated (HL7, 2019j).	A.3.4
FHIRText	Child of the media resource that provides a narrative describing the host resource in human-readable terms (HL7, 2019i).	A.3.5
FHIRType	Child of the media resource that contains a FHIRTypeCoding child that indicates what type of media the host resource contains (HL7, 2019b).	A.3.6, A.3.7
FHIRModality	Child of the media resource that contains a FHIRModalityCoding child that provides information to better describe the host resource (such as its kind or purpose) (HL7, 2019k).	A.3.8, A.3.9
FHIRSubject	Child of the media resource that indicates who or what the subject of the host resource is (HL7, 2019d).	A.3.10
FHIROperator	Child of the media resource that indicates who captured the host resource (HL7, 2019e).	A.3.11
FHIRDevice	Child of the media resource that indicates what device captured the host resource (HL7, 2019f).	A.3.12
FHIRContent	Child of the media resource that contains data on the resource itself (HL7, 2019g).	A.3.13

4.2.2 Serialising the result

The serialisation of the JSON data needed to provide context to captured photos is done within the CaptureScreenUICmd command's BuildJSON() method. Firstly, a 'tree' is created when the BuildJSON() method assembles FHIRBundle (variable name bundle) and FHIRMedia (img) objects. Both structures in turn initialise other FHIR structures

inside themselves that form this hierarchy. After populating several of its properties, the img object is then added to bundle's entry list. After that, all annotations are serialised into FHIRAnnotation objects that are also inserted into bundle's entry list. Finally, bundle itself is then serialised into a string variable and also written into a file for backup purposes.

After that, the image file's name and the JSON are passed on to the ScreenHolderHandler component (shown in Section **??**) for storage until it's time to be sent to a server.

4.3 Controllers & Handlers components

Class	Description	Appendix
AnnotationStorage	Maintains the display of annotations when the touch-to-annotate is being used and the lifetime of line renderer objects that are used for visualising a line between sets of UI points and annotation boxes.	A.4.1
ARCoreEnabledScene Controller	Manages the ARCore session when running in real-time measurement scene.	A.4.2
CameraController	Controls the camera feed displayed to the user when the ARCore-based system is not in use (for example, when the fallback scene is being used or manual point-and-shoot has been selected as an option).	A.4.3
CanvasClickHandler	Handles user input on any UI element (ie, within the Canvas GameObject on either scene). On Also polls for possible input to see what UI elements the raycast generated from a possible screen touch hits.	A.4.4
CaptureScreenHandler	Handles when the user presses the 'Take' button.	A.4.5
ExtrasMenuHandler	Child of the media resource that contains a FHIRTypeCoding child that indicates what type of media the host resource contains (HL7, 2019b).	A.4.6
OpenFilePathHandler	Child of the media resource that contains a FHIRModalityCoding child that provides information to better describe the host resource (such as its kind or purpose) (HL7, 2019k).	A.4.7

PanelToggleHandler	Child of the media resource that indicates who or what the subject of the host resource is (HL7, 2019d).	A.4.8
PrefabsHolder	Carries and exposes various prefabricated user interface elements to be used elsewhere in the application.	A.4.9
Class	**Description**	**Appendix**
ScreensHolderHandler	Used for caching saved assessment results in lieu of being selected for sending to a server.	A.4.10
SendScreensHandler	Child of the media resource that contains data on the resource itself (HL7, 2019g).	A.4.11
ShowGalleryHandler	Child of the media resource that contains data on the resource itself (HL7, 2019g).	A.4.12
StartTakeHandler	Child of the media resource that contains data on the resource itself (HL7, 2019g).	A.3.13
ToggleHandler	Child of the media resource that contains data on the resource itself (HL7, 2019g).	A.4.14
UIHandler	Child of the media resource that contains data on the resource itself (HL7, 2019g).	A.4.15
UIController	Implements the class of the same name from the UML design in Figure 19, which is used for managing the usage and lifetime of invoked commands.	A.4.16

4.4 Real-time measurement components

4.4.1 RankedPlane

The RankedPlane class is used for transporting a reference to an ARCore-created DetectedPlane object along with some variables used for assessing plane quality. This includes a list of IDs of feature points found to be within a plane's boundaries and the total confidence of all the feature points found. A method is given for retrieving the average confidence, which is calculated by dividing total confidence by the number of items in the feature point ID list.

The RankedPlane class code is shown in Appendix A.5.1.

4.4.2 PlaneRanker

The PlaneRanker class is used for assessing the quality of a given plane. Since ARCore does not fundamentally expose how it decides when a collection of feature points is reliable enough to form a workable plane from them, the PlaneRanker attempts to implement its own quality system to help other parts of the application use the strongest planes available. Two principle values are used for assessing quality: the total number of feature points that lie within the bounds of a given plane and the average confidence of those feature points. The assessment of how close a feature point has to be to be considered within bounds is governed by a threshold value. The class has two methods - RefreshRankedPlanes() and RankRankedPlanes() - that are called one after the other every Update() cycle.

4.4.2.1 RefreshRankedPlanes

RefreshRankedPlanes simply maintains a list of possible planes stored as RankedPlane objects. Each update cycle, the method will request a list of newly tracked planes via ARCore's session tracking API, and then add any discovered planes to the list.

4.4.2.2 RankRankedPlanes

RankRankedPlanes performs some position comparisons to determine if a given feature point is within the bounds of a given plane. The method holds a nested loop; firstly for each found plane and secondly through the current list of detected feature points retrieved from ARCore's point cloud API. When looping, the code will first check if a feature point is simply close to a plane's centre position by comparing distance against the aforementioned threshold value. If it's found to be close, the feature point's ID is added to the plane's list of feature IDs and its confidence value is summed into the plane's total confidence value.

If it's not found to be close from that simple comparison, the code will then compare the Y-axis values specifically. If this check passes, the code will then compare the feature point's position against the possible extent for each plane. Once again, if a feature point is found to be within the bounds of a plane's extent, the feature point's ID is added to the plane's list of feature IDs and its confidence value is summed into the plane's total confidence value.

The PlaneRanker class code is shown in Appendix A.5.2.

4.4.3 PlaneGenerator

The PlaneGenerator class is used for visualising detected planes on the scene as a visual aid for the user. The class is based on the ARCore SDK's DetectedPlaneVisualiser, although it is rewritten to use the plane quality assessment provided by the PlaneRanker class. Using the quality ranking, it will colour-code the detected plane visualisations to indicate which planes are of high quality. The colour rules are as follows.

- If the plane's colour is green, it is considered a strong plane.
- If the plane's colour is red, it is considered a weak plane.
- If the plane's colour is black, it is still being assessed and thus shouldn't be used.

The PlaneGenerator class code is shown in Appendix A.5.3.

4.5 UI command implementations

Class	Description	Appendix
IUICmd	Interface for all other classes to implement.	A.6.1
AnnotateSubject	Places UI elements for an annotation. It will make a UI point or annotation box depending on the status reported by the AnnotationStorageHandler.	A.6.2
CaptureScreen	Captures a screen and saves the resulting image and possible annotation data into a JSON pack.	A.6.3
ScreensCounterUpdate	Updates UI text element used for displaying the number of saved screens presently stored on the host device.	A.6.7
SendScreensUICmd	Sends held screens to a remote server.	A.6.8
ShowGalleryUICmd	Opens and populates the UI element that contains the checkbox list of currently-stored screens.	A.6.9
StartTakeUICmd	Begins the touch-to-annotate process when ARCore is not being used.	A.6.10
TogglePanelUICmd	Generic UI element display toggle, typically invoked when a menu button is used.	A.6.11

4.6 Remote test server

For use with developing and testing the remote data transfer functionality, a test server was set up. This test server was a PHP application with a MySQL database and was deployed via Docker for the duration of the last two phases of the project.

File	Description	Appendix
db_engine.php	Code for handling data being inserted and retrieved from a backend MySQL database.	A.7.1
index.php	Basic index page showing results that have been sent to the server.	A.7.2
result.php	Page for viewing the JSON content of a given result.	A.7.3
table_structure.sql	SQL file containing the table definition for the database.	A.7.4
upload.php	Page the application sends the JSON result via POST request.	A.7.5

5 Evaluation

5.1 Measurement reliability

To assess the basic accuracy of the software, two tests were devised that would attempt to evaluate if the application can measure the same distance as a manual measurement within a certain range of accuracy. REU staff have quoted their off-hand expected accuracy from clinicians' freehand measurement (approximately 1 to 3cm), so to obtain a more solid benchmark for comparison, literature on posture measurement repeatability was considered. The typical measurement routine was starting the application, sweeping the area several times to allow ARCore to gain the needed understanding of the environment, and then attempting to mark the two targets within the application.

5.1.1 Target reliability

As aforementioned, clinicians stated their expected 1 to 3cm accuracy for their current measurements. However, this was considered anecdotal, so literature on posture assessment reliability and repeatability was reviewed to obtain more concrete targets. Mcginley et al. (2008) found that most studies on three-dimensional gait analysis reported standard deviations or errors of less than 5 degrees and summarised that a variability of 2 degrees or less is considered 'acceptable', between 2 to 5 degrees is considered 'reasonable', and 5 degrees or more is considered 'concerning'. Thus, these values were used as a benchmark for this application's reliability with each subsequent test below reporting their results' mean, median, standard deviation, standard error and coefficient of variation. Less than 2cm deviation would be considered the ideal accuracy, and 2cm to 5cm deviation would be considered acceptable.

5.1.2 Measuring a surface

For this first test, the application's accuracy was evaluated against a measuring tape on a well-lit surface, with 0 and 30 centimetres being the target positions on the tape to measure between.

Figure 27: Surface measurement test control

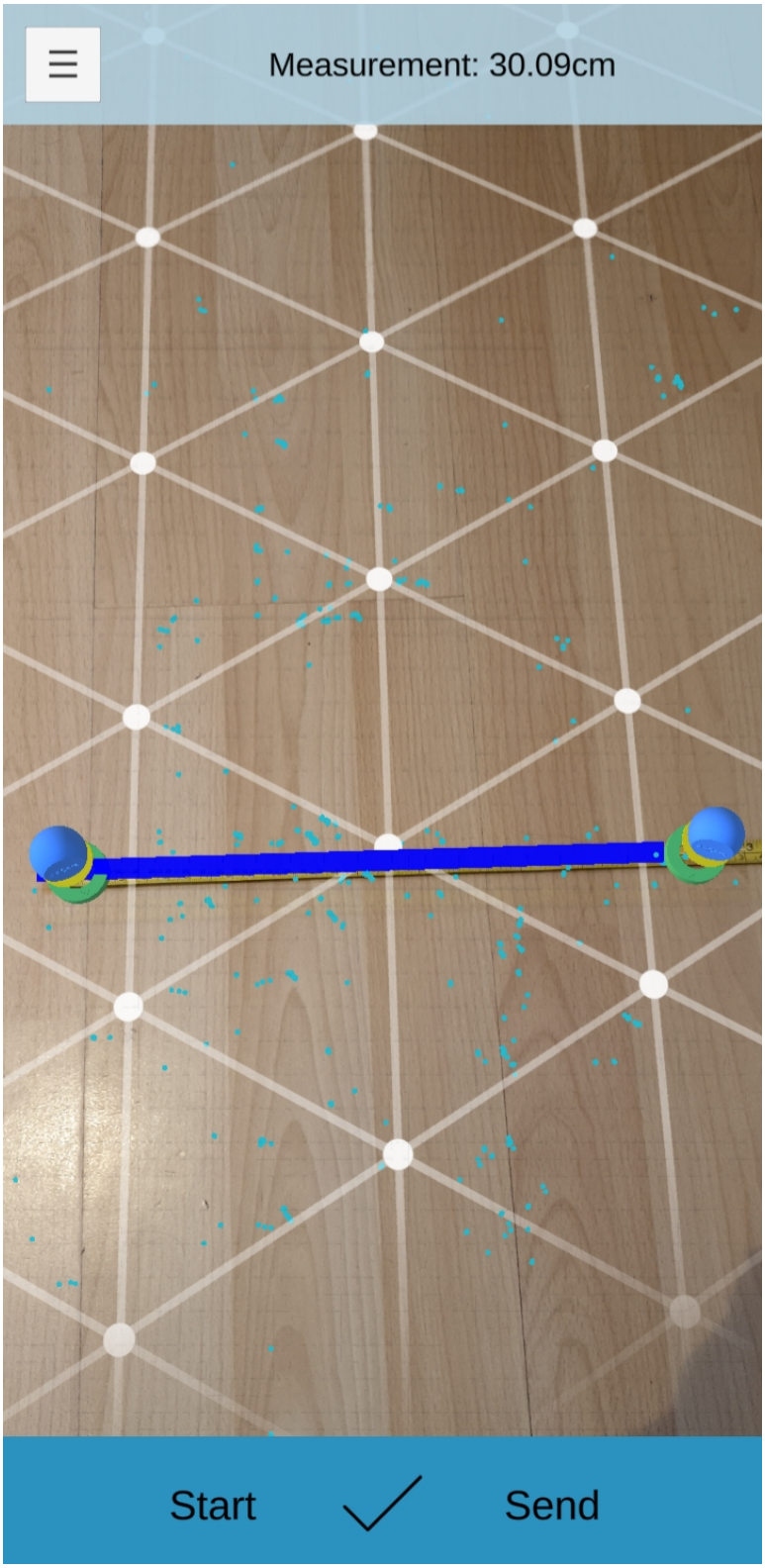

Figure 28: Example of taking a measurement of surface within the application

#	**Measurement**	**Difference to manual**
1	29.93cm	-0.07cm
2	30.09cm	+0.09cm
3	30.45cm	+0.45cm
4	30.42cm	+0.42cm
5	30.21cm	+0.21cm
6	28.71cm	-1.29cm
7	30.36cm	+0.36cm
8	30.83cm	+0.83cm
9	30.45cm	+0.45cm
10	30.67cm	+0.67cm
Mean	30.212cm	+0.212cm
Median	30.39cm	+0.39cm
Standard deviation	0.589cm	-
Standard error	0.186cm	-
Coefficient of variation	1.95%	-

Figure 29: Table of measurements taken of a surface from the application to compare against manual measurement

5.1.3 Measuring a subject

The control for this experiment was using the same mannequin in the same well-lit lighting conditions with two pieces of tape on either side of the upper torso as targets for the measuring tape and the application's markers. The measured distance was approximately 26 centimetres.

Figure 30: Subject measurement test control

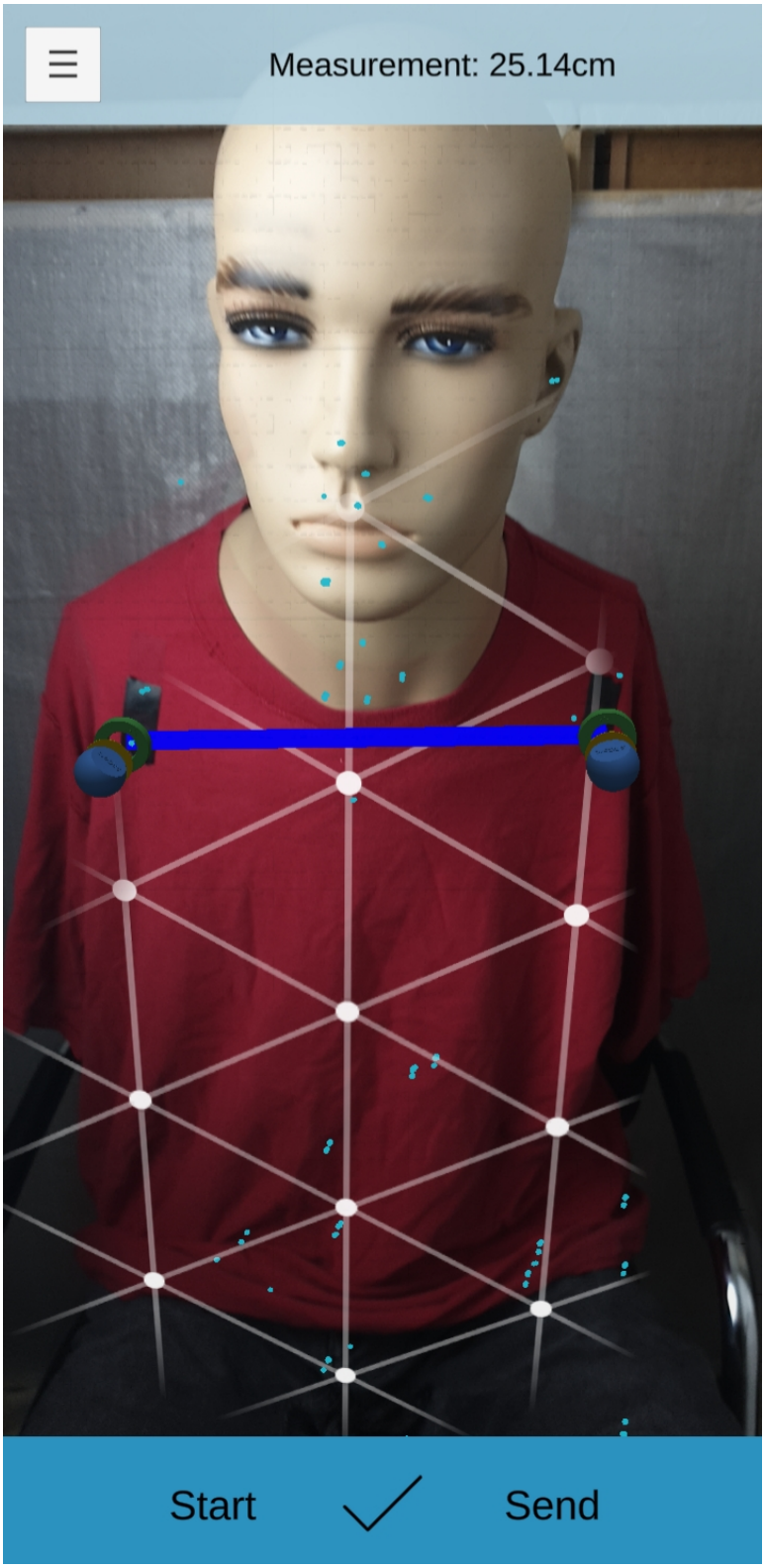

Figure 31: Example of taking a measurement of subject within the application

#	Measurement	Difference to manual
1	26.32cm	+0.32cm
2	24.58cm	-1.42cm
3	29.57cm	+3.57cm
4	25.3cm	-0.7cm
5	25.74cm	-0.26cm
6	25.14cm	-0.86cm
7	26.12cm	+0.12cm
8	25.76cm	-0.24cm
9	27.10cm	+1.1cm
10	25.46cm	-0.54cm
Mean	26.11cm	+0.11cm
Median	25.75cm	-0.25cm
Standard deviation	1.399cm	-
Standard error	0.442cm	-
Coefficient of variation	5.358%	-

Figure 32: Table of measurements taken of a test subject from the application to compare against manual measurement

5.1.4 Results

It was determined that:

- As per mean and median, the application could measure within the expected accuracy from clinicians' manual measurements.
- As per standard deviation, the application could measure within a deviation considered 'acceptable' by literature.

As shown by Figure 32, test #3 was the only measurement attempt that breached the expected range by measuring 29.57cm between the two pieces of tape. Tests #2 and #9 were the only other tests whose difference from manual measurement was above 1cm, with the remaining 70% showing a difference of less than 1cm from manual measurement.

5.1.5 Additional informal testing

During the development of the application, testing how real-time measurement performs at various distances was also conducted but not formally completely as a controlled test. It was found that testing significant distances was problematic due to the difficulty in placing markers with fingers on the constant phone screen size. This could only

be mitigated with a new feature addition that grants zoom capabilities, which is a consideration for future development.

5.2 Clincian interactions

The first and third objectives demanded interaction with clinical staff to obtain data needed to develop and improve the application. The first round of data collecting was the 13th November 2019 interview with PMC staff. The main takeaways from this interaction were:

- There was minimal to no data collecting standardisation.
- Attempted technological solutions did not meaningfully fix the problem.
- Established applications on the market as of Q4 2019 did not receive a warm reception.

The second round of data collecting was the 4th January 2021 survey with PMC staff, where they were asked to watch a video demonstration of the application, review an interactive PDF model of the application's features and answer a series of questions regarding planned features and it's three-step process. The main takeaways from this interaction were:

- 75% to 80% thought the three steps of operation were 'easy and intuitive to use.
- All respondents agreed the user interface was easy to understand and navigate.
- Data points of the initial patient information entry and annotation templates were suggested.

5.3 Code Quality

The code is largely produced to a clean and commented standard with appropriate use of classes and methods for compartmentalising code. If given more time, some improvements could be made, however. Due to trying various different solutions whilst debugging Unity compilation issues and attempting to harness ARCore for real-time measurement, some instances of leftover comments and disused code are still present. Ideally, these should be documented to explain why they were tried and subsequently removed. There are also instances of lacking comments to explain variables and some functionality. Attempts to plug these gaps were made to great effect near the project end, but some gaps may remain.

Outside of the command pattern, FHIR and ARCore Unity SDK compliance outlined in the next section, appropriate use of Unity's in-built structures was also utilised.

Use of the GameObject-Component pattern is a given since it's a requirement for attaching any functionality to objects in the application's scenes, but appropriate use was made of Unity's JsonUtility to parse entire classes representing FHIR structures into JSON and of coroutines to allow code to execute outside of the typical Update pattern when needed.

5.4 Design & Tool Choices

The main design choices for the application were the command-based design pattern, HL7 FHIR API, and the use of Unity and ARCore as the development tools. The command pattern allows for a versatile and easy means to add and manage functionality. The encapsulation of commands allows the code to be easily read and understood by a programmer and allows the commands to be passed around and manipulated by the code. Another benefit of the system implemented on the application is that commands are queued to prevent any race conditions from being met. HL7 FHIR is an industry standard, so its inclusion was essential for this application. Nonetheless, it provided a robust API for packaging and sending data from the application and to a remote server. The needs of the various FHIR structures also helped inform what sort of data should be requested and stored in other parts of the application.

As evidenced from the previous subsection, ARCore ultimately proved to be a useful and reasonably accurate toolkit to use for providing real-time measurement. However, it was not without its complexities though. The main roadblock encountered was that much of the low-level processing is hidden away from access and manipulation under the official ARCore Unity SDK and the Unity project at large, meaning being able to accurately determine how ARCore was calculating and constructing planes and being able to adjust its thinking was not possible. Hence, a post-processing plane ranking system had to be implemented at a higher code level to determine the quality of prospective planes since ARCore could not expose the needed information to determine this instantly.

The Unity engine overall was suitable for the task at hand, but there were a few hiccups in getting the application to compile into a working Android build. Issues experienced ranged from Unity being unable to locate the needed Android SDK and NDK toolkits, Gradle version mismatches, and in some cases stalled APK building that made no progress even after allowing almost two hours for it to build.

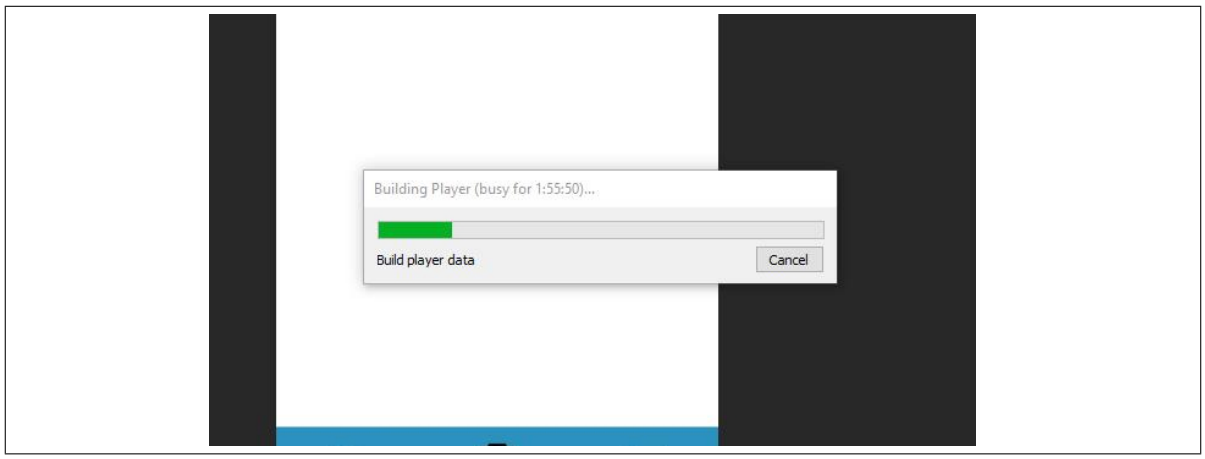

Figure 33: Stalled Android application building in Unity 2021

Build stability was eventually achieved by using a long-term support version of Unity 2019 and supplying external copies of the Android SDK and NDK and Gradle for Unity to use as the build toolchain.

6 Conclusion

6.1 Meeting objectives

As outlined within Section 1, three objectives were established at the start of this project as goals to achieve the aim of developing a mobile application that can be used to assist PMC clinicians by enhancing the postural assessment data collection process.

6.1.1 First objective

The first objective was to investigate the present issues with the PMC's postural assessment process. This was attempted by an initial interview with staff across the PMC and a follow-up questionnaire survey once a prototype of the application was completed. The interview was held early in the project on 13th November 2019. The full details of what was discovered can be read in Section 2.5. But, the important details learned were that standardisation of data recording was currently not present, previous attempts to rectify this with standardised data entry and technology did not meaningfully fix the problem, and existing posture assessment applications did not receive a warm reception. From this, information on what PMC staff do and do not want was gathered that helped shape how the application would be developed. Of note, the desired features of this application were ease of use, a 'touch-to-mark' annotation system, and automatic measurement.

The questionnaire was held late in the project, beginning on 4th January 2021 and was allowed to run for two weeks. At this stage in the project, a prototype application capable of meeting the second objective described below was developed. This questionnaire was designed to ensure the application was being tailored for their needs, thus the questions asked surrounded evaluating the presentation of the features and requesting any suggestions to improve them. Overall, the response was positive. Suggestions for new data points, annotation templates, and how to improve the user interface were also given and are explained in Section 2.6.

Both attempts at gauging PMC staff opinion yielded information critical to the development of this project. The responses from the initial questionnaire established what else this project needed to achieve to be considered a successor (namely, the other two objectives described below). The questionnaire later reaffirmed that the project was progressing on the right track by confirming that using the application was an intuitive experience and that the features then in development were useful.

6.1.2 Second objective

The second objective was to develop an application capable of gathering data, both by manual point-and-shoot and real-time measurement techniques, needed to aid a posture assessment in a way that is standardised and easy to use. The approach taken to achieve this was using industry standards, developing features outlined in the PMC staff interview, and informed user interface design.

The basis industry standard used was HL7 FHIR. This is a framework designed specifically for transmitting clinical and administrative healthcare data for contexts such as a mobile application. FHIR provides an API that enforces how data is handled within the application and sent to a remote server. This means any output from the application is guaranteed to be in the same format. This was tested against FHIR's test servers and a self-implemented server for validation. For this 'middle' objective, photo taking and annotation were the core features to be developed in this part of the project. This ultimately resulted in a 'point and shoot' style application that allows the user to annotate any part of the subject photo in a 'free-hand' manner that they are already accustomed to, but thanks to the FHIR implementation, it's ultimately stored and transmitted in a standardised format with the coordinates of the annotation's position preserved for reference.

Through research, it was found that the key to a user-friendly design was getting an effective suitable menu design and structure. The benefits and drawbacks of major menu types were considered and a hybrid approach was ultimately taken in an attempt to strike a balance between simplicity and ensuring any secondary options were visible when needed. The main navigation through the application is done through a bottom bar-style menu that has buttons representing each of the three stages of use for this application, allowing for simple progression throughout the data gathering, selection, and sending process. A secondary menu is provided by a 'hamburger'-style menu whose options change depending on the current stage. The menu used for the secondary menu is selecting templated annotations.

Implementing a self-designed system to achieve real-time measurement achieve would require mechanisms capable of correcting lens distortion, handling different lighting conditions, and using image data processing to gain an understanding of the environment. Due to the predicted complexity of developing such a solution for a single-year project, the use of emerging technologies was considered and ultimately applied. Google's ARCore framework was chosen due to its wide range of device support and availability of measurement applications using the framework that indicated viability. However, employing ARCore did not complete this objective entirely. Most AR-powered measurement applications found relied on distinct, horizontal surfaces to operate. Measuring the human body creates additional variables for reliability, thus a new mechanism (in the form of plane ranking) was devised to help ensure only planes with high confidence and large extents were used as bases for markers.

6.1.3 Third objective

The third objective was to evaluate the application during and after development to ensure the system is accurate and reliable enough to be a viable supplement or replacement to manual measuring techniques. To that end, a prototype of the application was demonstrated to clinicians and the application's reliability was tested.

A functional prototype of the application during a late stage of development was made available for the PMC questionnaire, which allowed for gathering feedback on the effectiveness of this application. As already stated in the analysing of the first objective, the application was deemed easy to use and intuitive by the respondents.

When it came to testing, there was an informal and a formal benchmark. Clinicians had quoted an expected accuracy of current measurement techniques to be 1 to 3cm, but reviewing literature found a more solid benchmark. Due to COVID-19, testing of the application was done against a dressed commercially-bought mannequin. Reliability and accuracy were tested by measuring the mannequin and comparing the results to manual measurements taken with a measuring tape, which is tried and explained in the evaluation. It was ultimately found that the application could perform within the expected accuracy range as the testing results provided a deviation of less than 2cm, with 2 or less being considered 'acceptable' variance by a review papers.

6.2 Impact & presentation

On 13th October 2021, several aspects of this project were presented at the All-Wales Medical Physics & Clinical Engineering Summer Meeting. Explained was the background issues that led to the application's development, its core features, and how accurate it was at measuring during testing.

It was also learned that even before the total conclusion of the project, the impact of this work has already started to take shape. The O365 National Programme team (a part of NHS' Digital Health and Care Wales department) have started developing a new application with capabilities derived from what was developed in this project. The main difference between that app and this one is that it is being developed with Microsoft Power Apps compatibility in mind.

6.3 Future work & improvements

6.3.1 Further interaction with clinical staff

Ideally, the project would have benefited greatly from more interaction with PMC staff. Plans to observe a posture assessment and provide live trials of the applications were considered. However, the circumstances brought forth by the COVID-19 pandemic

restricted the project's access to clinicians greatly. Insight into their posture assessment process, the ability to conduct live tests with the application, and further feedback were lost due to this. Whilst the objectives described above were still rudimentarily met, even limited clinician access could have helped sharpen design goals further and facilitated easier and more involved user engagement. This would have further enhanced the usefulness of the application, bringing it another step closer to a viable tool to enhance data gathering during a postural assessment.

6.3.2 Expanded user access to the application for testing

For further validation of the application, granting wider access to staff outside the PMC and to patients could be considered. Involving more staff could gather more broad insight and increase the validity of any testing and evaluation. This could potentially be done as a future round of testing and then any new feedback could be considered and implemented in a future application revision. Involving patients in a real-life setting could be invaluable for gathering more feedback and identifying potential issues in the field. This testing could also be compared against a clinician's manual measurement for further validation of the application's capability.

6.3.3 More comprehensive testing

Whilst the application was found to perform desirably in the testing carried out, said testing was limited in scope to what could be considered ideal lighting conditions. It would be useful to further test the application under more adjusted variables to understand what its operating limits are, which could be used to inform recommended operating conditions to users. Eklind and Stark (2018) have explored the capability of ARCore's feature detection, which could be used as inspiration for further testing, including validation against different light intensities and angles. Different clothing on the patient could also be another variable to test.

6.3.4 Additional annotation templates

The application has only two annotation templates implemented. Whilst these sufficiently demonstrate the feature is operational, they do not represent the wide spectrum of tests a clinician may want to conduct and describe. The addition of more templates would be a straightforward task with no additional code required; create new prefabrications of annotation boxes in the approximate location and add the option to place it via Unity's scene designer.

6.3.5 Zoom on press marker placement

On reflection, a potential concern with placing annotations and markers - especially when viewing an individual at distance - is that they could be hard to place with just the user's finger. A possible feature to mitigate this could be providing a zoom when the user taps the screen, allowing for a more clear placement. Such a feature could operate by measuring the succession of the user's presses to accept an initial zoom press and then a placement press.

6.3.6 Evaluate application's efficacy

Finally, further study into how well the application improves data recording efficiency and how effective it is at standardising said data recording would be an important follow-up to ensure the ultimate goal of making a long-term solution to the initial problems identified is realised.

7 References

Abreu, L. (2014). *Why and How to Avoid Hamburger Menus*. URL: https://lmjabreu.com/post/why-and-how-to-avoid-hamburger-menus/ (visited on 19/04/2021).

Ahmed Zemirline, A. Z. (2019). *User's guide - iGonio*. URL: https://sites.google.com/site/igoniowebsite/user-s-guide (visited on 04/11/2019).

Akhtar, S. (2020). *Implementing a Command Design Pattern in Unity*. URL: https://faramira.com/implementing-a-command-design-pattern-in-unity/ (visited on 26/05/2021).

Apple App Store (2019). *PostureZone on the App Store*. URL: https://apps.apple.com/us/app/posturezone/id631253399?ign-mpt=uo%5C%3D4 (visited on 04/11/2019).

Babich, N. (2020). *Essential Patterns of Mobile Navigation*. URL: https://xd.adobe.com/ideas/principles/app-design/essential-patterns-mobile-navigation/ (visited on 07/06/2021).

Baron, D. (2019). *Unity's architecture*. URL: https://subscription.packtpub.com/book/game_development/9781789349337/1/ch01lvl1sec11/unity-s-architecture (visited on 26/05/2021).

BodyZone LLC (2019). *Posture Assessment Software*. URL: https://www.posturezone.com/products/PostureZone-Software-BOXED.html (visited on 04/11/2019).

BodyZone LLC (2021). *PostureZone App*. URL: https://www.posturezone.com/products/PostureZone-Software-BOXED.html (visited on 23/06/2021).

Boston Children's Hospital (2021). *Lordosis*. URL: https://www.childrenshospital.org/conditions-and-treatments/conditions/l/lordosis (visited on 19/06/2021).

CareFlex (2021a). *Anterior Pelvic Tilt*. URL: https://www.careflex.co.uk/health/posture/anterior-pelvic-tilt/ (visited on 21/06/2021).

CareFlex (2021b). *Pelvic Rotation*. URL: https://www.careflex.co.uk/health/posture/pelvic-rotation/ (visited on 21/06/2021).

CareFlex (2021c). *Posterior Pelvic Tilt*. URL: https://www.careflex.co.uk/health/posture/posterior-pelvic-tilt/ (visited on 21/06/2021).

CareFlex (2021d). *Spotlight On Pelvic Obliquity*. URL: https://www.careflex.co.uk/info-centre/blogs/spotlight-on-pelvic-obliquity/ (visited on 21/06/2021).

CareFlex (2022). *Postural Management and Good Sitting Posture – What's All the Hype About?* URL: https://www.careflex.co.uk/info-centre/blogs/postural-management-and-good-sitting-posture-whats-all-the-hype-about/ (visited on 13/06/2022).

Cleveland Clinic (2020). *Kyphosis: Symptoms, Types, Treatments*. URL: https://my.clevelandclinic.org/health/diseases/17671-kyphosis (visited on 19/06/2021).

Colorado Comprehensive Spine Institute (2016). *Understanding Spinal Anatomy: Ligaments, Tendons and Muscles*. URL: https://www.coloradospineinstitute.com/education/anatomy/ligaments-tendons-muscles/ (visited on 18/06/2021).

cPrime (2021). *What is AGILE? — What is SCRUM?* URL: https://www.cprime.com/resources/what-is-agile-what-is-scrum/ (visited on 23/04/2021).

Cruickshank, J., M. Koike and R. Dickson (1989). 'Curve patterns in idiopathic scoliosis. A clinical and radiographic study'. In: *The Journal of Bone and Joint Surgery. British volume* 71-B.2. PMID: 2925744, pp. 259–263. DOI: 10.1302/0301-620X.71B2.2925744. eprint: https://doi.org/10.1302/0301-620X.71B2.2925744. URL: https://doi.org/10.1302/0301-620X.71B2.2925744.

Eklind, A. and L. Stark (2018). 'An exploratory research of ARCore's feature detection'. PhD thesis. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-254357.

Eldad, A. (2021). *Unity vs Unreal, What Kind of Game Dev Are You?* URL: https://www.incredibuild.com/blog/unity-vs-unreal-what-kind-of-game-dev-are-you (visited on 24/08/2021).

Ferrantelli, J. (2017). *PostureScreen First Assessment*. URL: https://www.youtube.com/watch?v=5BFJM3pXBeE (visited on 23/06/2021).

Franko, O. (2011). *Scoligauge — TopOrthoApps*. URL: http://toporthoapps.com/2011/11/01/scoligauge/ (visited on 04/11/2019).

Fu, Y., H. Jiang, D. Zhang and X. Zhang (2019). 'Comparison of Perceptual Differences Between Users and Designers in Mobile Shopping App Interface Design: Implications for Evaluation Practice'. eng. In: *IEEE access* 7, pp. 23459–23470.

GetBodySmart (2021). *Hip Bone Anatomy – Medial or Internal Markings*. URL: https://www.getbodysmart.com/lower-limb-bones/hip-bone-anatomy-medial-or-internal-markings (visited on 30/06/2021).

Glover, J. (2017). *Learn and Understand Raycasting in Unity3D*. URL: https://gamedevacademy.org/learn-and-understand-raycasting-in-unity3d/ (visited on 06/06/2021).

Google (2020a). *GoogleARCore.CameraIntrinsics*. URL: https://developers.google.com/ar/reference/unity/struct/GoogleARCore/CameraIntrinsics?hl=en (visited on 06/06/2021).

Google (2020b). *GoogleARCore.PointCloud*. URL: https://developers.google.com/ar/reference/unity/class/GoogleARCore/Frame/PointCloud?hl=en (visited on 06/06/2021).

Google (2020c). *GoogleARCore.TrackableHit*. URL: https://developers.google.com/ar/reference/unity/struct/GoogleARCore/TrackableHit?hl=en (visited on 06/06/2021).

Google (2021a). *Fundamental concepts — ARCore*. URL: https://developers.google.com/ar/discover/concepts?hl=en (visited on 05/06/2021).

Google (2021b). *GoogleARCore.CameraConfig*. URL: https://developers.google.com/ar/reference/unity/struct/GoogleARCore/CameraConfig?hl=en (visited on 06/06/2021).

Google (2021c). *GoogleARCore.LightEstimate*. URL: https://developers.google.com/ar/reference/unity/struct/GoogleARCore/LightEstimate?hl=en (visited on 06/06/2021).

Google Play (2019). *PostureZone - Apps on Google Play*. URL: https://play.google.com/store/apps/details?id=com.posturezone (visited on 04/11/2019).

Google Play (2021). *PostureScreen Mobile - Apps on Google Play*. URL: https://play.google.com/store/apps/details?id=com.postureco.posturescreen (visited on 23/06/2021).

GOV.UK (2021). *Medical devices: software applications (apps)*. URL: https://www.gov.uk/government/publications/medical-devices-software-applications-apps (visited on 18/06/2022).

Harvey, R. (2020). *Why Posture Management Is Important*. URL: https://www.rifton.com/adaptive-mobility-blog/blog-posts/2020/july/why-postural-management-is-important (visited on 13/06/2022).

Healthline (2021). *Everything You Need to Know About Scoliosis*. URL: https://www.healthline.com/health/scoliosis (visited on 19/06/2021).

Highsmith, J. M. (2021). *Spinal Anatomy Center*. URL: https://www.spineuniverse.com/anatomy (visited on 18/06/2021).

Hillman, S. J. and J. Hollington (2016). 'A quantitative measurement method for comparison of seated postures'. In: *Medical Engineering Physics* 38.5, pp. 485–489. DOI: https://doi.org/10.1016/j.medengphy.2016.01.008. URL: https://www.sciencedirect.com/science/article/pii/S1350453316000291.

HL7 (2019a). *Datatypes - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/datatypes.html#Annotation (visited on 30/05/2021).

HL7 (2019b). *Datatypes - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/datatypes.html#Coding (visited on 30/05/2021).

HL7 (2019c). *Introducing HL7 FHIR*. URL: https://www.hl7.org/fhir/summary.html (visited on 26/05/2021).

HL7 (2019d). *Media - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/media-definitions.html#Media.subject (visited on 30/05/2021).

HL7 (2019e). *Media - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/media-definitions.html#Media.operator (visited on 30/05/2021).

HL7 (2019f). *Media - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/media-definitions.html#Media.device (visited on 30/05/2021).

HL7 (2019g). *Media - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/media-definitions.html#Media.content (visited on 30/05/2021).

HL7 (2019h). *Media-example.json - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/media-example.json.html (visited on 26/05/2021).

HL7 (2019i). *Narrative - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/narrative.html (visited on 30/05/2021).

HL7 (2019j). *Resource - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/resource.html (visited on 30/05/2021).

HL7 (2019k). *Valueset-media-modality - FHIR v4.0.1*. URL: https://www.hl7.org/fhir/valueset-media-modality.html (visited on 30/05/2021).

HL7 (2021). *Bundle - FHIR v4.0.1*. URL: hthttps://www.hl7.org/fhir/bundle.html (visited on 26/05/2021).

Information Comissioner's Office (2022a). *Principle (f): Integrity and confidentiality (security)*. URL: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/integrity-and-confidentiality-security/ (visited on 17/06/2022).

Information Comissioner's Office (2022b). *What is valid consent?* URL: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/consent/what-is-valid-consent/ (visited on 17/06/2022).

kanbanize (2019). *Different Agile Methodologies Explained*. URL: https://www.kanbanize.com/blog/right-agile-methodology-for-your-project (visited on 23/04/2021).

Kayalioglu, G. (2009). 'Chapter 3 - The Vertebral Column and Spinal Meninges'. In: *The Spinal Cord*. Ed. by C. Watson, G. Paxinos and G. Kayalioglu. San Diego: Academic Press, pp. 17–36. DOI: https://doi.org/10.1016/B978-0-12-374247-6.50007-9. URL: https://www.sciencedirect.com/science/article/pii/B9780123742476500079.

Kenhub (2021). *Bony pelvis: Ilium, ischium, pubis*. URL: https://www.kenhub.com/en/library/anatomy/the-pelvis (visited on 21/06/2021).

Lavieri, E. (2015). *Getting Started with Unity 5*. Birmingham: Packt Publishing Ltd.

Lumen Learning (2021). *The Pelvic Girdle and Pelvis*. URL: https://courses.lumenlearning.com/suny-ap1/chapter/the-pelvic-girdle-and-pelvis/ (visited on 21/06/2021).

Mansfield, P. J. and D. A. Neumann (2019). 'Chapter 8 - Structure and Function of the Vertebral Column'. In: *Essentials of Kinesiology for the Physical Therapist Assistant (Third Edition)*. Ed. by P. J. Mansfield and D. A. Neumann. Third Edition. St. Louis (MO): Mosby, pp. 178–232. DOI: https://doi.org/10.1016/B978-0-323-54498-6.00008-4. URL: https://www.sciencedirect.com/science/article/pii/B9780323544986000084.

Mayfield Brain Spine (2018). *Scoliosis, Degenerative scoliosis, adult spinal deformity, curvature of the spine*. URL: https://www.coloradospineinstitute.com/education/anatomy/ligaments-tendons-muscles/ (visited on 18/06/2021).

Mayo Clinic (2021a). *Kyphosis*. URL: https://www.mayoclinic.org/diseases-conditions/kyphosis/symptoms-causes/syc-20374205 (visited on 18/06/2021).

Mayo Clinic (2021b). *Scoliosis*. URL: https://www.mayoclinic.org/diseases-conditions/scoliosis/symptoms-causes/syc-20350716 (visited on 18/06/2021).

Mcginley, J., R. Baker, R. Wolfe and M. Morris (Dec. 2008). 'The reliability of three-dimensional kinematic gait measurements: A systematic review'. In: *Gait posture* 29, pp. 360–9. DOI: 10.1016/j.gaitpost.2008.09.003.

Microsoft Corporation (2021). *SerializableAttribute Class (System) — Microsoft Docs*. URL: https://docs.microsoft.com/en-us/dotnet/api/system.serializableattribute?view=net-5.0 (visited on 28/05/2021).

Moen, R. D. (2020). *Why 24-hour posture care management is key*. URL: https://blog.madeformovement.com/24-hour-posture-care-management-key (visited on 20/06/2022).

Mount Sinai (2021). *Lordosis - lumbar*. URL: https://www.mountsinai.org/health-library/symptoms/lordosis-lumbar (visited on 18/06/2021).

Negrini, S., M. Romano, P. Pizzetti, F. Saveri and V. Ziliani (Dec. 2009). 'Arm positioning and postural sagittal variation: are kyphosis and lordosis measurements using x-ray reliable?' eng. In: *Scoliosis* 4.Suppl 2. PMC2793443[pmcid], O18–O18. DOI: 10.1186/1748-7161-4-S2-O18. URL: https://doi.org/10.1186/1748-7161-4-S2-O18.

NHS (2018). *Kyphosis*. URL: https://www.nhs.uk/conditions/kyphosis/ (visited on 19/06/2021).

NHS Education for Scotland (2017). *Postural care – protection of body shape*. URL: https://www.nes.scot.nhs.uk/media/dzwncusg/postural_care_learning_byte.pdf (visited on 13/06/2022).

Norton, K. and T. Olds (1996). *Anthropometrica: A Textbook of Body Measurement for Sports and Health Courses*. UNSW Press. URL: https://books.google.co.uk/books?id=Bkk8FuB0P4IC.

OriGym (2020). *Postural Assessment: Guide for Fitness Professionals*. URL: https://origympersonaltrainercourses.co.uk/blog/postural-assessment (visited on 01/07/2021).

Oxford Reference (2021a). *acromiale*. URL: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095348122 (visited on 30/06/2021).

Oxford Reference (2021b). *radiale*. URL: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100400574 (visited on 30/06/2021).

Panunzio, M. and T. Vardanega (2014). 'A component-based process with separation of concerns for the development of embedded real-time software systems'. In: *Journal of Systems and Software* 96, pp. 105–121. DOI: https://doi.org/10.1016/j.jss.2014.05.076. URL: https://www.sciencedirect.com/science/article/pii/S0164121214001381.

Partlow, A., C. Gibson and J. Kulon (2021). '3D Posture Visualisation From Body Shape Measurements Using Physics Simulation, to Ascertain the Orientation of the Pelvis and Femurs in a Seated Position'. eng. In.

Physiopedia (2021). *Anatomy of the Pelvic Girdle*. URL: https://www.physio-pedia.com/Anatomy_of_the_Pelvic_Girdle (visited on 21/06/2021).

PostureCo, Inc. (2018). *PostureScreen Posture Analysis Assessment Evaluation App Software iPad*. URL: https://www.postureanalysis.com/posturescreen-posture-movement-body-composition-analysis-assessment/ (visited on 04/11/2019).

PostureCo, Inc. (2019a). *Augmented Reality Supported Devices*. URL: https://www.postureanalysis.com/knowledge-base/augmented-reality-supported-devices/ (visited on 04/11/2019).

PostureCo, Inc. (2019b). *PostureScreen Augmented Reality (AR)*. URL: https://www.postureanalysis.com/knowledge-base/posturescreen-augmented-reality-ar/ (visited on 04/11/2019).

Radiopaedia (2021a). *Anterior superior iliac spine*. URL: https://radiopaedia.org/articles/anterior-superior-iliac-spine?lang=gb (visited on 30/06/2021).

Radiopaedia (2021b). *Vertex*. URL: https://radiopaedia.org/articles/vertex?lang=gb (visited on 30/06/2021).

Seffinger, M. A. and R. J. Hruby (2007). 'CHAPTER 3 - Manual Diagnostic Procedures Overview'. In: *Evidence-Based Manual Medicine*. Ed. by M. A. Seffinger and R. J. Hruby. Philadelphia: W.B. Saunders, pp. 35–58. DOI: https://doi.org/10.1016/B978-1-4160-2384-5.50007-9. URL: https://www.sciencedirect.com/science/article/pii/B9781416023845500079.

Singla, D. and Z. Veqar (Apr. 2014). 'Methods of postural assessment used for sports persons'. eng. In: *Journal of clinical and diagnostic research : JCDR* 8.4. PMC4064851[pmcid], LE01–LE4. DOI: 10.7860/JCDR/2014/6836.4266. URL: https://doi.org/10.7860/JCDR/2014/6836.4266.

Singla, D., Z. Veqar and M. E. Hussain (2017). 'Photogrammetric Assessment of Upper Body Posture Using Postural Angles: A Literature Review'. In: *Journal of Chiropractic Medicine* 16.2, pp. 131–138. DOI: https://doi.org/10.1016/j.jcm.2017.01.005. URL: https://www.sciencedirect.com/science/article/pii/S1556370717300032.

TeachMeAnatomy (2020). *The Vertebral Column*. URL: https://teachmeanatomy.info/back/bones/vertebral-column/ (visited on 17/06/2021).

TNW (2021). *UX designers: Side drawer navigation could be costing you half your user engagement*. URL: https://thenextweb.com/news/ux-designers-side-drawer-navigation-costing-half-user-engagement (visited on 19/04/2021).

Tsiodoulos, D. (2016). 'Comparison of hamburger and bottom bar menu on mobile devices for three level navigation'. MA thesis. KTH, School of Computer Science and Communication (CSC).

Unity Technologies (2015). *Unity - Manual: The GameObject-Component Relationship*. URL: https://docs.unity3d.com/510/Documentation/Manual/TheGameObject-ComponentRelationship.html (visited on 26/05/2021).

Unity Technologies (2017). *Unity - Manual: GameObjects*. URL: https://docs.unity3d.com/Manual/GameObjects.html (visited on 26/05/2021).

Unity Technologies (2021a). *Augmented Reality Development Software — AR Engine for Apps — Unity*. URL: https://unity.com/unity/features/ar (visited on 23/04/2021).

Unity Technologies (2021b). *Unity - Manual: Important Classes - MonoBehaviour*. URL: https://docs.unity3d.com/Manual/class-MonoBehaviour.html (visited on 28/05/2021).

Unity Technologies (2021c). *Unity - Manual: Introduction to components*. URL: https://docs.unity3d.com/Manual/Components.html (visited on 26/05/2021).

Unity Technologies (2021d). *Unity - Scripting API: JsonUtility.ToJson*. URL: https://docs.unity3d.com/ScriptReference/JsonUtility.ToJson.html (visited on 28/05/2021).

University of Maryland (2003). *A Patient's Guide to Anatomy and Function of the Spine*. URL: https://www.umms.org/ummc/health-services/orthopedics/services/spine/patient-guides/anatomy-function (visited on 18/06/2021).

usability.gov (2021). *User Interface Elements*. URL: https://www.usability.gov/how-to-and-tools/methods/user-interface-elements.html (visited on 07/06/2021).

UX Planet (2017). *UI/UX Design Glossary. Navigation Elements*. URL: https://uxplanet.org/ui-ux-design-glossary-navigation-elements-b552130711c8 (visited on 07/06/2021).

UX Planet (2018). *Top 8 Mobile Navigation Menu Design for Your Inspiration*. URL: https://uxplanet.org/top-8-mobile-navigation-menu-design-for-your-inspiration-8a2d925bffc0 (visited on 07/06/2021).

Veritas Health (2017). *Types of Scoliosis*. URL: https://www.spine-health.com/conditions/scoliosis/types-scoliosis (visited on 18/06/2021).

Veritas Health (2021). *Anatomy of the Coccyx (Tailbone)*. URL: https://www.spine-health.com/conditions/spine-anatomy/anatomy-coccyx-tailbone (visited on 18/06/2021).

Weiniger, S. (2014). *Posture Assessment Free App - PostureZone for iPhone iPad*. URL: https://www.youtube.com/watch?v=IRQG2h6ma7c (visited on 23/06/2021).

Yorkshire Care Equipment (2020). *What causes pelvic obliquity and how can specialist seating help?* URL: https://www.yorkshirecareequipment.com/advice-tips/pelvic-obliquity-specialist-seating/ (visited on 21/06/2021).

A Appendix

A.1 Interview for Posture & Mobility Centre Staff script

A.1.1 Introduction

Introduce the project: Developing a mobile app that is designed to enhance the way clinicians can record data during postural assessments with a visualised environment.

A.1.2 1. What are the current steps in the process of recording data during a postural assessment?

To inform:

- An idea of what the full process entails
- How can the steps being fulfilled in a mobile app environment

A.1.3 2. Are there any issues with the current processes?

Give examples: lack of standard way of writing out assessments, lack of convenience, valuable time taken away by filling reports, and/or paper wastage. To inform:

- What are the deficiencies being experienced during the current process

A.1.4 3. Have any technological resolution been attempted before to resolve said issues?

To inform:

- What could work
- What to avoid

A.1.5 4. How do they feel about a mobile app being used in the workplace to supplement/replace current methods of recording postural assessments?

I will describe (in brevity) some of the features on the table for context, including:

- Guided process of taking photos what have a good perspective

- Tap-to-create anatomical marking
- Data sync so data can immediately be accessed elsewhere
- Range of devices the app can be deployed to

To inform:

- Feasibility and usefulness of planned features
- What to avoid

A.1.6 5. What would they like to see from this mobile app?

To inform:

- The balance between simplicity/speed of accessing functions and quantity of functions presented by the UI
- What features are a needs or wants (establishing priority)

A.2 Posture Assessment Data Gathering & Image Capturing Application Survey design

A.2.1 Page 1: Introduction

My name is Khalid and I am a research student at the University of South Wales, working on a project entitled 'Digital Human Model for Postural Assessment' in collaboration with the Posture & Mobility Centre. My research project is developing a mobile application that is designed to enhance the way clinicians can record data during postural assessments within a visualised environment. This is not designed to replace the postural assessment protocol, but rather to provide an easy-to-use tool for capturing the postural information quickly and reliably. The ideas on the table for this application included a guided process of capturing and annotating images, touch-to-mark anatomical marking, and gathering a patient's anthropometric measurements (e.g. location of anatomical landmarks, angle measurements).

Based on what I have already learned from meeting with PMC staff several months ago, my application is being developed on the principles of being complication-free and ensuring images captured with it are produced in a standardised way. To make sure that the application continues to meet your requirements, this survey is designed to gather your feedback on the proposed functionality, various features of the user interface and possible ways of improvements.

If you haven't done so already, please take a look at the two resources linked below for context on what the application does. The video is a live introduction and overview of using the application, and the wireframe interactable PDF is a hands-on tutorial on how to use the application.

Video: https://www.youtube.com/watch?v=a-RxMSeMz6Ifeature=youtu.be

PDF tutorial (please download to your computer and then open for it to work properly): https://drive.google.com/file/d/1RmSWCcgtnK1llLwZBfD26wZG8VE2I3wh/view?usp=sharing

A.2.2 Page 1: Start

'Start' is the first step in the data collection as part of the assessment, where you can enter relevant patient data. The information entered here will be bundled with the images and annotations sent off when the assessment is complete.

This can be seen on pages 1 to 2 on the PDF tutorial. This section focuses on feedback for the concept itself and input fields.

A.2.3 Page 2: Take & Annotation Part 1 (touch-to-mark)

The next step is 'Take', which allows you to capture a still frame image and allows you to annotate notes onto the screen. Once finished, pressing the same button again will take the app back to a live feed image and store the previous photo and its annotated ready for sending on demand. You can take multiple photos at this stage before moving onto the next.

This can be seen on pages 3 to 6 on the PDF tutorial. This section focuses on feedback for the concept itself and the annotation feature.

A.2.4 Page 3: Annotation Part 2 (templates)

The current development plan includes several predefined annotation templates that would enable quicker notetaking. Templates can be used in the 'Take' step by clicking the appropriate option in the side menu. When you add a template, several markers will appear on the screen that are linked to given annotation fields now present in the side menu - they are separated in this way to distinguish them from any custom annotations you make as described in the last section. You will also be able to drag the markers into the correct place if refinement is needed.

This can be seen on pages 7 to 10 on the PDF tutorial. This section focuses on feedback for the concept itself and possible options that should be included.

A.2.5 Page 4: Send

The final step is 'Send', where you specify what photos you wish you save and upload online from a checklist. This takes the form of a checklist with thumbnail previews of your images where you check which ones you want to keep or discard, and the ones you want to keep will be uploaded.

This can be seen on page 10 to 13 on the PDF tutorial. This section focuses on feedback for the concept itself, the selection dialogue, and any other tasks that should be complete during this stage.

A.2.6 Page 5: User Interface

As usability is a core goal of the project, this section will focuses on the user interface and experience throughout the application. As such, this is applicable to the entire video and all pages in the PDF tutorial.

A.2.7 Page 6: Summary

This final section reflects on everything else, focusing on questions that will help me prioritise the features that need advancing or are not fit-for-purpose.

A.2.8 Page 7: All done

Thank you for taking the time to fill out this survey! With this feedback, I hope to be able to better channel my focus and attention towards developing the features you will find useful and providing them in an inituitive manner.

A.2.9 Questions

I Do you believe this is easy and intuitive to use? Think about the general layout of the data fields and the buttons that operate the menu, etc.

II If not, what do you think is wrong with it and how could it be improved?

III Do you believe all relevant patient information is represented here?

IV If no, what else would you like to see?

V Do you believe this step is easy and intuitive to use? Think about the button operation and the process of tapping twice to place an annotation, etc.

VI If not, what do you think is wrong with it and how could it be improved?

VII Do you feel that the touch-to-annotate feature as-is will be useful?

VIII If no, is there anything that could be done to make it better?

IX Is there anything else you would like to see during this stage? For example, other ways to add/annotate data to the image.

X Do you believe this is a useful feature?

XI If no, what do you think is wrong or not useful about it?

XII If yes, what sort of templates do you think could be useful? Think about what areas of the body you usually need to analyse and thus should be included as an annotation automatically. Details on the number of markers wanted and compliance with the Oxford assessment form:

XIII Do you believe this step is easy and intuitive to use? Think about the menu used for selecting the images, etc.

XIV If no, what do you think is wrong with it and how could it be improved?

XV Do you feel that the checklist feature as-is is appropriate and fine on its own?

XVI If no, is there anything that could be added to make it better?

XVII Are user interface elements such as menus, text and buttons easy to understand or navigate?

XVIII If no, what elements do you believe are not easy to understand or navigate?

XIX Do you believe the colours and/or font styling of various user interface elements are appropriate (legible and easy on the eyes)?

XX If no, what elements and colours are problematic? Regardless of choice above, feel free to suggest colours and other styling tweaks too.

XXI Do you like how templated annotations (generated) and custom annotations (ones you can place by tapping the screen) are separated from each other?

XXII If no, could you describe the issue with the separation:

XXIII Overall, could you summarise how you feel about different parts of the application in its present state. This would help inform me of what features I should prioritise.

XXIV If you want to elaborate on your answer(s) above:

XXV Any other thoughts or remarks you would like to share?

A.3 FHIR implementation code

A.3.1 FHIRBundle class

```
using System;
using System.Collections.Generic;

/// <summary>
/// Implements FHIR bundle JSON structure.
/// Based on example: https://www.hl7.org/fhir/bundle.html
/// </summary>
[Serializable]
public class FHIRBundle
{
    public string resourceType = "Bundle";

    public string id = "bundle-transaction";

    public FHIRMeta meta = new FHIRMeta();

    public string type = "collection";

    public List<string> entry = new List<string>();
}
```

A.3.2 FHIRMedia class

```
using System;

/// <summary>
/// Implements FHIR media JSON structure.
/// Based on example: https://www.hl7.org/fhir/media-example.json.html
/// </summary>
[Serializable]
public class FHIRMedia
{
    public string resourceType;

    public string id;

    public FHIRText text = new FHIRText();

    public string status;

    public FHIRType type = new FHIRType();
```

```
19
20      public FHIRModality modality = new FHIRModality();
21
22      public FHIRSubject subject = new FHIRSubject();
23
24      public string createdDateTime;
25
26      public string issued;
27
28      public FHIROperator @operator = new FHIROperator();
29
30      public FHIRDevice device = new FHIRDevice();
31
32      public int height;
33
34      public int width;
35
36      public int frames;
37
38      public FHIRContent content = new FHIRContent();
39  }
```

A.3.3 FHIRAnnotation class

```
1   using System;
2
3   /// <summary>
4   /// Implements FHIR annotation JSON structure.
5   /// Based on example: https://www.hl7.org/fhir/datatypes.html#Annotation
6   /// </summary>
7   [Serializable]
8   public class FHIRAnnotation
9   {
10      public string authorString;
11
12      public string time;
13
14      public string text;
15
16      public float pointX;
17
18      public float pointY;
19  }
```

A.3.4 FHIRMeta class

```
using System;

/// <summary>
/// Implements FHIR Bundle Meta sub-structure.
/// </summary>
[Serializable]
public class FHIRMeta
{
    public string lastUpdated;
}
```

A.3.5 FHIRText class

```
using System;

/// <summary>
/// Implements FHIR Media Text sub-structure.
/// </summary>
[Serializable]
public class FHIRText
{
    public string status;

    public string div;
}
```

A.3.6 FHIRType class

```
using System;
using System.Collections.Generic;

/// <summary>
/// Implements FHIR Media Type sub-structure.
/// </summary>
[Serializable]
public class FHIRType
{
    public List<FHIRTypeCoding> coding = new List<FHIRTypeCoding>();

    public FHIRType()
    {
        coding.Add(new FHIRTypeCoding());
```

```
    }
}
```

A.3.7 FHIRTypeCoding class

```
using System;

/// <summary>
///
/// </summary>
[Serializable]
public class FHIRTypeCoding
{
    public string system = "http://terminology.hl7.org/CodeSystem/media-type"
        ↪ ;

    public string code;

    public string display;
}
```

A.3.8 FHIRModality class

```
using System;
using System.Collections.Generic;

/// <summary>
/// Implements FHIR Media Modality sub-structure.
/// </summary>
[Serializable]
public class FHIRModality
{
    public List<FHIRModalityCoding> coding = new List<FHIRModalityCoding>();

    public FHIRModality()
    {
        coding.Add(new FHIRModalityCoding());
    }
}
```

A.3.9 FHIRModalityCoding class

```
using System;
```

```
2
3  /// <summary>
4  ///
5  /// </summary>
6  [Serializable]
7  public class FHIRModalityCoding
8  {
9      public string system = "http://terminology.hl7.org/CodeSystem/media-
       ↪ modality";
10
11     public string code;
12 }
```

A.3.10 FHIRSubject class

```
1  using System;
2
3  /// <summary>
4  /// Implements FHIR Media Subject sub-structure.
5  /// </summary>
6  [Serializable]
7  public class FHIRSubject
8  {
9      public string reference;
10 }
```

A.3.11 FHIROperator class

```
1  using System;
2
3  /// <summary>
4  /// Implements FHIR Media Operator sub-structure.
5  /// </summary>
6  [Serializable]
7  public class FHIROperator
8  {
9      public string reference;
10 }
```

A.3.12 FHIRDevice class

```
1  using System;
2
```

```
/// <summary>
/// Implements FHIR Media Device sub-structure.
/// </summary>
[Serializable]
public class FHIRDevice
{
    public string display;
}
```

A.3.13 FHIRContent class

```
using System;

/// <summary>
/// Implements FHIR Media Content sub-structure.
/// </summary>
[Serializable]
public class FHIRContent
{
    public string id;

    /// <summary>
    /// The content's MIME media type.
    /// </summary>
    public string contentType;

    /// <summary>
    /// The encoded data in Base64 string.
    /// </summary>
    public string data;

    /// <summary>
    /// Simple date for this content's data.
    /// Format: YYYY-MM-DD
    /// </summary>
    public string creation;
}
```

A.4 Controllers & Handlers components code

A.4.1 AnnotationStorage class

```
using System.Collections.Generic;
using UnityEngine;
```

```
3  using Color = UnityEngine.Color;
4
5  /// <summary>
6  /// Stores all annotation placement data.
7  /// </summary>
8  public class AnnotationStorage : MonoBehaviour
9  {
10     /// <summary>
11     /// Reference to canvas object that contains
12     /// all line instances generated for displaying
13     /// annotations.
14     /// </summary>
15     [SerializeField]
16     private GameObject linesHolder;
17
18     /// <summary>
19     /// List of all stored annotations. The dictionary maps
20     /// the coordinates of the UI point to the UI annotation box.
21     /// </summary>
22     private Dictionary<Vector2, Vector2> annotes;
23
24     /// <summary>
25     /// Used for storing the UI point position whilst the program
26     /// waits for the user to give the UI annotation box position.
27     /// </summary>
28     private Vector2 temp;
29
30
31     private void Start()
32     {
33         annotes = new Dictionary<Vector2, Vector2>();
34         temp = Vector2.zero;
35     }
36
37     /// <summary>
38     /// Handles being given a annotation marker point. This method should
39     /// be called twice; one for the initial UI point to indicate where the
       ↪ annotation
40     /// points to, and one for the UI annotation box to reside at.
41     /// </summary>
42     /// <param name="pnt">2D vector of a screen space co-ordinate.</param>
43     public void GivePoint(Vector2 pnt)
44     {
45         if (temp == Vector2.zero) temp = pnt;
46         else
47         {
```

```
48              Canvas canvas = FindObjectsOfType<Canvas>()[0];
49              GameObject prefab = canvas.GetComponent<PrefabsHolder>().line;
50
51              var obj = Instantiate(
52                  prefab,
53                  prefab.GetComponent<Transform>().position,
54                  prefab.GetComponent<Transform>().rotation);
55
56              var lr = obj.GetComponent<LineRenderer>();
57              obj.transform.SetParent(linesHolder.transform, false);
58              obj.transform.SetPositionAndRotation(Vector3.zero, Quaternion.
                    ↪ identity);
59              lr.material.color = Color.blue;
60
61              lr.SetPosition(0, new Vector3(temp.x, temp.y, 15f));
62              lr.SetPosition(1, new Vector3(pnt.x, pnt.y, 15f));
63              lr.startWidth = 0.05f;
64              lr.endWidth = 0.05f;
65
66              annotes.Add(temp, pnt);
67              temp = Vector2.zero;
68          }
69      }
70
71      /// <summary>
72      /// Flags what coordinate is needed:
73      /// - false for UI point position.
74      /// - true for UI annotation box position.
75      /// </summary>
76      public bool IsNextStage()
77      {
78          if (temp != Vector2.zero)
79              return true;
80          else
81              return false;
82      }
83  }
```

A.4.2 ARCoreEnabledSceneController class

```
1   /*
2       Controller for the ARCore-enabled scene. Based on the HelloARController
3       by Google.
4    */
5
```

```
6  using GoogleARCore;
7  using GoogleARCore.Examples.Common;
8  using System.Collections.Generic;
9  using UnityEngine;
10 using UnityEngine.EventSystems;
11
12 #if UNITY_EDITOR
13 // Set up touch input propagation while using Instant Preview in the editor.
14 using Input = GoogleARCore.InstantPreviewInput;
15 #endif
16
17 /// <summary>
18 /// Operates the core functionality for the ARCoreEnabledScene
19 /// </summary>
20 public class ARCoreEnabledSceneController : MonoBehaviour
21 {
22     /// <summary>
23     /// Reference to canvas object that contains
24     /// all line instances generated for displaying
25     /// annotations.
26     /// </summary>
27     [SerializeField]
28     private GameObject linesHolder;
29
30     /// <summary>
31     /// Reference to FileDebug script.
32     /// </summary>
33     private FileDebug debug;
34
35     /// <summary>
36     /// Reference to the UIController that
37     /// is used to invoke with.
38     /// </summary>
39     protected UIController ctrl;
40
41     /// <summary>
42     /// The first-person camera being used to render the passthrough camera
       ↪ image (i.e. AR
43     /// background).
44     /// </summary>
45     [SerializeField]
46     private Camera FirstPersonCamera;
47
48     /// <summary>
49     /// The Depth Setting Menu.
50     /// </summary>
```

```
51      [SerializeField]
52      private DepthMenu depthMenu;
53
54      /// <summary>
55      /// The Instant Placement Setting Menu.
56      /// </summary
57      [SerializeField]
58      private InstantPlacementMenu instantPlacementMenu;
59
60      /// <summary>
61      /// True if the app is in the process of quitting due to an ARCore
            ↪ connection error,
62      /// otherwise false.
63      /// </summary>
64      private bool isQuitting = false;
65
66      /// <summary>
67      /// Check and update the application lifecycle.
68      /// </summary>
69      private void UpdateApplicationLifecycle()
70      {
71          // Exit the app when the 'back' button is pressed.
72          if (Input.GetKey(KeyCode.Escape))
73          {
74              Application.Quit();
75          }
76
77          // Only allow the screen to sleep when not tracking.
78          if (Session.Status != SessionStatus.Tracking)
79          {
80              Screen.sleepTimeout = SleepTimeout.SystemSetting;
81          }
82          else
83          {
84              Screen.sleepTimeout = SleepTimeout.NeverSleep;
85          }
86
87          if (isQuitting)
88          {
89              return;
90          }
91
92          // Quit if ARCore was unable to connect and give Unity some time for
                ↪ the toast to
93          // appear.
94          if (Session.Status == SessionStatus.ErrorPermissionNotGranted)
```

```
95          {
96              ShowAndroidToastMessage("Camera permission is needed to run this
                    ↪ application.");
97              isQuitting = true;
98              Invoke("DoQuit", 0.5f);
99          }
100         else if (Session.Status.IsError())
101         {
102             ShowAndroidToastMessage("ARCore encountered a problem connecting.
                    ↪ Please start the app again.");
103             isQuitting = true;
104             Invoke("DoQuit", 0.5f);
105         }
106     }
107
108     /// <summary>
109     /// Actually quit the application.
110     /// </summary>
111     private void DoQuit()
112     {
113         Application.Quit();
114     }
115
116     /// <summary>
117     /// Show an Android toast message.
118     /// </summary>
119     /// <param name="message">Message string to show in the toast.</param>
120     private void ShowAndroidToastMessage(string message)
121     {
122         AndroidJavaClass unityPlayer = new AndroidJavaClass("com.unity3d.
                ↪ player.UnityPlayer");
123         AndroidJavaObject unityActivity =
124             unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");
125
126         if (unityActivity != null)
127         {
128             AndroidJavaClass toastClass = new AndroidJavaClass("android.widget
                    ↪ .Toast");
129             unityActivity.Call("runOnUiThread", new AndroidJavaRunnable(() =>
130             {
131                 AndroidJavaObject toastObject =
132                     toastClass.CallStatic<AndroidJavaObject>(
133                         "makeText", unityActivity, message, 0);
134                 toastObject.Call("show");
135             }));
136         }
```

```
137     }
138
139     /// <summary>
140     ///
141     /// </summary>
142     /// <param name="touch"></param>
143     /// <returns></returns>
144     private List<TrackableHit> FireRaycast(Touch touch)
145     {
146         List<TrackableHit> hits = new List<TrackableHit>();
147         TrackableHitFlags raycastFilter = TrackableHitFlags.
            ↪ PlaneWithinPolygon | TrackableHitFlags.
            ↪ FeaturePointWithSurfaceNormal;
148         raycastFilter |= TrackableHitFlags.Depth;
149
150         if (Frame.RaycastAll(touch.position.x, touch.position.y,
            ↪ raycastFilter, hits))
151             return hits;
152         else
153             return null;
154     }
155
156     /// <summary>
157     /// The Unity Awake() method.
158     /// </summary>
159     public void Awake()
160     {
161         // Enable ARCore to target 60fps camera capture frame rate on
            ↪ supported devices.
162         // Note, Application.targetFrameRate is ignored when QualitySettings.
            ↪ vSyncCount != 0.
163         Application.targetFrameRate = 60;
164     }
165
166     private void Start()
167     {
168         debug = GetComponent<FileDebug>();
169         ctrl = GetComponent<UIController>();
170     }
171
172     /// <summary>
173     /// The Unity Update() method.
174     /// </summary>
175     public void Update()
176     {
177         UpdateApplicationLifecycle();
```

```
178
179         // For sampling touch input.
180         Touch touch;
181
182         // If the player has not touched the screen, we are done with this
                ↪ update.
183         if (Input.touchCount < 1 || (touch = Input.GetTouch(0)).phase !=
                ↪ TouchPhase.Began)
184             return;
185
186         // Should not handle input if the player is pointing on UI.
187         if (EventSystem.current.IsPointerOverGameObject(touch.fingerId))
188             return;
189
190         List<TrackableHit> hits = FireRaycast(touch);
191         if (depthMenu != null)
192             depthMenu.ConfigureDepthBeforePlacingFirstAsset();
193         if (hits.Count > 0)
194             ctrl.Invoke(new MarkSubjectUICmd(hits, linesHolder,
                    ↪ FirstPersonCamera));
195     }
196 }
```

A.4.3 CameraController class

```
1  using UnityEngine;
2  using UnityEngine.UI;
3
4  /// <summary>
5  /// Temporary, to provide camera output on screen
6  /// until a more advanced solution is needed when
7  /// measurement system is developed.
8  /// </summary>
9  public class CameraController : MonoBehaviour
10 {
11     /// <summary>
12     /// Canvas object to project the camera's output
13     /// on to.
14     /// </summary>
15     [SerializeField]
16     private RawImage img;
17
18     /// <summary>
19     ///
20     /// </summary>
```

```
21      [SerializeField]
22      private RectTransform imgRT;
23
24      /// <summary>
25      ///
26      /// </summary>
27      [SerializeField]
28      private AspectRatioFitter imgARF;
29
30      /// <summary>
31      ///
32      /// </summary>
33      [SerializeField]
34      private int selectedCameraIndex;
35
36      /// <summary>
37      /// Cached list of available cameras attached to this
38      /// device. Unity recommends this be cached instead of
39      /// queried when needed.
40      /// More: https://docs.unity3d.com/ScriptReference/WebCamTexture-devices.
            ↪ html
41      /// </summary>
42      private WebCamDevice[] devices;
43
44      /// <summary>
45      /// The currently-selected camera device.
46      /// </summary>
47      private WebCamDevice current;
48
49      /// <summary>
50      /// The output texture from the currently-selected
51      /// device.
52      /// </summary>
53      private WebCamTexture tex;
54
55      /// <summary>
56      /// Flags that the camera's aspect ratio should be
57      /// checked to enforce aspect ratio on the camera feed's
58      /// display iamge.
59      /// </summary>
60      private bool aspectRatioChkRequested = true;
61
62      /// <summary>
63      /// Ensures that the image is configured to
64      /// display the WebCamTexture's output properly.
65      /// </summary>
```

```
66      private void EnforceAspectRatio()
67      {
68          int rotNeeded = -tex.videoRotationAngle;
69          if (tex.videoVerticallyMirrored) rotNeeded += 180;
70
71          imgRT.localEulerAngles = new Vector3(0f, 0f, rotNeeded);
72          float videoRatio = (float)tex.width / (float)tex.height;
73          imgARF.aspectRatio = videoRatio;
74
75          if (tex.videoVerticallyMirrored)
76              img.uvRect = new Rect(1, 0, -1, 1);
77          else
78              img.uvRect = new Rect(0, 0, 1, 1);
79
80          aspectRatioChkRequested = false;
81      }
82
83      void Start()
84      {
85          devices = WebCamTexture.devices;
86
87          if (devices.Length > 0)
88          {
89              for (int i = 0; i < devices.Length; i++)
90                  Debug.Log("Found Cam #" + (i + 1).ToString() + ": " + devices[
                        ↪ i].name);
91
92              current = devices[selectedCameraIndex];
93
94              tex = new WebCamTexture();
95              tex.deviceName = current.name;
96              img.texture = tex;
97              tex.Play();
98          }
99      }
100
101     void Update()
102     {
103         if (aspectRatioChkRequested)
104         {
105             // Checks that the WebCamTexture is reporting realistic
                    ↪ information
106             // before attempting the operation. This compensates for a known
                    ↪ potential
107             // Unity bug.
108             if (tex.width > 100) EnforceAspectRatio();
```

```
109         }
110     }
111
112
113     /// <summary>
114     /// Flags if the webcam's live image is
115     /// currently playing.
116     /// </summary>
117     public bool IsPlaying() { return tex.isPlaying; }
118
119     /// <summary>
120     /// Pauses the webcam's live image.
121     /// </summary>
122     public void Pause() { tex.Pause(); }
123
124     /// <summary>
125     /// Resumes the webcam's live image.
126     /// </summary>
127     public void Resume() { tex.Play(); }
128
129     /// <summary>
130     /// Returns the available cameras attached to this device.
131     /// This method should be used instead of WebCamTexture.devices
132     /// due to the performance impact of querying devices frequently.
133     /// </summary>
134     public WebCamDevice[] GetAvailCameras() { return devices; }
135
136     /// <summary>
137     /// Changes the currently-selected camera to a different one.
138     /// </summary>
139     /// <param name="index">Index the camera is at in CameraController.
        ↪ GetAvailCameras() or WebCamTextures.devices</param>
140     /// <returns>True if change can be made, false if not (index out of range
        ↪ )</returns>
141     public bool ChangeCamera(int index)
142     {
143         if (index < devices.Length)
144         {
145             tex.Stop();
146             tex.deviceName = devices[index].name;
147             tex.Play();
148             aspectRatioChkRequested = true;
149             return true;
150         }
151
152         return false;
```

```
153     }
154 }
```

A.4.4 CanvasClickHandler class

```
1  using System.Collections.Generic;
2  using UnityEngine;
3  using UnityEngine.EventSystems;
4  using UnityEngine.UI;
5
6  /// <summary>
7  /// Handles user clicking on the main UI
8  /// canvas.
9  /// </summary>
10 public class CanvasClickHandler : UIHandler
11 {
12     /// <summary>
13     /// Reference to the main canvas to register clicks
14     /// from.
15     /// </summary>
16     [SerializeField]
17     private Canvas triggerCanvas;
18
19     [SerializeField]
20     private GameObject annotesHolder;
21
22     /// <summary>
23     /// Overridable method to flag when the SubjectFeed
24     /// within canvas has been clicked.
25     /// </summary>
26     protected virtual bool PollSubjectFeed()
27     {
28         if (Input.GetKeyDown(KeyCode.Mouse0))
29         {
30             PointerEventData pointerData = new PointerEventData(EventSystem.
                   ↪ current);
31             List<RaycastResult> hits = new List<RaycastResult>();
32             pointerData.position = Input.mousePosition;
33             triggerCanvas.GetComponent<GraphicRaycaster>().Raycast(pointerData
                   ↪ , hits);
34
35             // First pass to check if there any other UI
36             // elements in the way
37             foreach (var hit in hits)
```

```
38                if (hit.gameObject.tag == "Annotation" || hit.gameObject.tag
                      ↪ == "Menu" || hit.gameObject.tag == "InteractablePanel")
39                    return false;
40
41            // Second pass to check if the subject feed
42            // was hit
43            foreach (var hit in hits)
44                if (hit.gameObject.tag == "SubjectFeed")
45                   return true;
46        }
47        return false;
48    }
49
50    protected override void DerivedStart()
51    {
52        base.DerivedStart();
53
54        // If no trigger has been specified,
55        // prevent this code from going any further.
56        if (triggerCanvas == null && annotesHolder == null)
57            Destroy(this);
58    }
59
60    private void LateUpdate()
61    {
62        if (PollSubjectFeed() && !GetComponent<CameraController>().IsPlaying
              ↪ ())
63            ctrl.Invoke(new AnnotateSubjectUICmd(annotesHolder));
64    }
65}
```

A.4.5 CaptureScreenHandler class

```
1  using UnityEngine;
2  using UnityEngine.UI;
3
4  /// <summary>
5  /// Handles user pressing the capture
6  /// screen (Take) button.
7  /// </summary>
8  public class CaptureScreenHandler : ButtonHandler
9  {
10     /// <summary>
11     /// Reference to the button that will be
12     /// disabled when the payload command is ran.
```

```
    /// </summary>
    [SerializeField]
    private GameObject takeBtn;

    /// <summary>
    /// Reference to the button that will be enabled
    /// when the payload command is ran.
    /// </summary>
    [SerializeField]
    private GameObject sendBtn;

    /// <summary>
    ///
    /// </summary>
    [SerializeField]
    private GameObject startPanel;

    protected override void DerivedStart()
    {
        base.DerivedStart();

        // If no trigger has been specified,
        // prevent this code from going any further.
        if (triggerBtn == null)
            Destroy(this);

        // Add listener to UI button to trigger
        // UI command on press.
        triggerBtn.onClick.AddListener(() => { ctrl.Invoke(new
            ↪ CaptureScreenUICmd(takeBtn, triggerBtn.gameObject, sendBtn,
            ↪ startPanel)); });
    }
}
```

A.4.6 ExtrasMenuHandler class

```
using UnityEngine;

public class ExtrasMenuHandler : ButtonHandler
{
    [SerializeField]
    private GameObject menu;

    protected override void DerivedStart()
    {
```

```
10          base.DerivedStart();
11
12          // If no trigger has been specified,
13          // prevent this code from going any further.
14          if (triggerBtn == null)
15              Destroy(this);
16
17          // Add listener to UI button to trigger
18          // UI command on press.
19          triggerBtn.onClick.AddListener(() => { ctrl.Invoke(new
                ↪ TogglePanelUICmd(menu)); });
20      }
21  }
```

A.4.7 OpenFilePathHandler class

```
1   using UnityEngine;
2
3   public class OpenFilePathHandler : ButtonHandler
4   {
5       [SerializeField]
6       private string path;
7
8       protected override void DerivedStart()
9       {
10          base.DerivedStart();
11
12          // If no trigger or subject have been specified,
13          // prevent this code from going any further.
14          if (triggerBtn == null)
15              Destroy(this);
16
17          // Add listener to UI button to trigger
18          // UI command on press.
19          triggerBtn.onClick.AddListener(() => { ctrl.Invoke(new
                ↪ OpenFilePathUICmd(path)); });
20      }
21  }
```

A.4.8 PanelToggleHandler class

```
1   using UnityEngine;
2
3   /// <summary>
```

```
/// Handles user requesting a panel to be
/// opened and closed.
/// </summary>
public class PanelToggleHandler : ButtonHandler
{
    /// <summary>
    /// Reference to the panel to be
    /// toggled.
    /// </summary>
    [SerializeField]
    private GameObject panel;

    /// <summary>
    /// Message to display when the trigger
    /// button is pressed.
    /// </summary>
    [SerializeField]
    private string clickMessage;

    protected override void DerivedStart()
    {
        base.DerivedStart();

        // If no trigger or subject have been specified,
        // prevent this code from going any further.
        if (triggerBtn == null || panel == null)
            Destroy(this);

        // Add listener to UI button to trigger
        // UI command on press.
        triggerBtn.onClick.AddListener(() => { ctrl.Invoke(new
            ↪ TogglePanelUICmd(panel, clickMessage)); });
    }
}
```

A.4.9 PrefabsHolder class

```
using UnityEngine;

/// <summary>
/// For initialising references to prefabs
/// needed in parts of the application.
/// </summary>
public class PrefabsHolder : MonoBehaviour
{
```

```
    /// <summary>
    /// Prefab for the UI dot.
    /// </summary>
    public GameObject point;

    /// <summary>
    /// Prefab for a UI annotation box.
    /// </summary>
    public GameObject annotation;

    /// <summary>
    /// Prefab for a UI line renderer.
    /// </summary>
    public GameObject line;

    /// <summary>
    /// Prefab for a ARCore marker.
    /// </summary>
    public GameObject marker;
}
```

A.4.10 ScreensHolderHandler class

```
using System.Collections.Generic;
using TMPro;
using UnityEngine;

/// <summary>
/// Allows for captured screens to be
/// cached as later sent in batch.
/// </summary>
public class ScreensHolderHandler : UIHandler
{
    /// <summary>
    /// Reference to UI object that holds the
    /// number of screens counter.
    /// </summary>
    [SerializeField]
    private GameObject counter;

    /// <summary>
    /// List of cached JSON screen packets
    /// that need to be sent.
    /// </summary>
```

```
22      private Dictionary<string, string> jsonPackets = new Dictionary<string,
            ↪ string>();
23
24      /// <summary>
25      ///
26      /// </summary>
27      private void UpdateCounter() { ctrl.Invoke(new ScreensCounterUpdateUICmd(
            ↪ counter, jsonPackets.Count)); }
28
29      protected override void DerivedStart()
30      {
31          base.DerivedStart();
32
33          // If no UI element has been specified,
34          // prevent this code from going any further.
35          if (counter == null)
36              Destroy(this);
37      }
38
39      /// <summary>
40      /// Returns cached JSON screen packets.
41      /// </summary>
42      public Dictionary<string, string> GetScreens() { return jsonPackets; }
43
44      /// <summary>
45      /// Adds a newly-captured JSON screen packet
46      /// to cache.
47      /// </summary>
48      /// <param name="fileName">Screen filename to cache</param>
49      /// <param name="json">JSON screen packet to cache</param>
50      public void AddScreen(string fileName, string json)
51      {
52          jsonPackets.Add(fileName, json);
53          UpdateCounter();
54      }
55
56      /// <summary>
57      ///
58      /// </summary>
59      /// <param name="fileName"></param>
60      public void RemoveScreen(string fileName)
61      {
62          jsonPackets.Remove(fileName);
63          UpdateCounter();
64      }
65  }
```

A.4.11 SendScreensHandler class

```
using UnityEngine;

public class SendScreensHandler : ButtonHandler
{
    [SerializeField]
    private ShowGalleryHandler showGalleryHandler;

    [SerializeField]
    private ListGalleryController listGallery;

    [SerializeField]
    private ScreensHolderHandler screensHolderHandler;

    protected override void DerivedStart()
    {
        base.DerivedStart();

        // If no trigger has been specified,
        // prevent this code from going any further.
        if (triggerBtn == null)
            Destroy(this);

        if (showGalleryHandler == null || listGallery == null ||
            ↪ screensHolderHandler == null)
            Destroy(this);

        // Add listener to UI button to trigger
        // UI command on press.
        triggerBtn.onClick.AddListener(() => { ctrl.Invoke(new
            ↪ SendScreensUICmd(this, showGalleryHandler, listGallery,
            ↪ screensHolderHandler)); });
    }
}
```

A.4.12 ShowGalleryHandler class

```
using UnityEngine;

public class ShowGalleryHandler : ButtonHandler
{
```

```
5       [SerializeField]
6       private SendScreensHandler sendScreensHandler;
7
8       [SerializeField]
9       private ListGalleryController listGallery;
10
11      protected override void DerivedStart()
12      {
13          base.DerivedStart();
14
15          // If no trigger has been specified,
16          // prevent this code from going any further.
17          if (triggerBtn == null)
18              Destroy(this);
19
20          if (listGallery == null)
21              Destroy(this);
22
23          // Add listener to UI button to trigger
24          // UI command on press.
25          triggerBtn.onClick.AddListener(() => { ctrl.Invoke(new
                ↪ ShowGalleryUICmd(this, sendScreensHandler, listGallery)); });
26      }
27  }
```

A.4.13 StartTakeHandler class

```
1   using UnityEngine;
2
3   public class StartTakeHandler : ButtonHandler
4   {
5       [SerializeField]
6       private GameObject acceptBtn;
7
8       protected override void DerivedStart()
9       {
10          base.DerivedStart();
11
12          // If no trigger has been specified,
13          // prevent this code from going any further.
14          if (triggerBtn == null)
15              Destroy(this);
16
17          // Add listener to UI button to trigger
18          // UI command on press.
```

```
19          triggerBtn.onClick.AddListener(() => { ctrl.Invoke(new StartTakeUICmd
                ↪ (acceptBtn, triggerBtn.gameObject)); });
20      }
21  }
```

A.4.14 ToggleHander class

```
1   using UnityEngine;
2   using UnityEngine.UI;
3
4   /// <summary>
5   /// Base class for all handlers that use a
6   /// trigger toggle.
7   /// </summary>
8   public class ToggleHandler : UIHandler
9   {
10      /// <summary>
11      /// The toggle that will invoke the payload
12      /// command when pressed.
13      /// </summary>
14      [SerializeField]
15      protected Toggle triggerToggle;
16  }
```

A.4.15 UIHandler class

```
1   using UnityEngine;
2
3   /// <summary>
4   /// Handlers establish the callback between UI
5   /// elements and command invocation.
6   /// </summary>
7   public class UIHandler : MonoBehaviour
8   {
9       /// <summary>
10      /// Reference to the UIController that
11      /// is used to invoke with.
12      /// </summary>
13      protected UIController ctrl;
14
15      /// <summary>
16      /// An overridable Start method that allows
17      /// child classes to add their own items into
18      /// Start() without fear of overriding.
```

```
19      /// </summary>
20      protected virtual void DerivedStart() { }
21
22      private void Start()
23      {
24          ctrl = GameObject.FindGameObjectWithTag("AppController").GetComponent
                ↪ <UIController>();
25          DerivedStart();
26      }
27  }
```

A.4.16 UIController class

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   /// <summary>
6   /// Handles the execution of invoked UI
7   /// commands.
8   /// </summary>
9   public partial class UIController : MonoBehaviour
10  {
11      /// <summary>
12      /// Reference to FileDebug script.
13      /// </summary>
14      private FileDebug debug;
15
16      /// <summary>
17      /// Invoked commands awaiting to be executed.
18      /// Commands get queued to avoid blocking when
19      /// ran in parallel.
20      /// </summary>
21      private List<IUICmd> cmdQueue = new List<IUICmd>();
22
23      /// <summary>
24      /// Flags if a command is presently being ran.
25      /// </summary>
26      private bool cmdRunning = false;
27
28      /// <summary>
29      /// Coroutine for executing a command
30      /// in parallel.
31      /// </summary>
32      /// <returns></returns>
```

```
33      private IEnumerator ExecuteCmd(IUICmd cmd)
34      {
35          cmd.Execute(gameObject);
36          yield return new WaitForEndOfFrame();
37          cmdRunning = false;
38      }
39
40      private void Start()
41      {
42          debug = FindObjectOfType<FileDebug>();
43      }
44
45      private void Update()
46      {
47          if (!cmdRunning)
48          {
49              if (cmdQueue.Count > 0)
50              {
51                  cmdRunning = true;
52                  Debug.Log("Executing " + cmdQueue[0].GetName());
53                  debug.Log("Executing " + cmdQueue[0].GetName());
54                  StartCoroutine(ExecuteCmd(cmdQueue[0]));
55                  cmdQueue.RemoveAt(0);
56              }
57          }
58      }
59
60      /// <summary>
61      /// Allows UIHandlers to invoke commands.
62      /// </summary>
63      /// <param name="cmd">Command to invoke</param>
64      public void Invoke(IUICmd cmd)
65      {
66          cmdQueue.Add(cmd);
67      }
68  }
```

A.5 Real-time measurement components code

A.5.1 RankedPlane class

```
1  using GoogleARCore;
2  using System.Collections.Generic;
3
```

```
/// <summary>
/// Struct binding a ARCore detected plane structure
/// to a number of point cloud points that form it.
/// </summary>
public class RankedPlane
{
    /// <summary>
    /// ID of this RankedPlane.
    /// </summary>
    public int ID { get; set; }

    /// <summary>
    /// A given detected plane.
    /// </summary>
    public DetectedPlane Plane { get; set; }

    /// <summary>
    /// The IDs of features that could form this plane.
    /// </summary>
    public List<int> FeatureIDs { get; set; }

    /// <summary>
    /// Number of features that could form this plane.
    /// </summary>
    public int FeatureCount { get => FeatureIDs.Count; }

    /// <summary>
    /// Total confidence within this plane.
    /// </summary>
    public float TotalConfidence { get; set; }

    /// <summary>
    /// Average confidence for this plane.
    /// </summary>
    public float AvgConfidence { get => TotalConfidence / FeatureCount; }
}
```

A.5.2 PlaneRanker class

```
using GoogleARCore;
using System.Collections.Generic;
using System.Linq;
using TMPro;
using UnityEngine;

```

```
7  /// <summary>
8  /// Retrieves planes from ARcore session and binds them to a
9  /// data structure that can also hold variables than can be used
10 /// to assess their quality.
11 /// </summary>
12 public class PlaneRanker : MonoBehaviour
13 {
14     /// <summary>
15     /// Reference to FileDebug script.
16     /// </summary>
17     private FileDebug debug;
18
19     /// <summary>
20     /// UI element to display the number of point clouds
21     /// on.
22     /// </summary>
23     [SerializeField]
24     private TextMeshProUGUI pointCloudCountUI;
25
26     /// <summary>
27     /// Threshold for flagging if a point forms a given plane.
28     /// </summary>
29     [SerializeField]
30     private float pointPlaneThres = 1f;
31
32     /// <summary>
33     /// List of DetectedPlanes bundled in a structure that can
34     /// accept variables to 'rank' their quality.
35     /// </summary>
36     private List<RankedPlane> rankedPlanes = new List<RankedPlane>();
37
38     /// <summary>
39     /// Refreshes list of planes.
40     /// </summary>
41     private void RefreshRankedPlanes()
42     {
43         // Get the 'raw' detected planes from ARCore's session.
44         List<DetectedPlane> planes = new List<DetectedPlane>();
45         Session.GetTrackables(planes, TrackableQueryFilter.New);
46
47         // Initialise a RankedPlane for each DetectedPlane and
48         // give it an ID.
49         for (var i = 0; i < planes.Count; i++)
50         {
51             var nPlane = new RankedPlane();
52             nPlane.ID = rankedPlanes.Count + 1;
```

```
53              nPlane.Plane = planes[i];
54              nPlane.FeatureIDs = new List<int>();
55              nPlane.TotalConfidence = 0f;
56
57              rankedPlanes.Add(nPlane);
58          }
59      }
60
61      /// <summary>
62      /// 'Ranks' all the planes in RankedPlanes by checking how
63      /// many feature points lie within plane bounds and computing
64      /// total/average confidence for the plane.
65      /// </summary>
66      private void RankRankedPlanes()
67      {
68          foreach (var plane in rankedPlanes)
69          {
70              // Plane's position
71              var planePos = plane.Plane.CenterPose.position;
72              // Half-extent of plane's X-axis
73              var planeExtXH = plane.Plane.ExtentX / 2;
74              // Half-extent of plane's Y-axis
75              var planeExtZH = plane.Plane.ExtentZ / 2;
76
77              for (var i = 0; i < Frame.PointCloud.PointCount; i++)
78              {
79                  var point = Frame.PointCloud.GetPointAsStruct(i);
80
81                  if (point.Id != -1) // Only proceed if point is flagged valid
                        ↪ (invalid == -1)
82                  {
83                      // Point's position
84                      var pointPos = point.Position;
85                      // Difference between point and plane's Y-coords
86                      var diffY = Mathf.Abs(pointPos.y - planePos.y);
87
88                      // Check if current point is simply close to the plane's
                            ↪ center
89                      if (Vector3.Distance(pointPos, planePos) <= pointPlaneThres
                            ↪ )
90                      {
91                          if (!plane.FeatureIDs.Contains(point.Id))
92                          {
93                              plane.FeatureIDs.Add(point.Id);
94                              plane.TotalConfidence += point.Confidence;
95                          }
```

```
96
97                  }
98                  // Check if current point is within the extents of a plane
99                  else if (diffY <= pointPlaneThres)
100                 {
101                     if ((pointPos.x >= (planePos.x - planeExtXH) &&
                            ↪ pointPos.x <= (planePos.x + planeExtXH))
102                         && pointPos.z >= (planePos.z - planeExtZH) &&
                                ↪ pointPos.z <= (planePos.z + planeExtZH))
103                     {
104                         if (!plane.FeatureIDs.Contains(point.Id))
105                         {
106                             plane.FeatureIDs.Add(point.Id);
107                             plane.TotalConfidence += point.Confidence;
108                         }
109                     }
110                 }
111             }
112         }
113     }
114
115     rankedPlanes.Sort((x, y) => x.AvgConfidence.CompareTo(y.AvgConfidence
            ↪ ));
116 }
117
118 private void Start()
119 {
120     debug = FindObjectOfType<FileDebug>();
121 }
122
123 private void Update()
124 {
125     // Check that motion tracking is tracking.
126     if (Session.Status != SessionStatus.Tracking)
127         return;
128
129     if (Frame.PointCloud.IsUpdatedThisFrame)
130     {
131         if (pointCloudCountUI)
132             pointCloudCountUI.text = Frame.PointCloud.PointCount.ToString
                    ↪ ();
133
134         RefreshRankedPlanes();
135         RankRankedPlanes();
136     }
137 }
```

```
138
139     /// <summary>
140     /// Returns a list of DetectedPlanes ranked by their relation
141     /// to established feature points.
142     /// </summary>
143     public List<RankedPlane> GetRankedPlanes() { return rankedPlanes; }
144 }
```

A.5.3 PlaneGenerator class

```
1  using System.Collections.Generic;
2  using GoogleARCore;
3  using GoogleARCore.Examples.Common;
4  using UnityEngine;
5
6  /// <summary>
7  /// Manages the visualisation of planes. This is based on the
8  /// DetectedPlaneGenerator class example included with the ARcore
9  /// Unity SDK.
10 /// </summary>
11 public class PlaneGenerator : MonoBehaviour
12 {
13     /// <summary>
14     /// Reference to FileDebug script.
15     /// </summary>
16     private FileDebug debug;
17
18     /// <summary>
19     /// A prefab for tracking and visualizing detected planes.
20     /// </summary>
21     public GameObject DetectedPlanePrefab;
22
23     /// <summary>
24     /// A list to hold new planes ARCore began tracking in the current frame.
25     /// </summary>
26     private List<DetectedPlane> _newPlanes = new List<DetectedPlane>();
27
28     /// <summary>
29     /// List of initialised planes as GameObjects binded with their center
       ↪ pose
30     /// vector.
31     /// </summary>
32     private Dictionary<Vector3, GameObject> initPlanes = new Dictionary<
       ↪ Vector3, GameObject>();
33
```

```
34      /// <summary>
35      /// Reference to the plane ranker object.
36      /// </summary>
37      [SerializeField]
38      private PlaneRanker planes;
39
40      /// <summary>
41      /// Threshold for flagging if a plane has been recalculated
42      /// and moved too much to count as the same plane.
43      /// </summary>
44      [SerializeField]
45      private float centerDriftThres = 0.75f;
46
47      /// <summary>
48      /// Creates a visual representation of the ranking of planes. The closer
49      /// to green a plane is, the more reliable a plane is judged to be.
50      /// </summary>
51      private void ColourPlanes()
52      {
53          // Retrieve pre-'ranked' planes from the plane ranker.
54          List<RankedPlane> rankedPlanes = planes.GetRankedPlanes();
55
56          if (rankedPlanes.Count > 0)
57          {
58              // Colour increment size difference between reach
59              // plane's rank colour.
60              float col_comp = 255f / rankedPlanes.Count;
61
62              for (int i = 0; i < rankedPlanes.Count; i++)
63              {
64                  // Check if ranked plane has an initialised
65                  // GameObject counterpart.
66                  foreach (KeyValuePair<Vector3, GameObject> j in initPlanes)
67                  {
68                      if (Vector3.Distance(j.Key, rankedPlanes[i].Plane.
                            ↪ CenterPose.position) <= centerDriftThres)
69                      {
70                          // Calculate the allocation of colour this plane should
                                ↪  have
71                          float col = Mathf.Clamp(col_comp * i * 2, 0, 255);
72                          //debug.Log(col.ToString());
73                          // Set the plane's colour. _GridColor is the variable
                                ↪ the
74                          // plane's shader uses.
75                          j.Value.GetComponent<MeshRenderer>().material.SetColor(
                                ↪ "_GridColor", new Color(1 - (col / 255), col /
```

```
                        ↪ 255, 0));
76
77                      break;
78                  }
79              }
80          }
81      }
82  }
83
84  private void Start()
85  {
86      debug = FindObjectOfType<FileDebug>();
87  }
88
89  private void Update()
90  {
91      // Check that motion tracking is tracking.
92      if (Session.Status != SessionStatus.Tracking)
93          return;
94
95      // Iterate over planes found in this frame and instantiate
            ↪ corresponding GameObjects to
96      // visualize them.
97      Session.GetTrackables(_newPlanes, TrackableQueryFilter.New);
98      for (int i = 0; i < _newPlanes.Count; i++)
99      {
100         // Instantiate a plane visualization prefab and set it to track
                ↪ the new plane. The
101         // transform is set to the origin with an identity rotation since
                ↪ the mesh for our
102         // prefab is updated in Unity World coordinates.
103         GameObject planeObject =
104             Instantiate(DetectedPlanePrefab, Vector3.zero, Quaternion.
                    ↪ identity, transform);
105         planeObject.GetComponent<DetectedPlaneVisualizer>().Initialize(
                ↪ _newPlanes[i]);
106         initPlanes.Add(_newPlanes[i].CenterPose.position, planeObject);
107     }
108
109     ColourPlanes();
110 }
111 }
```

A.6 UI command implementations code

A.6.1 IUICmd interface

```
using UnityEngine;

/// <summary>
/// Default interface for all UI commands.
/// </summary>
public interface IUICmd
{
    /// <summary>
    /// Runs the contents of this command.
    /// </summary>
    /// <param name="context">Allows command to access other entities in the
        ↪ scene</param>
    void Execute(GameObject context);

    /// <summary>
    /// Returns a name of this command for debugging
    /// use.
    /// </summary>
    string GetName();
}
```

A.6.2 AnnotateSubjectUICmd class

```
using System;
using TMPro;
using UnityEngine;

/// <summary>
/// Command for placing an element for an annotation.
/// It will make a UI point or annotation box depending
/// the status reported by an AnnotationStorage object .
/// </summary>
public class AnnotateSubjectUICmd : IUICmd
{
    /// <summary>
    /// Canvas that the annotations should be
    /// made to.
    /// </summary>
    protected Canvas canvas;

    /// <summary>
```

```
19      /// Prefabricated object of a UI point.
20      /// </summary>
21      protected GameObject pointPrefab;
22
23      /// <summary>
24      /// Prefabricated object of a UI annotation box.
25      /// </summary>
26      protected GameObject annotationPrefab;
27
28      /// <summary>
29      /// Reference to the annotation storage medium.
30      /// </summary>
31      protected AnnotationStorage annotes;
32
33      /// <summary>
34      /// Reference to UI element that holds all the annotations.
35      /// </summary>
36      protected GameObject annotesHold;
37
38      /// <summary>
39      /// Places an prefab of annotation point or box where the mouse pointer
            ↪ is
40      /// currently at.
41      /// </summary>
42      protected virtual GameObject PlacePrefab(GameObject prefab)
43      {
44          // Create an instance of an annotation
45          var obj = UnityEngine.Object.Instantiate(prefab, prefab.GetComponent<
                ↪ RectTransform>().position, prefab.GetComponent<RectTransform>()
                ↪ .rotation);
46          obj.transform.position += new Vector3(0, 0, -20);
47          obj.transform.SetParent(canvas.transform, false);
48          var rectT = obj.GetComponent<RectTransform>();
49
50          // Set annotation's location to mouse position
51          // Important to translate the value from screen space
52          // to local within the rectangle's space.
53          Vector2 anchorPos;
54          RectTransformUtility.ScreenPointToLocalPointInRectangle(rectT, Input.
                ↪ mousePosition, Camera.main, out anchorPos);
55          rectT.anchoredPosition = anchorPos;
56          annotes.GivePoint(rectT.TransformPoint(rectT.rect.center));
57          obj.transform.SetParent(annotesHold.transform);
58
59          return obj;
60      }
```

```
61
62      public AnnotateSubjectUICmd(GameObject annotesHolder)
63      {
64          annotesHold = annotesHolder;
65      }
66
67      public void Execute(GameObject context)
68      {
69          canvas = UnityEngine.Object.FindObjectsOfType<Canvas>()[0];
70          pointPrefab = canvas.GetComponent<PrefabsHolder>().point;
71          annotationPrefab = canvas.GetComponent<PrefabsHolder>().annotation;
72          annotes = UnityEngine.Object.FindObjectsOfType<AnnotationStorage>()
                ↪ [0];
73
74          if (!annotes.IsNextStage())
75              PlacePrefab(pointPrefab);
76          else
77          {
78              var obj = PlacePrefab(annotationPrefab);
79              // Set the date value inside the annotation
80              obj.transform.GetChild((int)AnnotationComponents.Date).
                    ↪ GetComponent<TextMeshProUGUI>().text = DateTime.Now.
                    ↪ ToString("yyyy-MM-dd HH-mm-ss");
81          }
82      }
83
84      public string GetName() { return "Annotate Subject Command (mouse variant
            ↪ )"; }
85  }
```

A.6.3 CaptureScreenUICmd class

```
1   using System;
2   using System.Collections.Generic;
3   using System.IO;
4   using TMPro;
5   using UnityEngine;
6   using UnityEngine.UI;
7
8   /// <summary>
9   /// Command for capturing the screen and saves it as a JSON packet.
10  /// </summary>
11  public class CaptureScreenUICmd : IUICmd
12  {
13      /// <summary>
```

```
14      /// Reference to the take button.
15      /// </summary>
16      protected GameObject take;
17
18      /// <summary>
19      /// Reference to the accept button.
20      /// </summary>
21      protected GameObject accept;
22
23      /// <summary>
24      /// Reference to the send button.
25      /// </summary>
26      protected GameObject send;
27
28      /// <summary>
29      ///
30      /// </summary>
31      protected GameObject start;
32
33      /// <summary>
34      /// Access to main UI canvas.
35      /// </summary>
36      protected Canvas canvas;
37
38      /// <summary>
39      /// File name of the saved screen.
40      /// </summary>
41      protected string fileName = "";
42
43      /// <summary>
44      /// The saved screen as a Unity texture.
45      /// </summary>
46      protected Texture2D shot;
47
48      /// <summary>
49      /// The saved screen as bytes.
50      /// </summary>
51      protected byte[] imgRaw;
52
53      /// <summary>
54      /// Assembled JSON packet as string.
55      /// </summary>
56      protected string json;
57
58      /// <summary>
59      /// Captures the main camera's output as a texture.
```

```
60      /// </summary>
61      private void CaptureScreen()
62      {
63          // Disable the menu interface.
64          canvas.transform.GetChild(1).gameObject.SetActive(false);
65
66          Camera cam = GameObject.FindGameObjectWithTag("MainCamera").
                ↪ GetComponent<Camera>();
67          int width = cam.pixelWidth;
68          int height = cam.pixelHeight;
69
70          RenderTexture rt = new RenderTexture(width, height, 24);
71          cam.targetTexture = rt;
72
73          shot = new Texture2D(width, height, TextureFormat.RGB24, false);
74          cam.Render();
75          RenderTexture.active = rt;
76          shot.ReadPixels(new Rect(0, 0, width, height), 0, 0);
77
78          cam.targetTexture = null;
79          RenderTexture.active = null;
80          UnityEngine.Object.Destroy(rt);
81
82          // Reenable the menu interface.
83          canvas.transform.GetChild(1).gameObject.SetActive(true);
84      }
85
86      /// <summary>
87      /// Ensures that the required directories for storing
88      /// locally-stored/backup files are created.
89      /// </summary>
90      protected virtual void CreateDirs()
91      {
92          if (!Directory.Exists(Application.dataPath + "/../Captures/"))
93              Directory.CreateDirectory(Application.dataPath + "/../Captures/");
94
95          if (!Directory.Exists(Application.dataPath + "/../JSON/"))
96              Directory.CreateDirectory(Application.dataPath + "/../JSON/");
97      }
98
99      /// <summary>
100     /// Saves a given texture in the file system. Virtual
101     /// so that this class can be inherented and modified with
102     /// different SaveScreen code for different platforms.
103     /// </summary>
104     protected virtual void SaveScreen()
```

```
105     {
106         fileName = string.Format("{0}/../Captures/screen_{1}.jpeg",
                ↪ Application.dataPath, DateTime.Now.ToString("yyyy-MM-dd_HH-mm-
                ↪ ss"));
107 
108         imgRaw = shot.EncodeToJPG(75);
109         File.WriteAllBytes(fileName, imgRaw);
110     }
111 
112     /// <summary>
113     /// Serialises the image and annotation data into a
114     /// FHIR bundle format in JSON.
115     /// </summary>
116     protected void BuildJSON()
117     {
118         string patientID = start.transform.GetChild(1).GetChild(0).GetChild
                ↪ (0).GetComponent<TMP_InputField>().text;
119         string patientFirstName = start.transform.GetChild(1).GetChild(0).
                ↪ GetChild(1).GetComponent<TMP_InputField>().text;
120         string patientLastName = start.transform.GetChild(1).GetChild(0).
                ↪ GetChild(2).GetComponent<TMP_InputField>().text;
121         string patientDOB = start.transform.GetChild(1).GetChild(0).GetChild
                ↪ (3).GetComponent<TMP_InputField>().text;
122         string patientGender = start.transform.GetChild(1).GetChild(0).
                ↪ GetChild(4).GetComponent<TMP_InputField>().text;
123         string patientAddress = start.transform.GetChild(1).GetChild(0).
                ↪ GetChild(5).GetComponent<TMP_InputField>().text;
124 
125         FHIRBundle bundle = new FHIRBundle();
126 
127         FHIRPatient patient = new FHIRPatient();
128         patient.identifier = patientID;
129         patient.name = patientFirstName + " " + patientLastName;
130         patient.gender = patientGender;
131         patient.birthDate = patientDOB;
132         patient.address = patientAddress;
133 
134         bundle.entry.Add(JsonUtility.ToJson(patient));
135 
136         FHIRMedia img = new FHIRMedia();
137         img.resourceType = "Media";
138         img.id = UnityEngine.Random.Range(int.MinValue, int.MaxValue).
                ↪ ToString();
139         img.text.status = "generated";
140         img.text.div = "<div xmlns=\"http://www.w3.org/1999/xhtml\">Photo of
                ↪ " + patientFirstName + " " + patientLastName + "</div>";
```

```
141         img.status = "completed";
142         img.type.coding[0].code = "photo";
143         img.type.coding[0].display = "Photo";
144         img.modality.coding[0].code = "diagram";
145         img.subject.reference = "Patient/xcda";
146         img.createdDateTime = DateTime.Now.ToString("yyyy-MM-dd");
147         img.issued = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
148         img.@operator.reference = "Practitioner/xcda-author";
149         img.device.display = WebCamTexture.devices[0].name;
150         img.height = Screen.height;
151         img.width = Screen.width;
152         img.frames = 1;
153         img.content.id = fileName;
154         img.content.contentType = "image/jpeg";
155         img.content.data = Convert.ToBase64String(imgRaw);
156         img.content.creation = DateTime.Now.ToString("yyyy-MM-dd");
157 
158         bundle.entry.Add(JsonUtility.ToJson(img));
159 
160     GameObject[] annotesUI = GameObject.FindGameObjectsWithTag("Annotation");
161         foreach (GameObject annoteUI in annotesUI)
162         {
163             FHIRAnnotation annote = new FHIRAnnotation();
164             annote.time = annoteUI.transform.GetChild((int)
                    ↪ AnnotationComponents.Date).GetComponent<TextMeshProUGUI>().
                    ↪ text;
165             annote.text = annoteUI.transform.GetChild((int)
                    ↪ AnnotationComponents.Text).GetChild(0).GetChild(1).
                    ↪ GetComponent<TextMeshProUGUI>().text;
166             annote.pointX = annoteUI.transform.position.x;
167             annote.pointY = annoteUI.transform.position.y;
168 
169             bundle.entry.Add(JsonUtility.ToJson(annote));
170         }
171 
172         bundle.meta.lastUpdated = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:
                ↪ ssZ");
173 
174         json = JsonUtility.ToJson(bundle, true);
175         string jsonFile = string.Format("{0}/../JSON/post_{1}.json",
                ↪ Application.dataPath, DateTime.Now.ToString("yyyy-MM-dd_HH-mm-
                ↪ ss"));
176         File.WriteAllText(jsonFile, json);
177     }
178 
179     /// <summary>
```

```
180     /// Cleans up all the annotations on screen.
181     /// </summary>
182     private void ClearAnnotations()
183     {
184         GameObject[] annotes = GameObject.FindGameObjectsWithTag("Annotation"
                ↪ );
185         foreach (GameObject annote in annotes)
186             UnityEngine.Object.Destroy(annote);
187 
188         GameObject[] points = GameObject.FindGameObjectsWithTag("Point");
189         foreach (GameObject point in points)
190             UnityEngine.Object.Destroy(point);
191 
192         GameObject[] lines = GameObject.FindGameObjectsWithTag("
                ↪ AnnotationLine");
193         foreach (GameObject line in lines)
194             UnityEngine.Object.Destroy(line);
195     }
196 
197     public CaptureScreenUICmd(GameObject takeBtn, GameObject acceptBtn,
            ↪ GameObject sendBtn, GameObject startPanel)
198     {
199         take = takeBtn;
200         accept = acceptBtn;
201         send = sendBtn;
202         start = startPanel;
203     }
204 
205     public void Execute(GameObject context)
206     {
207         canvas = UnityEngine.Object.FindObjectsOfType<Canvas>()[0];
208 
209         CreateDirs();
210         CaptureScreen();
211         SaveScreen();
212         ClearAnnotations();
213         BuildJSON();
214 
215         UnityEngine.Object.FindObjectsOfType<ScreensHolderHandler>()[0].
                ↪ AddScreen(fileName, json);
216 
217         UnityEngine.Object.FindObjectsOfType<CameraController>()[0].Resume();
218         take.SetActive(true);
219         accept.SetActive(false);
220         send.GetComponent<Button>().interactable = true;
221     }
```

```
222
223     public string GetName() { return "Capture Screen Command (Windows variant
            ↪ )"; }
224 }
```

A.6.4 ChangeSceneUICmd class

```
1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3
4  /// <summary>
5  /// Command for changing the app's
6  /// scene.
7  /// </summary>
8  class ChangeSceneUICmd : IUICmd
9  {
10     /// <summary>
11     /// Index of the scene to change to.
12     /// </summary>
13     private int scene;
14
15     public ChangeSceneUICmd(int sceneIndex)
16     {
17         scene = sceneIndex;
18     }
19
20     public void Execute(GameObject context)
21     {
22         SceneManager.LoadScene(scene);
23     }
24
25     public string GetName() { return "Change Scene Command"; }
26 }
```

A.6.5 MarkSubjectUICmd class

```
1  using GoogleARCore;
2  using GoogleARCore.Examples.Common;
3  using System;
4  using System.Collections.Generic;
5  using UnityEngine;
6
7  /// <summary>
8  /// Command for placing an element for an marker measurement.
```

```
9  /// </summary>
10 public class MarkSubjectUICmd : IUICmd
11 {
12     /// <summary>
13     /// Reference to FileDebug script.
14     /// </summary>
15     private FileDebug debug;
16
17     /// <summary>
18     /// Canvas that the annotations should be
19     /// made to.
20     /// </summary>
21     protected Canvas canvas;
22
23     /// <summary>
24     /// Prefabricated object of a marker.
25     /// </summary>
26     protected GameObject markerPrefab;
27
28     /// <summary>
29     /// Prefabricated object of a UI line.
30     /// </summary>
31     protected GameObject linePrefab;
32
33     /// <summary>
34     /// Reference to canvas object that contains
35     /// all line instances generated for displaying
36     /// annotations.
37     /// </summary>
38     [SerializeField]
39     private GameObject linesHold;
40
41     /// <summary>
42     /// List of all hitsList received.
43     /// </summary>
44     protected List<TrackableHit> hitsList;
45
46     /// <summary>
47     ///
48     /// </summary>
49     protected Camera cam;
50
51     /// <summary>
52     /// Reference to the app's notification controller
53     /// script.
54     /// </summary>
```

```
55      protected NotificationsController notesCtrl;
56
57      /// <summary>
58      /// Places a marker where the user touch the screen.
59      /// </summary>
60      protected virtual void PlaceMarker()
61      {
62          //int selected = hitsList.Count - 1;
63
64          int selected = int.MaxValue;
65          float extent = 0;
66
67          if (hitsList.Count == 1)
68          {
69              if (hitsList[0].Trackable is DetectedPlane)
70              {
71                  selected = 0;
72                  DetectedPlane temp = hitsList[0].Trackable as DetectedPlane;
73                  extent = temp.ExtentX + temp.ExtentZ;
74              }
75          }
76          else
77          {
78              // Select the largest plane.
79              for (int i = 0; i < hitsList.Count; i++)
80              {
81                  if (hitsList[i].Trackable is DetectedPlane)
82                  {
83                      if (selected == int.MaxValue)
84                      {
85                          selected = i;
86                          DetectedPlane temp = hitsList[i].Trackable as
                                ↪ DetectedPlane;
87                          extent = temp.ExtentX + temp.ExtentZ;
88                      }
89                      else
90                      {
91                          DetectedPlane temp = hitsList[i].Trackable as
                                ↪ DetectedPlane;
92                          if (temp.ExtentX + temp.ExtentZ > extent)
93                          {
94                              selected = i;
95                              extent = temp.ExtentX + temp.ExtentZ;
96                          }
97                      }
98                  }
```

```
99              }
100
101             // Catch if above checks fail.
102             if (selected == int.MaxValue)
103                 selected = 0;
104         }
105
106         debug.Log("Selected plane index: " + selected.ToString());
107
108         if ((hitsList[selected].Trackable is DetectedPlane) && Vector3.Dot(
                ↪ cam.transform.position - hitsList[selected].Pose.position,
                ↪ hitsList[selected].Pose.rotation * Vector3.up) < 0)
109         {
110             debug.Log("Hit at back of the current DetectedPlane.");
111         }
112         else
113         {
114             // Instantiate prefab at the hit pose.
115             var obj = UnityEngine.Object.Instantiate(markerPrefab, hitsList[
                    ↪ selected].Pose.position, hitsList[selected].Pose.rotation);
116
117             // Compensate for the hitPose rotation facing away from the
                    ↪ raycast (i.e. camera).
118             obj.transform.Rotate(0, 180, 0, Space.Self);
119
120             // Create an anchor to allow ARCore to track the hitpoint as
                    ↪ understanding of
121             // the physical world evolves.
122             var anchor = hitsList[selected].Trackable.CreateAnchor(hitsList[
                    ↪ selected].Pose);
123
124             // Make game object a child of the anchor.
125             obj.transform.parent = anchor.transform;
126
127             // Initialize Instant Placement Effect.
128             if (hitsList[selected].Trackable is InstantPlacementPoint)
129             {
130                 obj.GetComponentInChildren<InstantPlacementEffect>().
                        ↪ InitializeWithTrackable(hitsList[selected].Trackable);
131             }
132
133             debug.Log("Placed: " + obj.transform.name);
134
135             GameObject[] markers = GameObject.FindGameObjectsWithTag("Marker")
                    ↪ ;
136             // Make measurement line between markers
```

```
137             if (markers.Length > 1)
138             {
139                 Transform otherMarker = markers[0].transform;
140 
141                 float dist = Vector3.Distance(otherMarker.position, obj.
                        ↪ transform.position) * 100;
142 
143                 if (notesCtrl)
144                     notesCtrl.QueueNote("Measurement: " + Math.Round(dist, 2).
                            ↪ ToString() + "cm", 10, true);
145 
146                 GameObject line = UnityEngine.Object.Instantiate(
147                     linePrefab,
148                     linePrefab.GetComponent<Transform>().position,
149                     linePrefab.GetComponent<Transform>().rotation);
150 
151                 var lr = line.GetComponent<LineRenderer>();
152                 var lineAnchor = Session.CreateAnchor(new Pose((otherMarker.
                        ↪ position + obj.transform.position) / 2, obj.transform.
                        ↪ rotation));
153                 lr.transform.SetParent(lineAnchor.transform, true);
154                 lr.transform.SetPositionAndRotation((otherMarker.position +
                        ↪ obj.transform.position) / 2, obj.transform.rotation);
155                 lr.material.color = Color.blue;
156 
157                 lr.SetPosition(0, otherMarker.position);
158                 lr.SetPosition(1, obj.transform.position);
159                 lr.startWidth = 0.01f;
160                 lr.endWidth = 0.01f;
161             }
162 
163         }
164     }
165 
166     public MarkSubjectUICmd(List<TrackableHit> hits, GameObject linesHolder,
            ↪ Camera firstPersonCamera)
167     {
168         debug = UnityEngine.Object.FindObjectOfType<FileDebug>();
169         canvas = UnityEngine.Object.FindObjectsOfType<Canvas>()[0];
170         markerPrefab = canvas.GetComponent<PrefabsHolder>().marker;
171         linePrefab = canvas.GetComponent<PrefabsHolder>().line;
172         linesHold = linesHolder;
173         cam = firstPersonCamera;
174         hitsList = hits;
175         notesCtrl = UnityEngine.Object.FindObjectOfType<
                ↪ NotificationsController>();
```

```
176     }
177
178     public void Execute(GameObject context)
179     {
180         PlaceMarker();
181     }
182
183     public string GetName() { return "Mark Subject Command"; }
184 }
```

A.6.6 OpenFilePathUICmd class

```
1  using UnityEngine;
2
3  class OpenFilePathUICmd : IUICmd
4  {
5      private string relativePath;
6
7      public OpenFilePathUICmd(string path)
8      {
9          relativePath = path;
10     }
11
12     public void Execute(GameObject context)
13     {
14         Application.OpenURL(Application.persistentDataPath + relativePath);
15     }
16
17     public string GetName() { return "Open File Path Command"; }
18 }
```

A.6.7 ScreensCounterUpdateUICmd class

```
1  using TMPro;
2  using UnityEngine;
3
4  /// <summary>
5  /// Command for updating the counter showing
6  /// how many screens are awaiting sending.
7  /// </summary>
8  class ScreensCounterUpdateUICmd : IUICmd
9  {
10     /// <summary>
11     /// Reference to the text element that
```

```
12      /// the counter updates.
13      /// </summary>
14      private GameObject textEle;
15
16      /// <summary>
17      /// Current counter value.
18      /// </summary>
19      private int counter;
20
21      public ScreensCounterUpdateUICmd(GameObject obj, int val)
22      {
23          textEle = obj;
24          counter = val;
25      }
26
27      public void Execute(GameObject context)
28      {
29          if (counter == 0)
30              textEle.SetActive(false);
31          else
32              textEle.SetActive(true);
33
34          textEle.transform.GetChild(0).GetComponent<TextMeshProUGUI>().text =
                ↪ counter.ToString();
35      }
36
37      public string GetName() { return "Screens Counter Update Command"; }
38  }
```

A.6.8 SendScreensUICmd class

```
1   using System.Collections.Generic;
2   using System.Net.Http;
3   using UnityEngine;
4
5   /// <summary>
6   /// Command for sending held screens to
7   /// a desired remote host.
8   /// </summary>
9   public class SendScreensUICmd : IUICmd
10  {
11      /// <summary>
12      /// Reference to the object storing the
13      /// awaiting-for-sending screens.
14      /// </summary>
```

```
15      private ScreensHolderHandler shH;
16
17      /// <summary>
18      /// Reference to the gallery selection.
19      /// </summary>
20      private ListGalleryController listGallery;
21
22      /// <summary>
23      /// List of screens as JSON to send.
24      /// </summary>
25      protected Dictionary<string, string> screens;
26
27      /// <summary>
28      /// HTTP client used for making all connections.
29      /// </summary>
30      protected HttpClient reqClient = new HttpClient();
31
32      /// <summary>
33      /// Web server to send POST requests
34      /// to.
35      /// </summary>
36      protected string reqURI = "http://139.59.174.78/upload.php";
37
38      /// <summary>
39      /// Reference to the app's notification controller
40      /// script.
41      /// </summary>
42      protected NotificationsController notesCtrl;
43
44
45      /// <summary>
46      /// Sends the list of screens' JSON data
47      /// to a specified web server using HTTP POST.
48      /// </summary>
49      protected virtual void Send()
50      {
51          int i = 1;
52          foreach (var screen in screens)
53          {
54              Debug.Log("Sending Screen " + i.ToString() + "/" + screens.Count.
                    ↪ ToString());
55
56              var content = new MultipartFormDataContent();
57
58              HttpContent valContent = new StringContent(screen.Value);
59              content.Add(valContent, "JSON");
```

```
60
61              var resp = reqClient.PostAsync(reqURI, content).GetAwaiter().
                    ↪ GetResult();
62
63              Debug.Log("POST Response Code: " + resp.StatusCode.ToString());
64              Debug.Log("POST Response Str: " + resp.Content.ReadAsStringAsync()
                    ↪ .GetAwaiter().GetResult().ToString());
65
66              if (resp.StatusCode.ToString() == "OK")
67                  if (notesCtrl)
68                      notesCtrl.QueueNote("Your result has been sent to the
                            ↪ server successfully.", 5);
69
70              i++;
71              shH.RemoveScreen(screen.Key);
72          }
73      }
74
75      public SendScreensUICmd(SendScreensHandler ssH, ShowGalleryHandler sGH,
            ↪ ListGalleryController gallery, ScreensHolderHandler sH)
76      {
77          ssH.enabled = false;
78          sGH.enabled = true;
79          listGallery = gallery;
80          shH = sH;
81          notesCtrl = Object.FindObjectOfType<NotificationsController>();
82      }
83
84      public void Execute(GameObject context)
85      {
86          screens = listGallery.GetSelected();
87          if (screens.Count > 0) Send();
88          listGallery.ResetList();
89      }
90
91      public string GetName() { return "Send Screens Command"; }
92  }
```

A.6.9 ShowGalleryUICmd class

```
1  using UnityEngine;
2  using UnityEngine.UI;
3
4  /// <summary>
5  /// Command for showing the list gallery
```

```
/// that allows the user to select what
/// screens they want for sending.
/// </summary>
class ShowGalleryUICmd : IUICmd
{
    /// <summary>
    /// Reference to the list gallery controller
    /// object.
    /// </summary>
    private ListGalleryController listGallery;

    public ShowGalleryUICmd(ShowGalleryHandler sGH, SendScreensHandler ssH,
        ↪ ListGalleryController gallery)
    {
        sGH.enabled = false;
        ssH.enabled = true;
        listGallery = gallery;
    }

    public void Execute(GameObject context)
    {
        if (listGallery.gameObject.active == false)
        {
            listGallery.gameObject.SetActive(true);
            listGallery.LoadGallery();
        }
    }

    public string GetName() { return "Show Gallery Command"; }
}
```

A.6.10 StartTakeUICmd class

```
using UnityEngine;

/// <summary>
/// Command for starting the take operation.
/// </summary>
public class StartTakeUICmd : IUICmd
{
    /// <summary>
    /// Reference to the take button.
    /// </summary>
    protected GameObject take;

```

```
13      /// <summary>
14      /// Reference to the accept button.
15      /// </summary>
16      protected GameObject accept;
17
18      /// <summary>
19      /// Reference to the app's notification controller
20      /// script.
21      /// </summary>
22      protected NotificationsController notesCtrl;
23
24      public StartTakeUICmd(GameObject acceptBtn, GameObject takeBtn)
25      {
26          take = takeBtn;
27          accept = acceptBtn;
28          notesCtrl = Object.FindObjectOfType<NotificationsController>();
29      }
30
31      public void Execute(GameObject context)
32      {
33          //Pause the camera output.
34          UnityEngine.Object.FindObjectsOfType<CameraController>()[0].Pause();
35
36          // Set the appropriate buttons
37          // to active/inactive.
38          take.SetActive(false);
39          accept.SetActive(true);
40
41          // Let the user know they can start
42          // annotatings.
43          if (notesCtrl)
44              notesCtrl.QueueNote("Please tap where you want your annotation to
                ↪ point to, then tap where to place the annotation text input
                ↪  box.", 5);
45      }
46
47      public string GetName() { return "Pause Screen Command"; }
48  }
```

A.6.11 TogglePanelUICmd class

```
1   using UnityEngine;
2
3   /// <summary>
4   /// Command for toggling a UI panel
```

```
/// active or inactive.
/// </summary>
class TogglePanelUICmd : IUICmd
{
    /// <summary>
    /// Reference to the menu's GameObject.
    /// </summary>
    private GameObject menu;

    /// <summary>
    /// Message to display as a notification.
    /// </summary>
    private string message;

    /// <summary>
    /// Reference to the app's notification controller
    /// script.
    /// </summary>
    protected NotificationsController notesCtrl;

    public TogglePanelUICmd(GameObject pnlMenu, string clickMessage = "")
    {
        menu = pnlMenu;
        message = clickMessage;
        notesCtrl = Object.FindObjectOfType<NotificationsController>();
    }

    public void Execute(GameObject context)
    {
        menu.SetActive(!menu.activeSelf);

        if (message != "" && menu.activeSelf == true && notesCtrl)
            notesCtrl.QueueNote(message, 5);
    }

    public string GetName() { return "Toggle Panel Command"; }
}
```

A.7 Remote server PHP code

A.7.1 db_engine.php

```
<?php

```

```
3  // Set server timezone to Europe/London
4  date_default_timezone_set("Europe/London");
5
6  // Get environment variables
7  $deploy_type = getenv("DEPLOY");
8  $db_host = getenv("DB_HOST");
9  $db_read_usr = getenv("DB_READ_USER");
10 $db_write_usr = getenv("DB_WRITE_USER");
11 $db_pass = getenv("DB_PASSWORD");
12
13 // Check if deployment is development or production
14 // And enable/disable error messages accordingly
15 if ($deploy_type == "Production")
16 {
17     ini_set("display_errors", 0);
18     ini_set("display_startup_errors", 0);
19     error_reporting(0);
20 }
21 else
22 {
23     ini_set("display_errors", 1);
24     ini_set("display_startup_errors", 1);
25     error_reporting(E_ALL);
26 }
27
28 // Initialise database connections for read and write operations
29 $db_read = new PDO("mysql:dbname=mres-json-server;host=".$db_host,
       ↪ $db_read_usr, $db_pass);
30 $db_read->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
31 $db_read->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, true);
32
33 $db_write = new PDO("mysql:dbname=mres-json-server;host=".$db_host,
       ↪ $db_write_usr, $db_pass);
34 $db_write->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
35 $db_write->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, true);
36
37 // Generates a 6-character random ID
38 function gen_id($len = 6)
39 {
40     $gen = "1";
41     $unique = false;
42
43     do
44     {
45         $gen = substr(str_shuffle(str_repeat($x='0123456789
              ↪ abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ', ceil(
```

```
            ↪ $len/strlen($x)))),1,$len);
        $unique = true;

        $sql = "SELECT * FROM Records WHERE Records.ID = '".$gen."'";
        $stmt = $GLOBALS["db_read"]->prepare($sql);
        $stmt->execute();
        $test = $stmt->fetchAll(PDO::FETCH_ASSOC);
        if (count($test) > 0)
            $unique = false;
    } while (!$unique);

    return $gen;
}

// Cleans a string of any special characters
function clean_str($str)
{
    return preg_replace('/[^0-9a-zA-Z\s]/', "", $str);
}

// Truncates a string by a certain amount of chars
function truncate_str($str, $count = 1)
{
    return substr_replace($str,"", -1 * abs($count));
}

// Parses an array into a format acceptable for SQL
function parse_array($array)
{
    if (!is_array($array))
        return "";

    $array_str = "";
    if (array() === $array)
    {
        for ($i = 0; $i < count($array); $i++)
            if ($array[$i] != "")
                $array_str .= $array[$i].", ";
    }
    else
    {
        $keys = array_keys($array);
        for ($i = 0; $i < count($array); $i++)
            if ($array[$keys[$i]] != "")
                $array_str .= $array[$keys[$i]].", ";
    }
```

```
91
92      return truncate_str($array_str, 2);
93  }
94
95  // Adds an incoming result to the database
96  function add_result($data)
97  {
98      // Data pack to send to the database
99      $pack = array(
100         "ID" => gen_id(),
101         "Received" => date("Y-m-d H:m:s"),
102         "Data" => $data);
103
104     // Attempt data insert
105     try
106     {
107         $str = "";
108         for ($i = 0; $i < count($pack); $i++)
109             $str .= "?,";
110         $str = truncate_str($str, 1);
111
112         $sql = "INSERT INTO Records (" . parse_array(array_keys($pack)) . ")
                ↪ VALUES (".$str.")";
113
114         $stmt = $GLOBALS["db_write"]->prepare($sql);
115         $stmt->execute(array_values($pack));
116
117         return array("result" => true, "success" => "Records item added
                ↪ successfully!");
118     }
119     catch (PDOException $e) { return array("result" => false, "error" => $e->
            ↪ getMessage()); }
120 }
121
122 // Retrieves saved results from the database
123 function get_results()
124 {
125     $sql = "SELECT * FROM Records ORDER BY Received";
126     $stmt = $GLOBALS["db_read"]->prepare($sql);
127     $stmt->execute();
128     return $stmt->fetchAll(PDO::FETCH_ASSOC);
129 }
130
131 // Retrieves saved result from the database by ID
132 function get_result($id)
133 {
```

```
    $sql = "SELECT * FROM Records WHERE Records.ID = '".$id."' ORDER BY
        ↪ Received";
    $stmt = $GLOBALS["db_read"]->prepare($sql);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

A.7.2 index.php

```
<?php

include "db_engine.php";

$results_html = "";
$results = get_results();

foreach ($results as $result)
{
    $results_html .= '
            <tr>
                <td>'.$result["ID"].'</td><td>'.$result["Received"].'</td><td
                    ↪ ><a href="result.php?id='.$result["ID"].'">View Data</a
                    ↪ ></td>
            </tr>';
}

echo '<!DOCTYPE html>
<html>
    <head>
        <title>Index | MRes Postural Assessment App JSON Server</title>
    </head>
    <body>
        <h1>MRes Postural Assessment App JSON Server</h1>
        <h2>Index</h2>
        <table>
            <tr>
                <th>ID#</th><th>Date Received</th><th>Link</th>
            </tr>'.$results_html.'
        </table>
    </body>
</html>';
```

A.7.3 result.php

```php
<?php

include "db_engine.php";

$id = "";
if (isset($_GET["id"])) $id = $_GET["id"];

$result_html = "";
if ($id == "") $result_html = "<p>Request invalid.</p>";
else
{
    $result = get_result($id);
    if (count($result) == 0) $result_html = "<p>Request invalid.</p>";
    else
    {
        $result_html .= '
        <p>'.$result[0]["Received"].'</p>
        <p>'.$result[0]["Data"].'</p>';
    }
}

echo '<!DOCTYPE html>
<html>
    <head>
        <title>Viewing Result #'.$id.' | MRes Postural Assessment App JSON Server</title>
    </head>
    <body>
        <h1>MRes Postural Assessment App JSON Server</h1>
        <h2>Viewing Result #'.$id.'</h2>'.$result_html.'
    </body>
</html>';
```

A.7.4 table_structure.sql

```
-- phpMyAdmin SQL Dump
-- version 5.1.0
-- https://www.phpmyadmin.net/
--
-- Host: mres-json-db
-- Generation Time: Jul 10, 2021 at 08:36 PM
-- Server version: 8.0.25
-- PHP Version: 7.4.16

```

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `mres-json-server`
--

-- --------------------------------------------------------

--
-- Table structure for table `Records`
--

CREATE TABLE `Records` (
  `ID` varchar(6) NOT NULL,
  `Received` datetime NOT NULL,
  `Data` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
COMMIT;

ALTER TABLE `Records` ADD PRIMARY KEY(`ID`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

A.7.5 upload.php

```
<?php

include "db_engine.php";

if (count($_POST) > 0)
{
    $result = add_result($_POST["JSON"]);
    die();
}
```