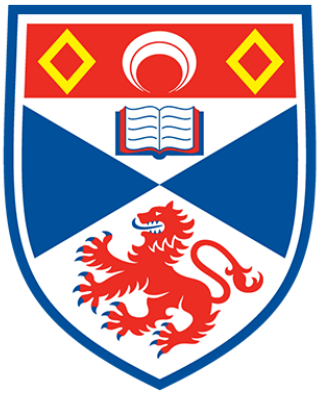
SensoGather: An iOS Application For Human Activity Data Collection

Arman Özkaya



University of
St Andrews

This thesis is submitted in partial fulfilment for the degree of
MSci Computer Science
at the University of St Andrews

Supervisor: Juan Ye

January 17, 2025

Abstract

The increasing adoption rate of wearable devices that have multiple motion sensing sensors such as accelerometer, gyroscopes, and magnetometers, and the development of efficient deep learning (DL) models provide a unique opportunity to unobtrusively capture and analyze data from users to detect human activity in real-time. However, researchers who are attempting to deploy software systems that are capable of human activity recognition (HAR) to a real world setting are facing numerous challenges. Most of these challenges can be traced back to the lack of a large, high quality, general purpose dataset of human activity sensor readings which also hinders the reproducibility of much research, slowing down improvements. Existence of general purpose datasets have proved to be crucial in the advancements of other DL research fields such as speech recognition or computer vision. This paper aims to help replicate the success of other DL research fields in the HAR field by systematically categorizing and summarizing existing work and major challenges, and providing a comprehensive analysis of the design, implementation, testing, and evaluation of a mobile software system that can be used to collect high quality data in order to enable reproducible research and construction of a general purpose dataset.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 16708 words long, including project specification and plan.

Contents

1. Introduction

In 2024, more than 7 billion mobile devices were in use [20]. In the past decade, the introduction of wearable devices, that are companion to and sold alongside mobile devices, to the consumer market has led to an increase in adoption rate of wearable technology among users. In 2023 the adoption rate of wearable technology among adults in the United States increased to 44% [2].

The increasing adoption rate of wearable devices that have multiple motion sensing sensors such as accelerometer, gyroscopes, and magnetometers, the increasing ability of mobile devices to offer more computational power and faster connectivity in our everyday lives, and the development of efficient deep learning methods provide a unique opportunity to unobtrusively capture and analyze data from the users to detect human behavior and activity in real-time.

Software systems capable of data collection and data analysis are referred to as Human Activity Recognition (HAR) systems which can provide improvements to the daily lives of many individuals from different groups. For instance, a HAR app can help users to better understand their habits to enable a healthier living style or help doctors to track the treatment progress of patients who suffer from pulmonary diseases.

However, researchers who are attempting to deploy a HAR system to real world settings are facing numerous challenges even though some systems perform adequately during development. Most of these challenges can be traced back to the lack of a high quality general purpose dataset of human activity which can be used to train deep learning models similar to other ML research fields such as speech recognition or computer vision. The contributions of this project are summarized as follows:

- Firstly, to provide an overview of the background of the human activity recognition research field, including the sensor suits the research community is utilising to obtain data, the application area of HAR systems, common research challenges that are experienced, and the characteristics of widely-used publicly available datasets for deep learning model training.
- Secondly, to introduce the system description, system requirements, system design, and system implementation of a mobile software app I developed which uses a companion smartwatch app. This mobile app can be used to address the challenges HAR researchers face and enable the construction of a standardized general purpose dataset with the aim of making HAR research more reproducible.
- Thirdly, to evaluate the software system to analyse whether this HAR app can be extended further to become a HAR system that can also do DL model training and inference.
- Fourthly, the app is open sourced so that other researchers can realize the benefits of the system designed for this project.

2. Context Survey

Human activity recognition (HAR) refers to a wide area of research that investigates the process of using various machine learning (ML), specifically deep learning (DL) algorithms to do detection of various human activity types.

The DL model, the model's hyperparameters, the aim of the HAR application, and the tensor shape of input data can vary due to but not limited to reasons such as HAR application's domain of use, model's training and inference location (on-device, on server, hybrid approach), the heterogeneity of edge devices on which a HAR application is deployed, the extent of computational power scarcity on these edge devices, and different approaches to manage resource constraints of edge devices. These listed differences naturally lead to many different types of challenges that researchers aim to address. Therefore, this chapter provides a context review of the available literature in the domain of HAR in order to enable a further understanding of our contributions in this paper.

Inspired by the research in [1], this project classifies existing research in the HAR field into four classes depending on the sensor suite utilized to obtain input data, application domain of the HAR application, and the research challenge that is addressed. The classification of the research area is shown in Figure 1.

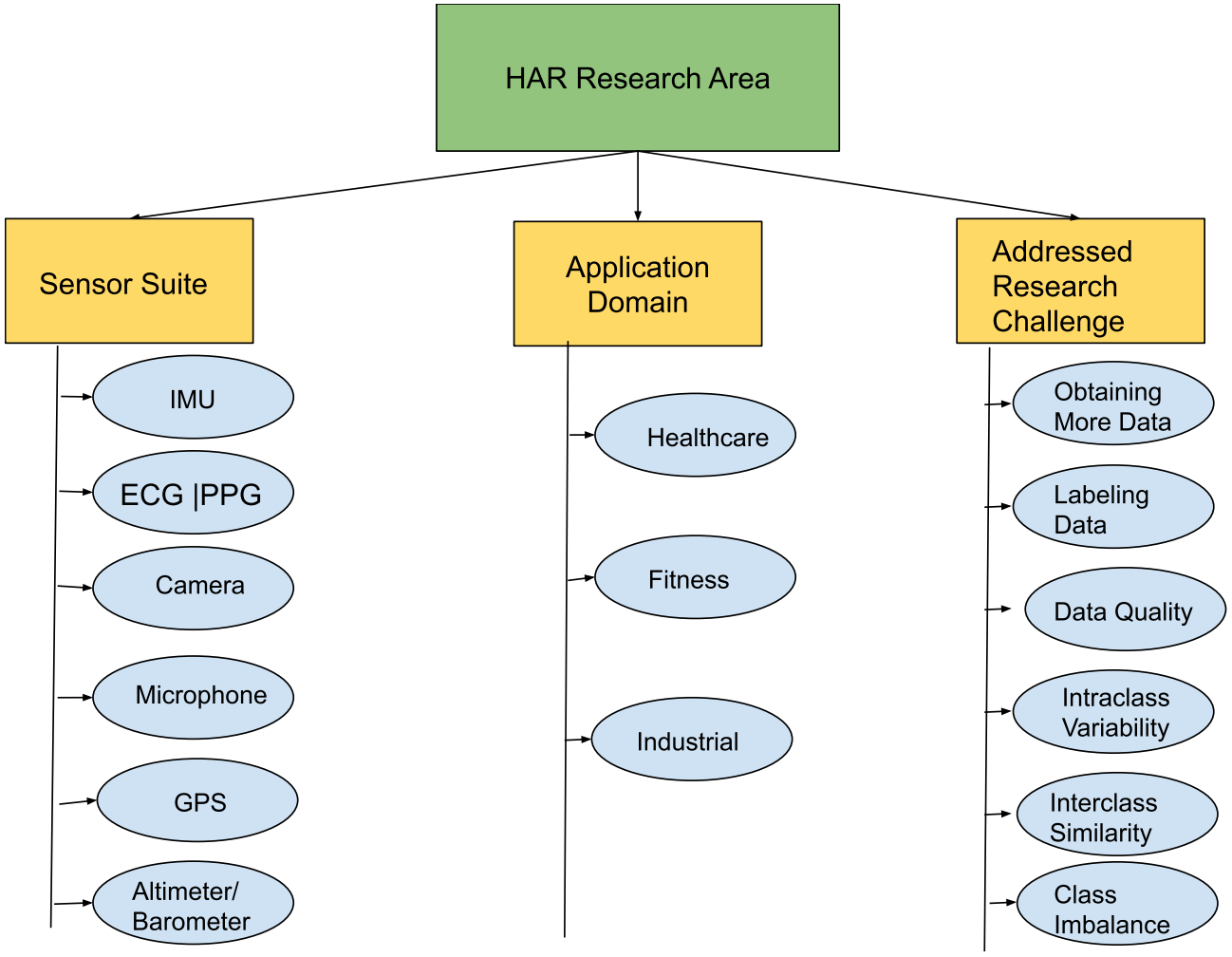


Figure 1: Classification of research in the HAR area

2.1 Sensor Suite

Mobile computation is the primary enabling factor for HAR as the sensor rich mobile devices in our everyday lives provide a unique opportunity to unobtrusively capture contextual information from the underlying human behavior in real-time [9]. Therefore, most research undertaken in this field is aimed at enabling a mobile HAR system that can be deployed to users' smartphones and smartwatches. Consequently, most researchers focus on using the sensors that are commonly available on mobile devices and wearable devices to obtain input data for DL models.

The most commonly available sensor suite on mobile devices is the inertial measurement unit (IMU) which is a sensor package that encapsulates accelerometer, gyroscope, and magnetometer [1]. An accelerometer detects linear motion and gravitational forces by measuring the acceleration in 3 axes (x, y, z), a gyroscope measures rotation rate (roll, yaw, and pitch), and the magnetometer is used to detect and measure the earth's magnetic fields [1].

Furthermore, most wearables today have ECG and PPG sensors on them which can be used for heart rate monitoring. ECG, also called EKG, detects the heart's electrical activity through electrodes attached to the body which is primarily employed to detect and diagnose cardiovascular disease and abnormal cardiac rhythms [1]. On the other hand, PPG sensors use infrared (IR) light to measure blood flow fluctuations caused by the expansion and contraction of the heart which can be used to obtain an estimate of heart rate [1].

Other sensors used in the field include a barometer for obtaining atmospheric pressure, altimeter for height sensing, GPS for location sensing, microphones for audio recording, and wearable cameras for image or video recording [1]. These additional sensors are utilized with the aim of improving a DL model's performance by obtaining more contextual information while an activity is being performed.

2.2 Application Domain

HAR research area is aimed at many applications such as fitness monitoring, healthcare, biometrics, surveillance, and human computer interaction [8]. This section thus provides an overview of the most prevelant domains where researchers have deployed HAR applications.

2.2.1 Fitness Monitoring

Regular physical activity such as running, swimming, rowing, and walking is continuously being linked to a reduction in risk for many chronic diseases including obesity, diabetes, cardiovascular disease, and has been shown to improve mental health [3]. Moreover, HAR encapsulates recognizing a mode of transportation an individual uses which has been linked to mitigating or leading to obesity-related medical conditions [1]. These findings have been shown to people around the world through TV shows, advertisements, and education curriculums which technology companies have leveraged to advertise wearable technologies that contain a HAR system. Commonly available wearable devices such as Apple Watches, WHOOP trackers, FitBit trackers, and Garmin watches all have dedicated and IP protected software and hardware that enables energy expenditure (EE) estimation which has grown to be an important reason why consumers buy wearable devices [1].

Furthermore, these trackers can also capture and display data after a physical activity that details the intensity of an activity alongside biometric data such as heart rate recovery which can give an estimation about the cardio fitness of an individual.
EE and cardio fitness estimation are outputs of HAR applications. Researchers are working on algorithms to improve the accuracy of these parameters which are used to enable self-regulation of one's own habit to prevent chronic diseases [1].

2.2.2 Healthcare

The digitization of healthcare tracking with the current proliferation of mobile technologies is helping to pave the path to a paradigm shift in which people's health information is timely and ubiquitously available [11]. On top of this digitization, portable and wearable sensors are increasingly utilized to collect data on individuals' biology, psychology and behavior [11].

Healthcare can be grouped into 2 major categories which is prevention and treatment. A HAR system that collects valuable data through wearable devices and analyzes this data can be used in both of these categories of healthcare. For instance, a HAR system can analyze collected data to objectively detect behavioral risk factors for chronic diseases, including but not limited to sedentary behavior, sleep, and physical activity through leveraging the personal or lived experiences of individuals [3]. Further, several researchers have introduced HAR systems for monitoring and assessing pulmonary disease symptoms such as Chronic Obstructive Pulmonary Disease (COPD), and asthma by using wearables to detect cough activity, a major symptom of pulmonary diseases [1].

Consequently, HAR applications are used in healthcare to reduce disease risks, optimize treatment outcomes, and yield new insights into the factors that lead to disease [11].

2.2.3 Human Computer Interaction

Modern HAR technology has provided us with flexible and convenient methods to control and communicate with electronics, computers, and robots [1]. HAR applications in the context of human computer interaction (HCI) can be categorized into two groups: Accessibility enabling HAR systems and industrial HAR systems. For instance, an apple watch can detect when a user double taps their thumb and index finger which is used to control various functionalities of the paired iPhone such as skipping a song. This feature can be helpful for individuals who have mobility issues. Further, there is ongoing research to bring HAR systems [13] to industrial settings to boost efficiency of enterprises by helping them achieve a seamless communication with various productivity machinery. For example, a HAR system can be used on an assembly line of automotives in order to help human robot interaction (HRI) systems to assess the current state of interaction with the human to predict the following events in the workflow [13].

2.3 DL Workflow

A typical DL workflow to build a HAR system involves specific steps. Understanding the steps involved in this workflow is necessary before we can analyze the research challenges in the HAR field.

Researchers in [5] introduced the concept of an Activity Recognition Chain (ARC) to standardize HAR systems' design and evaluation. ARC concept comprises data acquisition, data preprocessing, data segmentation, feature extraction, feature selection, training, classification, decision fusion, and performance

evaluation steps [5]. Each stage of the ARC framework can be implemented using a variety of methods, for instance, we can select a specific set of features or a specific classifier [5]. The selected methods and their parameters directly influence the overall activity recognition performance of the system [5]. In addition, several stages of the framework form a pipeline and need to be evaluated jointly to achieve high recognition performance [5]. The optimal solution to this problem can only be found by using sophisticated, multidimensional optimization procedures which are outside the scope of this paper [5]. Instead, in this section we present a brief summary of the involved steps of the ARC framework in order to introduce the challenges pertaining to the ARC framework which generalizes to any machine learning research undertaken in this field.

2.3.1 Sensor Data Acquisition And Preprocessing

The first stage of any ML workflow is data acquisition and preprocessing. Data is typically acquired through different sensors that were discussed previously in section 2.1. Each of the sensors is sampled at regular intervals, which results in a multivariate time series [5]. Since sensors may provide multiple values (e.g., an acceleration sensor provides a 3D acceleration typically referred to as x, y, and z direction), or multiple sensors are jointly sampled, vector notation is used to describe the sensor's output [5]:

During preprocessing, data is synchronized in case of varying sampling rates across different sensor modalities. Furthermore, the introduced signal noise during data acquisition due to hardware or environmental interference is removed to prepare the data for feature extraction [5]. The preprocessing stage transforms the raw time series data into a preprocessed time series D':

$$D' = \left(d_1', \ldots, d_n'\right)^T \quad \textit{where } d_i' \textit{ corresponds to one dimension of the preprocessed time series, n to the number of total data dimensions, and t to the number of samples [5].}$$

The transformation aims to enhance the robustness of the feature extraction by applying signal preprocessing techniques that reduce noise or filter out unwanted artefacts such as linear interpolation, spline interpolation, low-pass finite impulse response filters, and high-pass Butterworth filter of low order [3,5].

2.3.2 Data Segmentation

The data segmentation stage identifies data segments that are likely to contain information about activities [5]. Information on activity segments is useful for improving classification performance, but it also can be used for other purposes such as resource allocation and power saving which is of huge interest in the edge computation community as well as the machine learning community [14, 5]. Each data segment has a start time t_1 and end time t_2 within the time series [5]. The segmentation stage yields a set of segments W containing a potential activity y:

$$W = \{w_1, w_2, \ldots, w_m\}$$

Segmenting a continuous sensor stream is a difficult task. Typically, data is segmented through the sliding window technique. In the sliding window technique, a static window is moved over the time series data to extract a data segment that is then used in subsequent ARC stages [5]. The window size directly affects model performance and it is subject to a tradeoff between segmentation precision and computational load [5]. Some research has been undertaken to determine the optimal window size to do data segmentation, and there has been

evidence to suggest that an optimal window size would be the duration of time to complete a single repetition of the activity that is aimed to be recognized [3]. However, this is not a particularly helpful technique in the context of a HAR system that aims to recognize many activities that are of different types such as an arm gesture and locomotion.

2.3.3 Feature Extraction and Selection

The feature extraction and selection stage reduces the signals into distinct features [5]. Feature extraction can be done automatically through deep learning methods such as autoencoders [1]. Features are extracted as feature vectors X_i on the set of segments W, with F being the feature extraction function:

$$X_i = F(D', w_i)$$

2.3.4 Training And Classification

The classification is typically performed by a supervised learning algorithm that has been trained to recognize patterns between features and labeled physical activities in the training dataset [3]. The reviewed literature included a broad range of classifiers ranging from simple machine learning models to architecturally complex deep learning models such as simple decision trees, k-nearest neighbors, support vector machines, logistic regression, random forest, XGBoost, AdaBoost45,96, and deep neural networks [3]. Researchers have demonstrated that ensemble classifiers tended to outperform individual or single classifiers and deep learning classifiers outperformed both individual and ensemble classifiers [1,3]. For on-device applications, continuous learning can optimise the selection of a low cost classification algorithm [3]. The choice for a particular inference method is subject to a tradeoff between computational complexity and recognition performance [5].

2.3.5 Decision Fusion

Multiple sensors or multiple classifiers were shown to increase classification performance of ML models [5]. The decision fusion stage combines several intermediate classification results into a single decision. The working principle of this stage can be thought to be similar to the stacking approach of ensemble classifiers. Although Bayesian approaches to fusion are favoured, the limited resources of edge computation systems often require limiting the complexity of the fusion approaches to alternatives such as majority voting and summation [5]. On top of the increased activity recognition performance of ML models, sensor fusion has additional benefits such as fault tolerance that may be needed in case sensors fail and reduced classification problem complexity through the use of low cost classifiers which is feasible after certain activities are pruned from the decision space by another sensor modality providing contextual information. For instance, a user's low heart rate may rule out the activities that involve high intensity [5].

2.3.6 Performance Evaluation step

Performance evaluation is the last stage of ARC. Within this project's literature review, it was discovered that analysis results were typically reported in terms of classification accuracy using various standard metrics like precision, recall, and F-score [3]. More nuanced summaries used the confusion matrix, which enables the examination of activities that are more likely to be classified incorrectly [3]. This approach was particularly useful for visualizing classification differences between similar activities, such as normal and fast walking. Additional statistics are usually provided in the context of HAR systems designed to operate on edge devices. In these cases, processing time, battery drain, and computation time was reported [3].

The overall sequence of the ARC framework can be found in Figure 2 below.

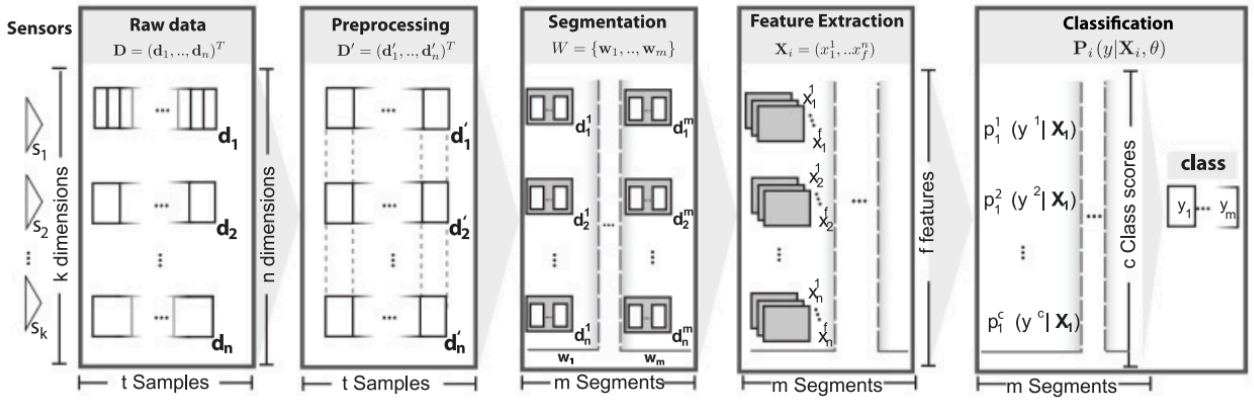


Figure 2: ARC Framework Steps [5]

2.4 Challenges Pertaining To The ARC Framework

This section will present the challenges that are associated with various stages of the ARC framework which summarizes challenges associated with the HAR field.

2.4.1 The Need For More Data

The lack of a sufficiently large size of labelled high-quality data is a major reason why DL methods cannot produce the expected improvements in classification accuracy [1]. Training a DL model with a limited dataset size will result in the model being prone to overfitting, and the generalizability of the model to other activity types can not be attained [1].

2.4.2 Difficulties of Labeling Data

Collection of annotated training data is an expensive and tedious task, as the annotator has to perform the annotation in real time [5]. In addition, sensor data recorded from an IMU is often more difficult to interpret than other sensor modalities such as cameras [5]. In stationary and laboratory settings, annotation can often be obtained by relying on post hoc labeling based on video footage, however, in daily life settings labeled data collection always involves the user who may not be trained on accurately labeling data [5].

2.4.3 Insufficient Data Quality Due to Variability In Sensor Characteristics

Data is crucial for developing accurate DL models as models' performance is directly related to the quality of the training data [1]. A challenge for implementing HAR in real-world applications is caused by the sensing equipment [5]. Firstly, mobile devices are often equipped with low cost commercial IMU sensors which are often inaccurate, and of limited granularity and range compared to dedicated sensors [9]. Secondly, a sensor's accuracy and capability of measurement may change over time through accidental device dropping [9].Thirdly, different sensors have varying sampling rate heterogeneity (SRH) [9]. An accelerometer in an iPhone can have a 100 Hz sampling rate while a Samsung Galaxy's accelerometer may be limited to 50 Hz. This heterogeneity makes it more difficult to train ML models [9]. Lastly, a number of factors, including delays in OS level timestamp attachment to sensor measurements and instantaneous I/O load, affect both the actual as well as the reported sampling rate of sensors on a device [9].

2.4.4 Intraclass Variability

Intraclass variability is a general pattern recognition problem which also applies to the field of HAR [5]. In HAR, the intra class variability presents itself in the form of different patterns in raw sensor recordings which occur due to the varying ability, understanding, or preference of different individuals while performing a given activity. Furthermore, intraclass variability can also occur when the same individual is performing the activity due to stress, fatigue or environmental factors in which the activity is performed [5]. For example, the walking gait of an individual might differ depending on whether it is the morning after a full night's rest when compared to the evening after a full day of work [5]. This issue can be mitigated by increasing the amount of training data of a ML model [5].

2.4.5 Interclass Similarity

Some activity classes that are fundamentally different such as drinking a cup of coffee or drinking a glass of water may show very similar readings in the sensor data [5]. Such close similarity can often only be resolved by capturing additional contextual information through different sensor types or by analyzing co-occurring activities, in this example the activities of using the coffee machine or opening the tap, respectively [5].

2.4.6 Class Imbalance

When an individual is observed over a period of time, only few activities occur often, such as sleeping or working, while most activities occur rather infrequently, such as taking a sip of a drink [5]. This problem can cause overfitting of the model to favour activities that are more frequently performed. In general pattern recognition, class imbalance can often be addressed rather easily by recording additional training data or by generating artificial training data through duplicating [5].

2.5 Datasets Used For Training Human Activity Recognition ML Models

In contrast to other ML research fields such as speech recognition or computer vision, the research community in the HAR field has not yet started a joint effort to collect general-purpose data of human physical activity [5]. However, there is an urgent need to start this initiative for a number of reasons. To begin with, as explored in section 2.4, most of the challenges associated with HAR can be addressed through the construction of a general-purpose dataset that has a large data size which contains high quality sensor data of various sensor types that is collected for a wide range of activities. Moreover, different researchers may focus on quite diverse data requirements such as large numbers of modalities or long-term recordings [5]. Lastly, designing and conducting a data collection experiment is a challenging task which prevents more research in the HAR field as researchers are faced with a tradeoff between unobtrusiveness and ease of use of the sensors; the time required to conduct the experiment; and the costs for participants, experimenters, and the equipment [5]. These reasons prove that the construction of a standardized dataset is crucial for reproducible research [5].
This section seeks to review the currently used datasets that are used to train ML models in order to address their shortcomings through our data collection app. A summary of the mostly cited datasets and their features can be found in table 1 [1].

Dataset	Application	Sensor Suit	Number Of Activity Classes	Dataset Participant Number	Sensor Sampling Rate	Citations Per Year
WISDM	Locomotion	Commercial (Smartphone and Smartwatch) 3D IMU	18	51	~20 Hz	217
ActRecTut	Hand Gestures	Scientific 9D IMU	12	2	32 Hz	153
UCI-HAR	Locomotion	Commercial (Smartphone) 9D IMU	6	30	50 Hz	78
SHO	Locomotion	Commercial (Smartphone) 9D IMU	7	10	50 Hz	52
UTD-MHAD	Locomotion & Daily Activities	Scientific 3D Accelerometer	27	8	50 Hz	39
HHAR	Locomotion	Commercial 3D Accelerometer	6	9	50–200 Hz	37
Daily & Sports Activities	Locomotion	9D IMU	19	8	25 Hz	37
MHEALTH	Locomotion & Gesture	Scientific 3D Accelerometer & ECG	12	10	50 Hz	33
PAMAP2	Locomotion & Activities	9D Scientific IMU & HR monitor	18	9	100 Hz	32

Table 1: Most Cited Datasets In The HAR Field

2.5.1 WISDM Dataset

The researchers collected data from the accelerometer and gyroscope sensors of a smartphone and smartwatch while participants were performing a scripted activity [4]. The access to sensor data was obtained through a custom-made app that ran on both the smartphone and the smartwatch while the participant was performing the activity [4]. The app wasn’t made open source.

Specifically, this dataset was constructed by having each participant perform 3 minutes of 18 different activities of daily living such as brushing teeth, jogging, eating pasta, etc [4]. Each participant wore the smartwatch on their dominant hand. The smartphone used to run the app was either the Google Nexus 5/5X or Samsung Galaxy S5 running Android 6.0 [4]. Additionally, the smartwatch was the LG G Watch running Android Wear 1.5 [4].

2.5.2 ActRecTut Dataset

The researchers collected accelerometer and gyroscope data through a custom IMU while participants were performing the scripted activity [5]. The IMU was placed on top of each participant's right hand, as well as on the outer side of the right lower and upper arm [5].

Specifically, this dataset was constructed by having each of the IMUs record arm movements of two people performing a continuous sequence of eight activities of daily living. Each activity lasted between two and eight seconds and was repeated 26 times resulting in a total dataset of about 70 minutes. To increase data diversity, researchers also recorded typical arm movements performed while playing tennis which are forehand, backhand, and a smash. The researchers also included periods with no specific activity where participants were instructed not to engage in any 11 specified activities which leads to a total of 12 class recognition problems. All recorded data was sent through Bluetooth to a laptop where data synchronization was performed through SenseHub synchronization software. The sampling rate for the sensors was 32 Hz. Participants were observed by an assistant who instructed them and manually annotated their current gesture.

2.5.3 UCI-HAR Dataset

The researchers collected data from the accelerometer and gyroscope sensors of a smartphone while participants were performing a scripted activity.

Specifically, this dataset was constructed by having 30 volunteers within the age bracket of 19-48 years performing 6 different activities such as walking, sitting, standing. Each participant wore a strap on their waist where a Samsung Galaxy S2 phone was placed. The sampling rate for the sensors was set at 50 Hz. The experiments have been video-recorded to label data manually.

2.5.4 SHO Dataset

The researchers collected data from the accelerometer, gyroscope, a magnetometer, and a linear acceleration sensor of a smartphone while participants were performing a scripted activity. A custom app was used for data collection.

Specifically, this dataset was constructed by having 10 male participants within the age bracket of 25-30 years performing 7 different activities with a sensor sampling rate of 50Hz. These activities were walking, running, sitting, standing, jogging, biking, walking upstairs and walking downstairs. Each activity was performed for 3 to 4 minutes. The experiments were carried out indoors except for biking. Each participant was equipped with five Samsung Galaxy SII smartphones on five body positions, with one position simulating a smart watch. These positions are as follows:

1. one in their right jeans pocket;
2. one in their left jeans pocket;
3. one on the belt position towards the right leg using a belt clip;

4. one on the right upper arm;
5. one on the right wrist.

2.5.5 UTD-MHAD Dataset

The researchers collected data from a Microsoft Kinect camera and a wearable accelerometer. In total researchers obtained four data modalities including RGB videos, depth videos, skeleton positions, and inertial signals.

More specifically, 861 data sequences obtained by 8 participants performing a scripted activity. Data labeling was done manually and offline. The accelerometer sensor was placed on the participant's right wrist or the right thigh depending on activity type. A full list of the activities performed can be found in the table below.

2.5.6 HHAR Dataset

The researchers collected data from the accelerometer of 8 smartphones and 2 smartwatches. There were 9 study participants within the age range of 25-30.

Specifically, participants performed six different daily activities: Biking, sitting, standing, walking, stair up and stair down. Each activity was conducted for 5 minutes. During the activity execution, all eight smartphones were kept in a tight pouch which was strapped to participants' waists. The data was obtained through a custom app.

2.5.7 Daily & Sports Activities Dataset

The researchers collected data from 5 scientific MTx 3-DOF IMU suits. The sensors were placed in five different places on the subject’s body.

Specifically, participants performed 19 activities. There were 4 female and 4 male participants in the study. The sensor suit was calibrated to acquire data at 25 Hz sampling frequency. Each activity was performed for 5 minutes. The participants' age range differed between 20 and 30.

2.5.8 MHEALTH Dataset

The researchers collected data from multiple scientific 3 axial accelerometer sensors named Shimmer2 which were placed on 3 different body locations. This sensor also was able to obtain vital data through a 2 lead ECG.

Specifically, 10 participants performed 12 different physical activities. The sampling rate for all of the sensors was set at 50Hz. The data was collected out of the lab.

2.5.9 PAMAP2 Dataset

The researchers collected data from 3 Colibri wireless inertial measurement units and a heart rate monitor.

Specifically, data was obtained while 9 subjects performed 19 scripted different physical activities.
The sampling frequency of the inertial sensors was set at 100Hz and the sampling frequency for the heart rate monitor was set at 9Hz.

2.6 Limitations of the Datasets

While these datasets are used in HAR research, they fail to become a general purpose dataset because none of the datasets can address all challenges discussed in section 2.4 to enable reproducible research. This section highlights how these datasets fail to address the requirements of reproducible research. Firstly, the WISDM dataset couldn't address the sampling rate heterogeneity. While the researchers specified a 20 Hz sampling rate for the sensors, this was a suggestion to the operating system and the polling of the sensors could be delayed if the processor was busy. Secondly, the ActRecTut, UTD-MHAD, Daily & Sports Activities, MHEALTH, PAMAP2 datasets all used different custom IMU suits which have different sensor characteristics that prevents the reproducibility of ML models' accuracy trained on these datasets when deployed to be used on commercial IMU suits that are available on wearable and mobile devices. Thirdly, these datasets have a small number of participants which aggregate the intraclass variability and interclass similarity issues. Fourthly, these datasets have a small data size which aggregate the class imbalance issue. Lastly, placement of sensors such as the waist area on the HHAR dataset isn't a natural mobile device or wearable device placement position.

3. System Description

The aim of this system is to develop an iOS application to support the machine learning community in HAR. This mobile app should enable the construction of a standardized database that contains various sensor recordings that originate from a wearable device. This iOS app needs to be able to communicate with an apple watch app which will be used to obtain the data through its sensors.

The software system will be used by different groups of people such as university students and professional researchers who may use the app for user studies, and user study participants who will interact with the app while performing physical activities. Due to the time limitation of the dissertation, the development aim is to create a minimum viable product (MVP) with the features most crucial for enabling reproducible machine learning research which can be refined over time based on feedback from target users.

The sensors that are most important for activity recognition and should be used to collect data are accelerometer, gyroscope, magnetometer, heart rate sensor, and GPS sensor. Each sensor recording session should result in a CSV file which contains sensor recordings. Each instance of sensor recording should be time stamped. For sensors that produce multiple values for an instance of recording (such as an accelerometer which produces 3 values for the x, y, and z axis per instance of recording) each value should be recorded separately.

The CSV file should be saved on the mobile device. The CSV file should have a name that can be used to identify the user who produced the recording, the label of the activity, and the timestamp of creation.

All users must be signed up to the application before they can use it. The signing up process requires first name, last name, and the birth date. Alternatively, users should be able to sign in with a 0 auth system to provide ease of use. Each sign up request will be checked and only those who are over the age of 18 will be accepted by the system.

Once logged in, users should be able to click on a predefined activity to start the sensor recordings for that activity. Alternatively, if the activity they wish to have sensor recordings for doesn’t already exist, the users should be able to define a new activity type to be used for sensor recordings. Upon clicking an activity, the user should be directed to a new page where they can see the activity type they selected, instructions to start sensor recordings and the duration of the recording session in seconds. If the user attempts to start a recording without opening the companion app on the apple watch, the user should be presented with an alert that reminds them to open the watch app. During the session recording, the remaining time in the session duration should be counted down. The user must be presented with the opportunity to end the session early. After a session has ended, the time should be reset and the system should be ready for another session recording.

On the apple watch app, the user should be easily able to see whether an activity recording session is in progress or whether the system is idle. The user should also be able to see sensor readings on the apple watch while an activity is being recorded. Additionally, the user should be able to use the watch app directly instead of interacting with the iOS app to start or end the session. On the iOS app, the user should be able to change their username, and turn off or on various sensors that are used to record data.

4. Specification of User Stories

This chapter details the user stories derived from the system description. The user story table contains each user story with the following fields: ID (identifier for the user story), Requirement (extracted from the system description), User Story (the derived user story), Acceptance Criterion (the criterion by which the user story is evaluated to be implemented), and Architectural Impact (how much a user story affects system architecture)

ID	Requirement	User Story	Acceptance Criterion	Architectural Impact		
				H	M	L
01	This iOS app needs to be able to communicate with an apple watch which will be used to obtain the data through its sensors.	As a user, I want to use the sensors of the Apple watch to obtain data so that I can produce reproducible research results.	**Scenario:** The user has access to an apple watch and an iphone. **Given:** That I have downloaded the iOS and watchOS app, both apps are open, and I have clicked on an activity type. **When:** I click "Start Activity" **Then:** I want sensors on apple watch to start producing data	✓		
02	The sensors that are most important for activity recognition and should be used to collect data are accelerometer, gyroscope, magnetometer, heart rate sensor, and GPS sensor.	As a user, I want the accelerometer, gyroscope, magnetometer, heart rate sensor, and GPS sensor to be available during data collections so that I can use well researched ML models for my ARC pipeline.	**Scenario:** The user is recording sensor data. **Given:** That I have started an activity recording session. **When:** The recording session ends. **Then:** I want to have access to recordings from accelerometer, magnetometer, gyroscope, heart rate and GPS sensors.	✓		
03	Each sensor recording session should result in a CSV file which contains sensor recordings.	As a user, I want to be able to save my data in a CSV format so that I can easily start my machine learning workflow.	**Scenario:** The user is saving recorded sensor data. **Given:** That I have finished an activity recording session. **When:** I export my data. **Then:** I want my data to be saved in CSV format.		✓	
04	Each instance of sensor recording should be time stamped.	As a researcher, I want my sensor recordings to be time stamped so	**Scenario:** The researcher is doing preprocessing. **Given:** That I have exported my data When: I am analyzing my data			✓

		that I can do data preprocessing easier.	**Then:** I want to be able to see timestamps of my sensor recordings.			
05	For sensors that produce multiple values for an instance of recording, each value should be recorded separately.	As a researcher, I want to have separate access to each sensor data value that is produced by the same sensor for an instance of recording so that I can make data preprocessing easier.	**Scenario:** A sensor that produce multiple values for an instance of recording is used to record data **Given:** That data has been collected. **When:** I open the CSV file. **Then:** I want to see all recordings of data to be separated with commas.			✓
06	The CSV file should be saved on the mobile device.	As a user, I want to be able to save my CSV data on my mobile device so that I can persist the recorded data.	**Scenario:** The recorded data is being saved. **Given:** That I have finished the data recording When: I click the “Export” button **Then:** I want to see an option to save the file to my mobile device.		✓	
07	The CSV file should have a name that can be used to identify the user who produced the recording, the label of the activity, and the timestamp of creation.	As a researcher, I want to be able to identify the user who produced the recording so that I can verify that data collection has been carried out as proposed in my research methodology.	**Scenario:** The researcher is verifying data recordings. **Given:** That user names have been distinct. **Then:** I want to be able to identify to which user a CSV recording belongs to.			✓
09		As a researcher, I want to be able to identify the label of the activity for a recording so that I can train my ML model.	**Scenario:** The researcher is training a ML model. **Given:** That sensor recordings have been saved. **Then:** I want to be able to understand which file belongs to which activity type.			✓
10		As a researcher, I want to be able to identify when a recording was created so that I can understand whether I have used the data	**Scenario:** The researcher is training a ML model **Given:** Sensor recordings have been saved **Then:** I want to be able to differentiate between recorded files based on their time of creation.			✓

		already for my model training.				
11	All users must be signed up to the application before they can use it.	As a prospective user I want to be able to sign up to the application so that I can start using the system.	**Scenario:** The user is attempting to access the system. **Given:** That the iOS app has been downloaded. **Then:** The user must be presented with a landing page to enable login when the iOS application is opened.	✓		
12	The signing up process requires first name, last name, and the birth date.	As a prospective user, I want to type in my first name, last name, and birth date on the landing page so that I can obtain correctly labeled data files.	**Scenario:** User is signing up to the app. **Given:** That I have navigated to the landing page of the app by opening the app. **Then:** I want to be presented with different fields to input my first name, last name, and birth date and submit my request to sign up by clicking the signup button again.		✓	
13	Users should be able to sign in with a 0 auth system to provide ease of use.	As a prospective user, I want to use "Sign in With Apple" so that I can sign up to the app more easily.	**Scenario:** User is signing up to the app. **Given:** That I have navigated to the landing page of the app by opening the app. **Then:** Then I want to be presented with a "Sign In With Apple" button to use a 0auth system.	✓		
14	Each sign up request will be checked and only those who are eligible will be accepted by the system.	As the developer, I want to ensure each sign-up request is checked for age eligibility (over 18) so that only users who meet the age requirement can use the system.	**Scenario:** User is signing up to the system. **Given:** That the user clicked on the signup button at the landing page, is less than 18 years of age, and they have indicated their birthdate correctly. **Then:** The details the user has inputted are automatically checked and an alert is displayed noting that they can not use the system due to their age.	✓		
15		As the developer, I want to ensure each sign-up request is checked	**Scenario:** User is signing up to the system. **Given:** That the user clicked on the signup button at the landing page after not filling in all the	✓		

		so that only users who provide their first name and birthdate can use the system.	required fields. **Then:** The details the user has inputted are automatically checked and the user is presented with an alert noting that they can not use the system because they haven't filled in at least one of the mandatory fields.			
16		As the developer, I want to ensure I develop a user friendly interface that easily enables the users to understand which login fields are mandatory.	**Scenario:** User is signing up to the system. **Given:** That the user clicked on the signup button at the landing page after not filling in all the required fields. **Then:** I want the user to see an asterisk next to fields that are mandatory for signing up.		✓	
17	Once logged in, users should be able to click on a predefined activity to start the sensor recordings for that activity.	As a logged in user, I want to click on a default activity type to start sensor recordings for that activity.	**Scenario:** User is navigated to the main page. **Given:** That the user has successfully logged in. **Then:** The user should be navigated to the main page where they are presented with a list of activities they can tap on to start the sensor recordings for that activity.	✓		
18	If the activity users wish to have sensor recordings for doesn't already exist, the users should be able to define a new activity type to be used for sensor recordings.	As a logged in user, I want to click on "Record New Activity" to create a new activity type to start sensor recordings for that activity.	**Scenario:** User is navigated to the main page. **Given:** That the user has successfully logged in. **Then:** The user should be navigated to the main page where they are presented with a list of activities they can tap on to start the sensor recordings for that activity. If this list doesn't contain the activity type they wish to do a sensor recording for, the user should have the option to create a new activity type by clicking on the "Record New Activity" button.		✓	
19	Upon clicking an activity, the user should be directed to a new page where they can see the activity type they selected, instructions to start sensor recordings and the duration of the recording session in seconds.	As a logged in user, I want to be directed to an activity detail page after clicking on an activity which shows the activity type I	**Scenario:** The user is in the process of starting an activity recording session. **Given:** The user has tapped on an activity type. **Then:** The user should be directed to an activity details page where they are able to see the activity type they selected, instructions to start sensor	✓		

		selected, instructions to start sensor recordings, and the duration of the recording session in seconds so that I can easily start my activity recording session after double checking I am recording data for the activity type I intended to record.	recordings and the duration of the recording session in seconds.			
20	If the user attempts to start a recording without opening the companion app on the apple watch, the user should be presented with an alert that reminds them to open the watch app.	As a user who will start a data collection session, I want to be reminded if I forgot to open the watch app so that I can easily troubleshoot any issues related to the connectivity of the companion watch app.	**Scenario:** The user is in the process of starting an activity recording session. **Given:** That the user has either not opened the watch app or the apple watch isn't connected to the iPhone. **Then:** The user should be presented with an alert that notes they have either not opened the watch app or the apple watch isn't connected to the iphone.		✓	
21	During the session recording, the remaining time in the session duration should be counted down.	As a user who is in a data collection session, I want to be able to see the remaining time in the session duration so that I can increase my situational awareness about the session's status.	**Scenario:** The user is in a data collection process. **Given:** That the user has clicked on the "Start Activity" button on the iPhone after connecting the apple watch and opening the watch app. **Then:** The user should be able to see the remaining time of the data collection session on the iphone.		✓	
22	The user must be presented with the opportunity to end the session early.	As a user who is in a data collection session, I want to be able to end the session before the time runs out	**Scenario:** The user is in a data collection process. **Given:** That the user has clicked on the "Stop Activity" button. **Then:** The user should receive the data up until the point they have tapped "Stop Activity".	✓		

		in my session so that I have more control over my data collection process.				
23	After a session has ended, the time should be reset and the system should be ready for another session recording.	As a researcher, I want to be able to do multiple data collection sessions so that I can obtain sufficient data for my research.	**Scenario:** The data collection process has ended. **Given:** That the user doesn't navigate away from the activity detail page. **Then:** I should be able to tap on "Start Activity" to start another data collection session.	✓		
24	On the apple watch app, the user should be easily able to see whether an activity recording session is in progress or whether the system is idle.	As a user, I want to be able to see whether an activity recording session is in progress on the apple watch app so that I can increase my situational awareness.	**Scenario:** The user interacts with the apple watch app. **Given:** That the watch app is in the foreground. **Then:** The user should be able to see at the top of the screen either "Recording" or "Standby" which indicates an active data collection session and an idle app respectively.	✓		
25	The user should also be able to see sensor readings on the apple watch while an activity is being recorded.	As a user who is in a data collection session, I want to be able to see sensor readings on the apple watch app so that I can increase my situational awareness.	**Scenario:** The user is in a data collection session. **Given:** That the watch app is in the foreground. **Then:** The user should be able to see the values produced by the sensors	✓		
26	Additionally, the user should be able to use the watch app directly instead of interacting with the iOS app to start or end the session.	As a user who is about to start a data collection session, I want to be able to start the session directly on the apple watch so that I have more control over the data collection session when the phone is away.	**Scenario:** The user is interacting with the apple watch app. **Given:** That the watch app is in the foreground. **Then:** The user should be able to start the data collection session by tapping on "Start Workout" button at the bottom of the page	✓		
27		As a user who is	**Scenario:** The user is interacting	✓		

		in a data collection session, I want to be able to stop the session directly on the apple watch so that I have more control over the data collection session when the phone is away.	with the apple watch app. **Given:** That the watch app is in the foreground. **Then:** The user should be able to stop the data collection session by tapping on “Stop Workout” button at the bottom of the page			
28	On the iOS app, the user should be able to change their username, and turn off or on various sensors that are used to record data.	As a user, I want to be able to change my username so that I can correct any mistakes during sign up process with ease	**Scenario:** The user is changing their preferences **Given:** That the user is in the Account & Preferences tab **Then:** The user should be able to see and change their personal information.	✓		
29		As a researcher, I want to be able to determine which sensors are used for data recording so that I can tailor the data collection session to suit my research needs.	**Scenario:** The user is changing their preferences **Given:** That the user is in the Account & Preferences tab **Then:** The user should be able to toggle select which sensors will be used for activity collection.	✓		
30	The user should also have the control to change the duration of a single session.	As a researcher, I want to be able to change the duration of a data collection session so that I can tailor the data collection session to suit my research needs.	**Scenario:** The user is changing their preferences **Given:** That the user is in the Account & Preferences tab **Then:** The user should be able to change the session duration with a slider	✓		

5. Software Engineering Process

5.1 Software Engineering Process Description

Since the articulation of the agile manifesto, agile software development has received global attention [15]. The agile manifesto values individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [16]. Agile software development is implemented through various frameworks such as Scrum, Extreme Programming, and Kanban. For my project, I have used a variant of the Scrum framework.

In Scrum, software development is divided into short development cycles called sprints which typically lasts 2 weeks. In addition to the development team, a Scrum team typically contains a Product Owner as well as a Scrum Master. The product owner represents the customer and is responsible for maximising the value of the product created by prioritising which user stories should be implemented first. On the other hand, the scrum master facilitates the scrum process by sprint planning, sprint reviews and stand up meetings. In short, sprint planning is the process of identifying the user stories to be completed during the sprint, stand-up meetings are short daily meetings where developers state what they have done so far in the sprint and what they aim to do that day which increases the efficiency of developers. These meetings also provide the opportunity for any small issues to be discussed which prevents small issues from turning to huge issues that may threaten the project timeline. Lastly, sprint reviews allow the team to discuss what was done during the sprint, how the efficiency of the team can be increased, and any uncompleted sprint tasks which need to be completed in the next sprint.
Moreover, I have used the Attribute-Driven Design (ADD) method which is an iterative method that helps the software developer to identify the architecture of the software system [17]. ADD produces a set of sketches of architectural views which I have used to identify the collection of architectural elements and their interactions [17].

5.2 Software Engineering Process Justification

Agile software development typically focuses on an agile team [15]. However, being agile is more than simply following a collection of practices [15]. Being agile is a philosophy that values flexibility, adaptability, coordination, incremental development, and continuous testing throughout the software development lifecycle. These are important values to develop a software system that hasn't been previously developed in an open sourced manner.

Specifically, flexibility and adaptability values were crucial for this project as I had to experiment with many Swift coding frameworks to arrive at an optimal coding solution. Moreover, I had to coordinate with my supervisor, who acted as the product owner, in order to maximize the value this application will generate for machine learning researchers. Lastly, I wasn't able to leverage the past experiences of other developers to develop the system while I was developing my code. Therefore, continuous testing and incremental development enabled robustness and correctness of my software solutions. Since scrum provides a framework that encapsulates discussed values, I used a scrum engineering process where I acted as both the development team and the scrum master.

Furthermore, using the attribute driven design method enabled me to capture the architecture of the system at different levels of abstraction which I have used to guide my software design process. Although there might appear to be an inherent tension between being agile and up-front architecture practices, the underlying philosophies are not at odds and can be married to great effect [17]. It has been demonstrated that successful software projects need a blend of the two approaches as too much up-front planning and commitment can be unresponsive to customers' needs, whereas too much agility can simply result in chaos [17]. Boehm and Turner, analyzing historical data from 161 industrial projects, examined the effects of up-front architecture and risk resolution efforts. They found that projects tend to have a "sweet spot" where some up-front architecture planning pays off and is not wasteful [17]. In this project, I have initially used the up front architecture, which will be further discussed in the design section of this paper and refactored the architecture in an agile manner to arrive at the sweet spot.

6. Ethics

Ethical approval to perform the user study was granted by the University of St Andrews School Ethics Committee. The approval letter can be found in Appendix C. The research does not raise any major ethical concerns.

7. System Design

In this chapter, we focus on the ideas of the design of the software system and unusual design features. The software system was designed using the attribute driven design (ADD) methodology. ADD provides a framework to capture architectural requirements and iteratively make design decisions to address these requirements. ADD process is executed as follows: Part of the system is chosen to design, the architecturally significant requirements (ASR) for that part are gathered, and a design pattern that can be used to implement that part of the system with respect to relevant architectural requirements is selected, and finally the design solution is refactored. These four steps are repeated until all parts of the system are implemented. The attribute driven design process I carried out can be found in Appendix A.

Architecturally significant requirements are the main drivers of the ADD process as they are used to evaluate whether an architectural pattern can be applicable to a software system. In chapter 4, the functional requirements that were marked to have a high architectural impact partially form the architecturally significant requirements. In addition, the ASRs also encapsulate contextual design constraints of the system, and quality attributes. In the following sections, we introduce the contextual design constraints and the quality attributes of our system, why our chosen design solution is best suited to address ASRs, and our novel technique for implementing the design pattern.

7.1 Design Constraints

A design constraint is a design decision with zero degrees of freedom [15]. That is, it’s a design decision that’s already been made [15]. We can classify design decisions in two categories. Technical constraints that originate from the development context of the system and project management constraints that originate from the project timeline and project context of the system.

7.1.1 Technical Constraints

(D1) - The application will be a mobile system.
(D2) - The application will be implemented with Swift and SwiftUI.
(D3) - Sensor data must originate from a wearable device.

7.1.2 Project Management constraints

(D4) - The project implementation has to be cost-effective.
(D5) - The project will be implemented by a single developer.
(D6) - The project is heavily time-constrained.
(D7) - The application must be easily usable.

7.2 Quality Attribute Requirements

Quality attributes are qualifications of the functional requirements or of the overall product [15]. A qualification of a functional requirement may describe how fast the function must be performed, or how resilient it must be to erroneous input [15]. A qualification of the overall product may describe how secure a system should be, or how it should scale up or down.

(Q1) - Usability: The system is developed with the end goal of establishing a standardized database . Therefore, an easy to use user interface is crucial to incentivize more researchers to use the system.
(Q2) - Interoperability: The system needs to be able to work with other systems as it may be extended in the future to use web APIs in order to send data to ML models
(Q3) - Reliability: The system must provide users with dependable data as ML models only perform as well as the data's quality.

7.3 Model, View, View Model (MVVM) Design Pattern

The MVVM pattern proved to be the most suitable design pattern to address systems functional requirements, design constraints, and quality attribute requirements after the ADD process was completed. The main idea of the MVVM pattern is to cleanly separate an application's business and presentation logic from its user interface (UI) [18]. The pattern's working principle which was inspired from Microsoft's definition is demonstrated in figure 3.

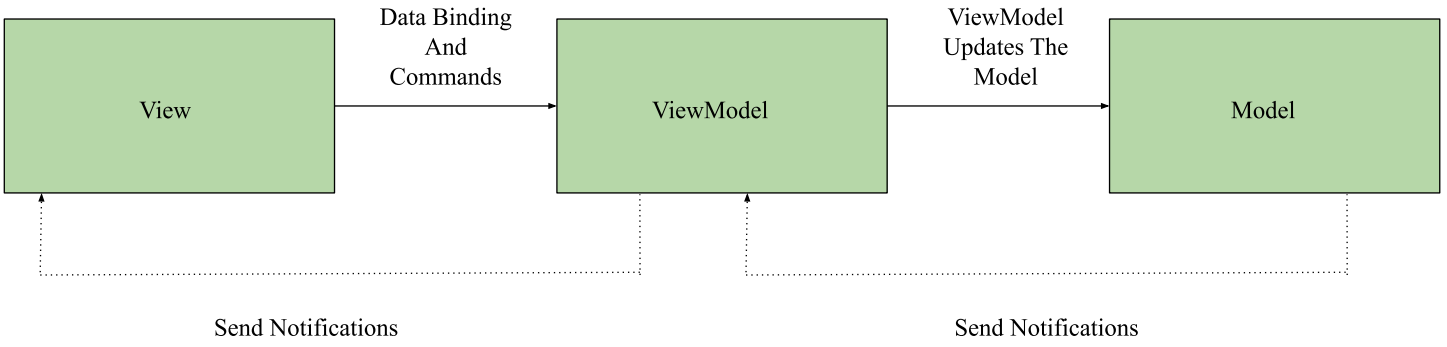


Figure 3: MVVM Working Principle

In the MVVM pattern, the view which contains the UI is aware of the ViewModel and the ViewModel which contains business logic is aware of the Model [18]. When a user interacts with the UI, the view binds. The model which contains the app's data is unaware of the view model, and the ViewModel is unaware of the view [18].
The ideas of using the MVVM pattern and which system requirements they address can be found in the table below:

Requirement ID	MVVM Ideas
Q2,Q3,D3	MVVM enforces a separation of concerns with distinct roles for the Model, View, and ViewModel components which makes the codebase more maintainable [19].
Q1,Q3	Developers can create unit tests for the view model and the model, without using the view [18].

Q1,D6	The app UI can be redesigned without touching the view model and model code which makes frontend development more agile and able to respond to customer feedback [18].
Q3,D4,D5,D6	The ViewModel can be used for multiple views which increases reusability [18].
Q2,Q3,D3	The MVVM design pattern has been shown to complement scalability [19].
D1,D2	The MVVM pattern is applicable to mobile applications.
F01	An application with an MVVM pattern has better interoperability with other APIs

7.4 Our Approach To Implementing the MVVM Design Pattern

Two applications were developed to implement the software system. An iOS application that runs on the iPhone and a complementary WatchOS application that runs on the Apple Watch. Both applications follow the MVVM pattern. I have tailored the Model, View, and the ModelView components in both applications to better suit my system needs. The main app is the iOS application which the user primarily interacts with. The design details and ideas are elucidated below.

7.4.1 View Class

The view is responsible for defining the structure, layout, and appearance of what the user sees on screen [18]. An unusual design feature of my system when compared to typical MVVM systems is the implementation of a central view. The central view is implemented in the MainPageView class which is the root view of the iOS app. It doesn't only contain UI elements and their structure, but also manages the navigation between different view classes. In most MVVM implementations, this navigation would be delegated to the ModelView component where view navigation is performed using imperative methods which are considered business logic. However, my approach naturally complements SwiftUI's declarative principle where the navigation is handled with a data driven API. That is to say that a navigation link is bound to a data type so that when the user interface has multiple UI elements, which are expressed as different data types, the navigation to other view classes can be handled based on what type of a UI element was interacted with rather than imperative methods.

Furthermore, the view classes contain some business logic if that business logic pertains to a pop-up element such as a sheet or an alert. Ordinarily, the MVVM pattern forbids the implementation of any business logic in the view classes, however, these pop up elements are too atomic to be implemented as a separate view. Therefore, these elements were treated as simple UI components such as buttons. In SwiftUI, simple UI components naturally contain some business logic as they directly support commands which can't be separated to a ViewModel. The choice to not separate the business logic of pop-up elements was done to make the code more readable by having harmony between all UI elements which makes debugging easier to do.

Lastly, another important design feature of the view classes is the organization of the code itself. The MVVM pattern's benefits can be maximized when the code is as compartmentalized as possible. More compartmentalization enhances the benefits of separation of concern as it makes the code easier to modify, more reusable, more readable, and more testable. Therefore, I have refactored my code so that sub elements of a UI such as buttons, activity icons, sheets, or grid views are contained in different structs which is similar to C programming language's structures.

7.4.2 ViewModel Class

The view model implements properties and commands to which the view can data bind to, and notifies the view of any state changes through change notification events [18]. The view model is also responsible for coordinating the view's interactions with any model classes that are required [18]. Contrary to view classes, there is only one ModelView class per application.

Unlike a mobile game, the UI of the app I developed does not respond to frequent user inputs. In a way, the UI can be thought to be less stateful which doesn't require frequent or complex commands to be raised. Therefore, the primary function of my view model class (which can be found under the filename ActivitiesViewModel in my code) is coordinating the view's interaction with model classes. This leads to the design of the view model encapsulating more complex properties than commands.

The most important model class that the view needs to coordinate with is the SensorModel class which contains sensor data. However, this data is generated by the sensors on the apple watch. To facilitate the generation and the access to data, the view model contains a WatchConnector property. Inline with the MVVM pattern, the WatchConnector's functionalities such as sending a command to the watch to start data recording or retrieve collected data is exposed to the view through the view model class. While this object is implemented in another file to make the code easier to understand, WatchConnector is a property of the ActivitiesViewModel class.

Finally, the ActivitiesViewModel class also contains a data conversion functionality where the collected data is made available to the view in a CSV format. In normal MVVM implementations, data conversions are centralized in a data conversion layer [18]. However, I have implemented this functionality into my view model class because the data that needs conversion is introduced to the app through the app's view model class thanks to its WatchConnector property. This leads to a more natural flow of data because of code colocation.

7.4.3 Model Class

Model classes are non-visual classes that encapsulate the app's data [18]. In my code, model classes end with the "Model" word. Model classes are typically used in conjunction with services or repositories that encapsulate data access and caching [18]. In our system, this service is offered by the ViewModel class.

Figure 4 shows the component diagram which describes the structure of the software system.

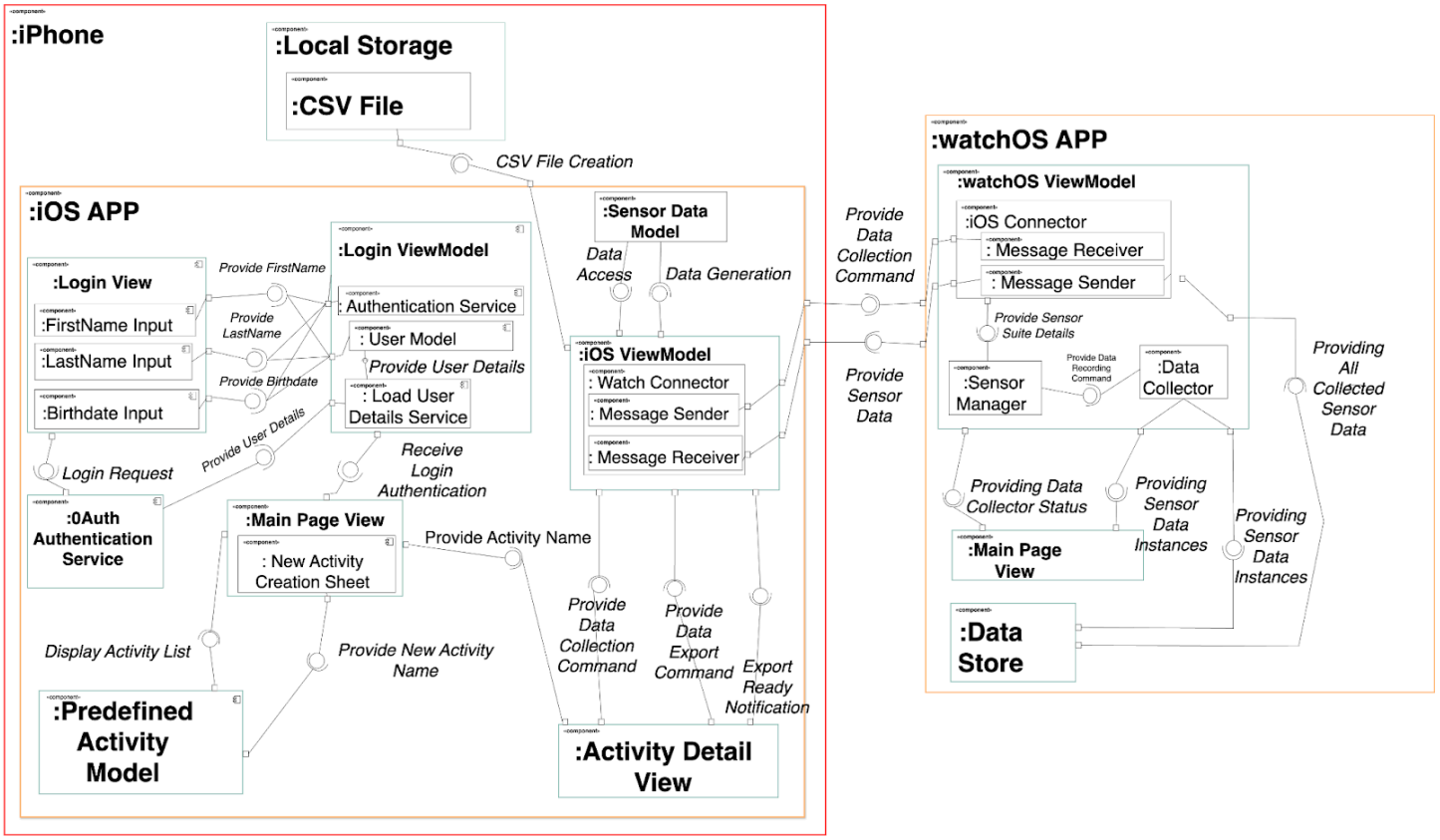


Figure 4: Component Diagram

8. Implementation and Testing

SensoGather is an open-source data collection app coded with the Swift and SwiftUI programming languages, split among two main repositories:

- HAR APP which contains code classes for the mobile application
- WatchHARApp Watch App which contains code classes for the companion apple watch application

This chapter provides the details for how the implementation of these applications was done and tested, with particular focus on important and novel algorithms, unusual implementation decisions, and novel user interface features.

8.1 Programming Language Choice and Target Hardware System

The system was implemented using Swift programming language and SwiftUI framework. The choice to implement the application with Swift and SwiftUI was made because of the target hardware.

8.1.1 Target Hardware

Upon considering the challenges faced in the field, this project aimed to address data quality issues caused by the sensing equipment which prevents reproducible research and deployable HAR systems.

As mentioned in the introduction, HAR software systems need to be deployed as mobile applications in order to unobtrusively capture and analyze data. Mobile software systems can be categorized into Android applications and iOS applications.

Android applications run on mobile devices that run on the open sourced Android operating system [21]. Android operating system enables different manufacturers to customize the operating system so that companies can offer a tailored experience to users [21]. While the open source nature of this operating system prevents monopolization by disallowing any central point of failure in which one industry player can restrict or control the innovations of any other player, the need to be adaptable for different hardware systems aggravate the issue of insufficient data quality due to the variability in sensor characteristics and OS behaviour. This operating system is used by many different manufacturers who outsource their inertial sensors from different providers. Furthermore, different manufacturers purchase sensors that greatly differ in quality and price bracket in order to stay competitive by minimizing their production cost. Consequently, it is infeasible to determine the characteristics of a sensing equipment and it is impossible to do any uncertainty analysis of the data that originates from an android device.

On the other hand, the iOS operating system is only used by Apple for their iPhone devices. The homogeneity of devices used to obtain data and the uncustomizable operating system enables a DL model to better extract data features from sensor data as sensor characteristics will be fixed due to stable data source. Moreover, an uncertainty analysis of data can be carried out by running a user study that compares the data obtained from an iOS application and data that originates from a scientific sensor for the same activity.

Furthermore, in HAR applications the placement of the mobile device significantly changes the presentation of data features [3]. Some individuals carry their mobile devices in their purses, some in their back pockets, some in their front pockets, and some in their coat pockets. This obstructs the deployability of a HAR system to a real

world setting. Researchers have proposed complex data preprocessing and machine learning procedures to handle this data presentation differentiation [3].

This issue can be addressed if the data originates from a wearable device such as an Apple Watch as these devices are predominantly worn on the user's left wrist. This minimizes the effect of sensor location on the HAR system's performance. Additionally, wearable devices are less prone to accidental dropping which minimizes the degradation of sensor quality.

Consequently we chose to implement our system as an iOS application which has a companion watchOS application that is coded using Swift and SwiftUI programming languages which minimizes the cost of diversity in sensor characteristics, OS level issues, sensor quality degradation, and sensor location impact on the overall performance and deployability of a HAR system.

8.2 User Interface

The user interface is specifically designed to increase usability, complement functionality and minimize user interaction flows that may introduce unwanted behaviour.

The user interface was developed by following Apple's human interface guidelines (HIG) in order to follow best practices. The iOS application is organized under 2 tabs: Home Tab and Account & Preferences Tab. The watchOS app has a scrollable single page interface.

8.2.1 iOS App Navigation

The user interface was designed to be as minimalistic as possible so that the user doesn't unnecessarily need to navigate around the app which increases the learning curve and decreases the ease of use.

The mobile application has three user interface views which are organized under two tabs. The active tab is highlighted in order to increase the navigational awareness of the user.

Two out of three views are accessed by tapping on the tabs. The only view that is not accessible is the ActivityDetailView which is the page that the user interacts with to start a data collection session. This page is accessed by tapping on an activity type on the home tab.

The tabs are always visible in all cases where the application is running on the forefront of the mobile device apart from the login process and when a data collection session is ongoing. This enables users to intuitively navigate around the app and quickly change data collection settings such as which sensors are selected or the duration of the data collection session in between recordings. More importantly, we discovered in our testing that if a user navigates away from an activity detail page without saving the collected data, this can introduce unwanted behaviour such as the saved data being recorded under false labels. This could happen if a user record data for basketball, navigates to the home menu, and taps on walking and clicks export. In this case the collected data for basketball will be saved as walking. The UI disallows this by not displaying any navigation buttons including tab buttons until after the user saves their data. In app images displaying this functionality can be found in figure 5 below.

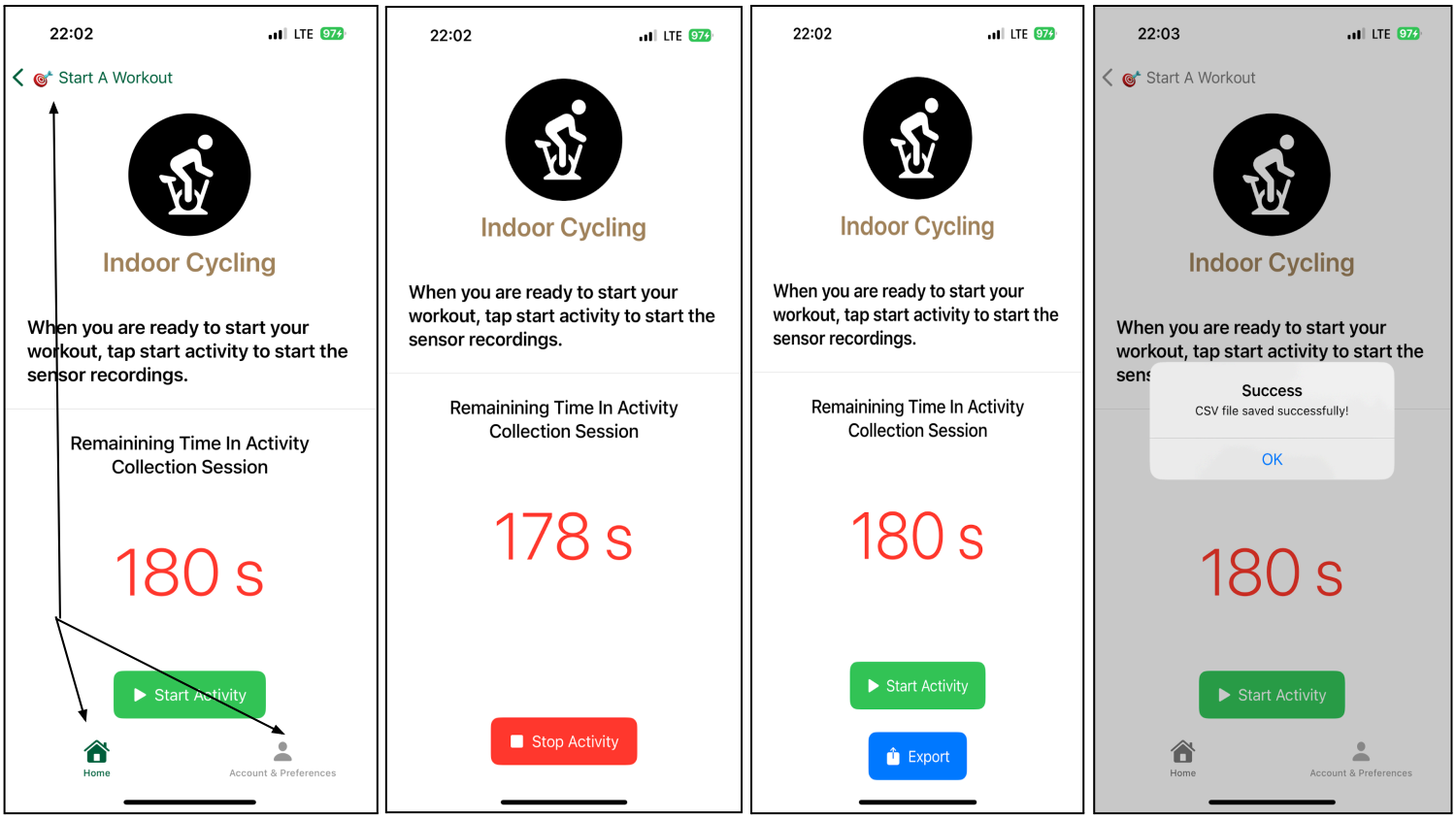


Figure 5: UI Functionality Demonstration: In app image (a) shows the view before a data collection session starts tab buttons at the bottom and navigation button at top left corner which all can be used to navigate away from the view. (b) shows the view during a data collection session where navigation buttons are hidden. (c) shows the view after a data collection session where the data is ready to be exported. Note that the navigation buttons are still hidden. (d) shows the view after the data is exported. Note that the navigation buttons are visible again.

8.2.2 iOS App Home Tab

Upon login, the user is guided to the home tab. The home tab has green accents in order to create a welcoming feel [22]. This home tab which is implemented in the MainPageView tab contains 9 activities that the user can tap on to start a data collection session. These activities are among the most common activities researchers are currently developing activity recognition models for which were determined following the analysis of other publicly available datasets and literature review.

At the top of the main page, a button is present to add a new activity type. This enables researchers to record data for any activity they wish. Tapping on the button opens a sheet view that has instructions on how to save a new activity type. The sheet has an explicit “X” to close the sheet view even though a “flicking down” gesture would get rid of the sheet. This is a best practice that was described in the HIG documents. The sheet view contains instructions to increase the usability of the app by putting instructions wherever users might need guidance. Figure 6 contains in app images that demonstrate the process of adding a Badminton activity type.

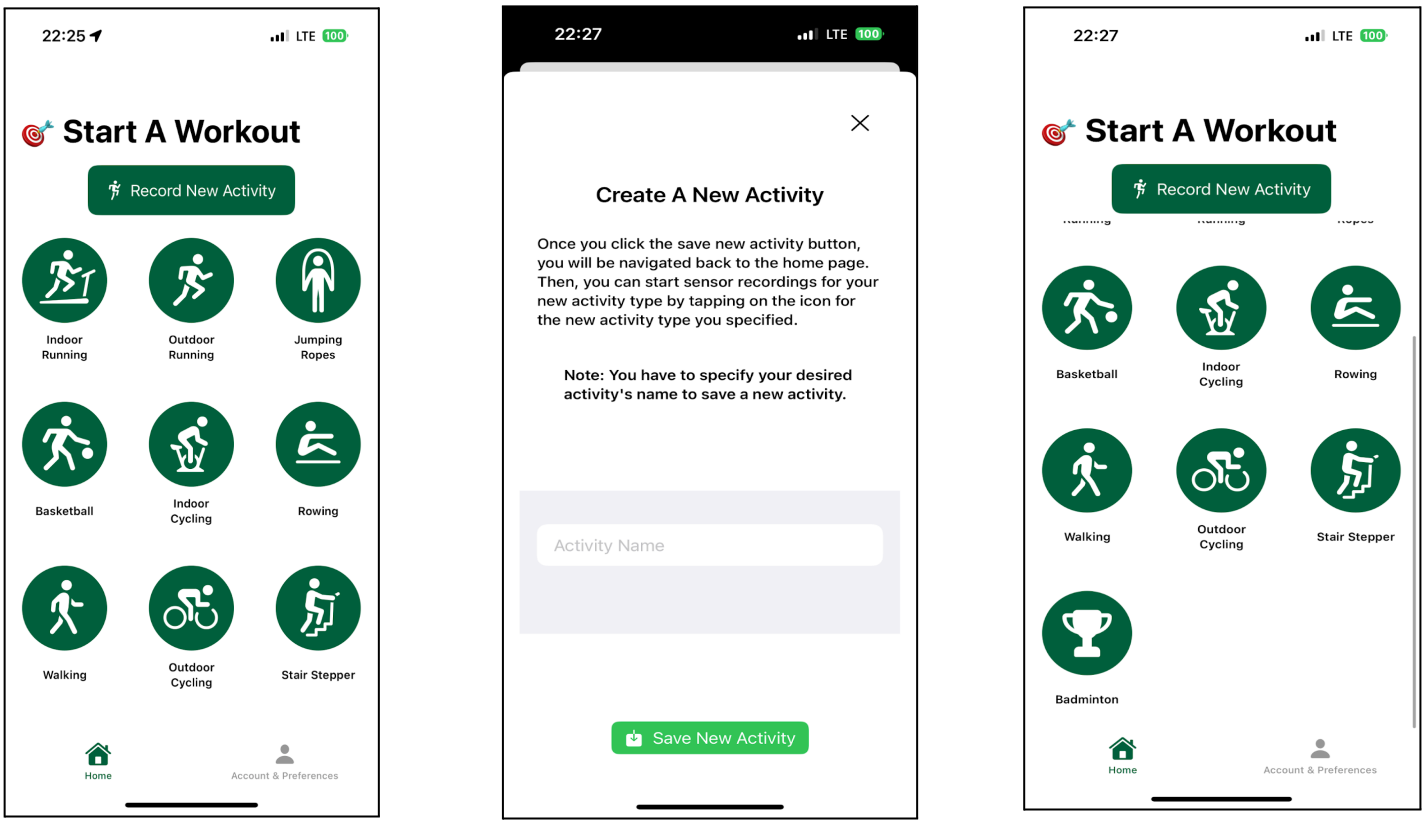


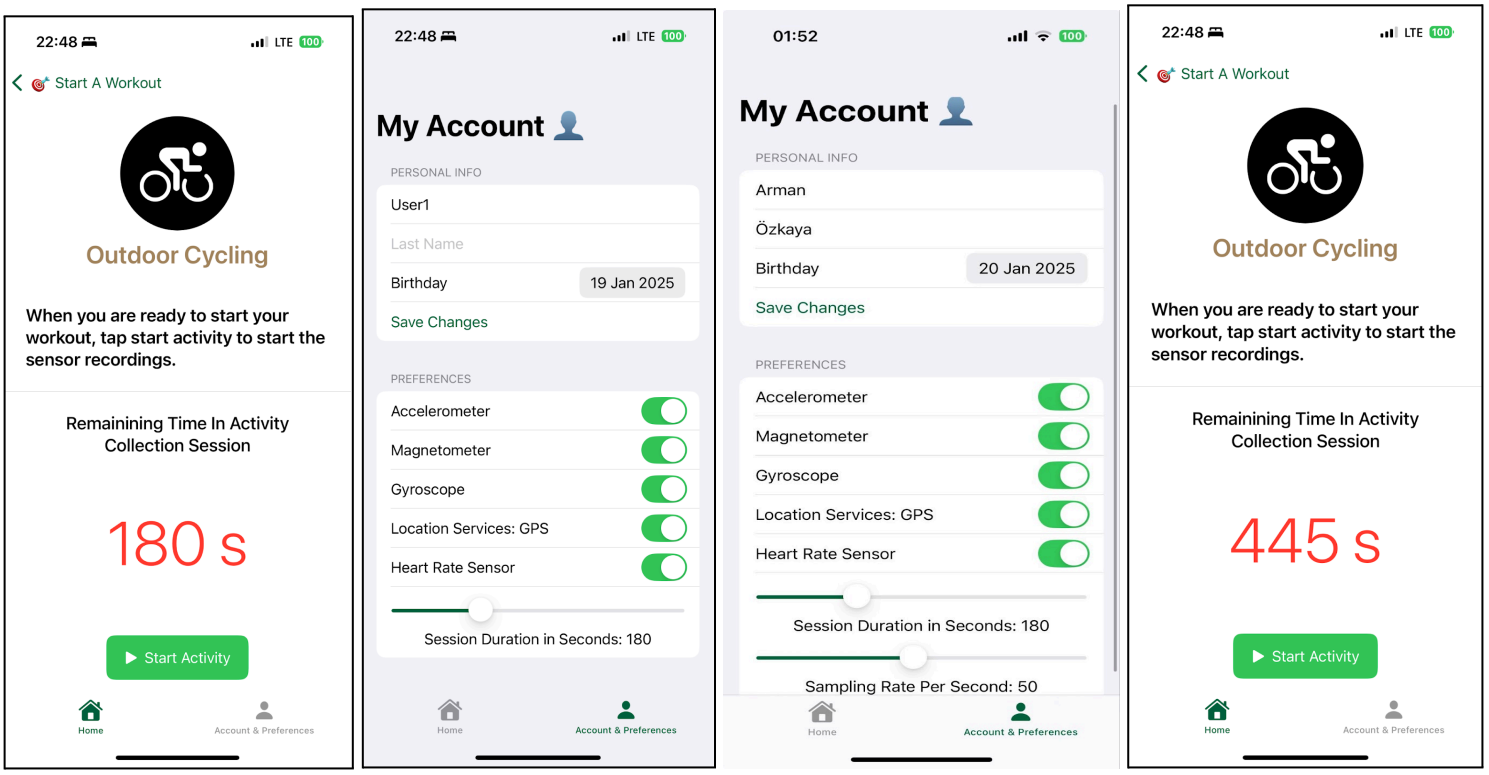
Figure 6: Activity Addition Demonstration

8.2.3 iOS App Account & Preferences Tab

Account and preferences tab enables a user to change their user information, change which sensors are used for recording, change the duration of the data collection, and change the sampling of inertial sensors. This page was crucial in allowing researchers who might use our app to collect data to tailor the app according to their data collection need. For instance, some researchers might only want to record data for an accelerometer or some researchers might prefer a longer data collection session. Figure 7 contains a demonstration of changing the data collection session's duration.

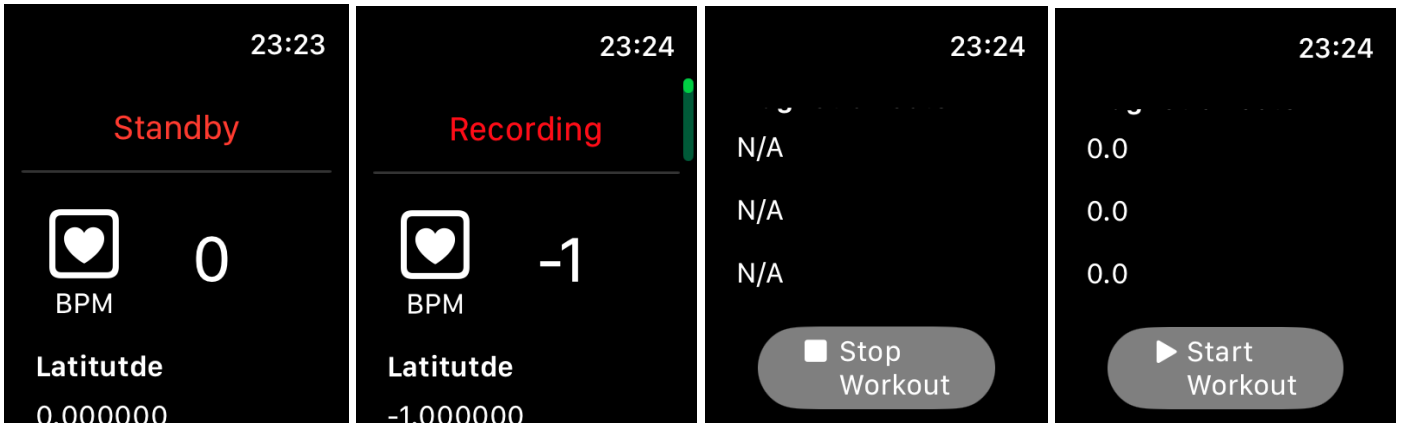
8.2.4 watchOS App UI

The watch app interface was designed so that if the person who is holding the phone and the person who is wearing the watch are different, they can still carry out the basic functionalities of the app directly from the watch. Therefore, the watch view displays the most crucial information and it contains the most crucial commands. Firstly, the view displays whether an activity collection session is in progress or not. Secondly, the view displays the sensor readings. If sensors have been switched off in the Account & Preferences tab, the view shows either N/A or -1 depending on the sensor type. The user also can start a data collection session or stop a data collection session. Figure 8 contains a demonstration of the watch view interface.



(a) (b) (c) (d)

Figure 7: Sensor Setting and Data Collection Duration Altering Demonstration



(a) (b) (c) (d)

Figure 8: Apple Watch User Interface Demonstration

8.3 Important Implementation Decisions

This section provides a discussion of important implementation decisions.

8.3.1 Obtaining Data

In my application, I chose to use the Core Motion framework to obtain inertial sensor data, Core Location to obtain gps data, and healthkit to obtain heart rate data. The implementation can be found in the ViewModel class of the watch app.

Some of the important implementation decisions were choosing the parameters of these frameworks. To begin with, when one starts a service that reports the device's attitude in three-dimensional space, Core Motion establishes a frame of reference for reporting pitch, roll, and yaw values [23]. All subsequent data values specify the device attitude relative to this frame of reference [23]. I chose to use CMAttitudeReferenceFrame.xTrueNorthZVertical as my reference frame as true north is an absolute reference point compared to arbitrary reference points which helps produce reproducible data.

Furthermore, another crucial decision was choosing between obtaining raw data or processed data. After consulting my supervisor, I decided that using the CMDeviceMotion framework to obtain processed data rather than raw data is better suited to my project needs. While the raw data is useful, it sometimes contains additional information which needs to be processed. For example, the raw accelerometer data contains both the acceleration caused by gravity and by the device's motion, and gravitational acceleration must be removed from the raw data values to obtain device acceleration [23]. Therefore, we chose to obtain processed data to make preprocessing data easier for when this data is used to train a DL model.

8.3.2 Temporarily Storing Data On The Apple Watch

In early development stages, I have stored all sensor data in an array. However, I noticed in testing that the memory allocated to the array data structure is not enough when the session duration exceeds 2 minutes. In order to solve this issue, I used Swift Data to commit all recorded sensor data locally to the apple watch rather than appending new data to an array data structure. When the data needs to be transferred to the iOS device, this data is fetched from the local device storage.

8.3.3 Transferring Data From The Apple Watch To The iPhone

Throughout the development stage, the most challenging task was reliable data transfer which includes sending commands to the watch to start or stop recording sensor data and transferring sensor data from the apple watch to the iPhone. The complication arose from the fact that my app is unlike any open sourced project that has a watch connectivity function which required a significant amount of trial and error in order to arrive at a solution that reliably operates.

This section discusses the methodologies that were tried, details why these methodologies were not suitable, solutions that have been attempted to remedy the issues, and describes the correct solution so that future work can take this document as a guide to correctly implement watch connectivity.

In order to do the data transfer between devices I have used the Watch Connectivity framework. This framework has multiple methods of data exchange. One of these methods is the instant data exchange method and the other method is the file transfer method.

Instant data exchange sends a message immediately to the paired device. Initially, I used this methodology so that every time a sensor produces data on the watch, it can be immediately sent to the iOS device eliminating the need to do any data handling on the watch device. The instant data exchange method requires the watch to be in an “available” state. During testing, I have discovered that the instant messaging functionality is disabled when the user naturally lowers their wrist because the watch app transitions from an “available” to a “background” state. The background state should not be confused with the watch device going to sleep. In this state, the watch app is still running on the wearable device, however, the app transitions to run in the background which can only be observed as the watch screen slightly dimming itself. This is an OS induced behaviour, presumably to save energy.

In order to programmatically solve this issue, I have implemented extended runtime sessions which continue running the app even after the user stops interacting with it. However, this solution didn’t resolve the issue. Therefore, I have switched to a different data transfer method to send sensor data from the watch to the iOS device while keeping instant data exchange for sending commands from the iOS device to the apple watch.

At first, switching to the file transfer method which required committing data to the apple watch as discussed in section 8.3.2 solved the issue. File transfer uses background threads to asynchronously transfer JSON files. Transitioning to this method required serialization steps on the apple watch and deserialization steps on the iOS device in order to convert sensor data into JSON format. This implementation can be found in the ModelSensorData class on the Watch App and SensorModel on the iOS app.

However, I have also discovered that the instant message sending, unrelated to watch state, is disabled if the apple watch has a battery level lower than 20%. This was an issue because this meant that I couldn’t get the command to start data recordings across when the apple watch battery is low. Apple watches don’t have large batteries which already limit data collection capability of the system, and this limitation made running a user study extremely limiting. Therefore, I have switched to using file transfer methodology in order to send commands across as well.

While this solved the issues related to instant messaging, this solution introduced buggy behaviour due to the asynchronous behaviour of the file transfer methodology. Specifically, file transfer is a “send and forget” protocol which doesn’t have any acknowledgment functionality. This meant that if a user pressed the start activity button on the iOS device, the iOS device believed there was an active data collection session in progress even if the apple watch was turned off. When the watch is finally turned on and the watch app is opened, the queued file transfer requests would introduce unpredictable behaviour.

I solved this issue with a protocol I devised. In this protocol the start command is sent as a file transfer only if the apple watch app state is active. After the watch receives this command, it sends back a file transfer containing the acknowledgment message. The iOS app checks the metadata of the file transfer in order to decide whether this file transfer contains an acknowledgement or sensor data. When an acknowledgment is received, the iOS app executes methods to implement functionalities of an activity data collection session. The stop command is sent without checking the watch app state because the watch app may have transitioned to the background mode while the user is performing an activity. The stop command also sends an acknowledgement. Finally, the watch app prepares and sends the sensor data through a separate file transfer. The execution graph of the protocol is shown in figure

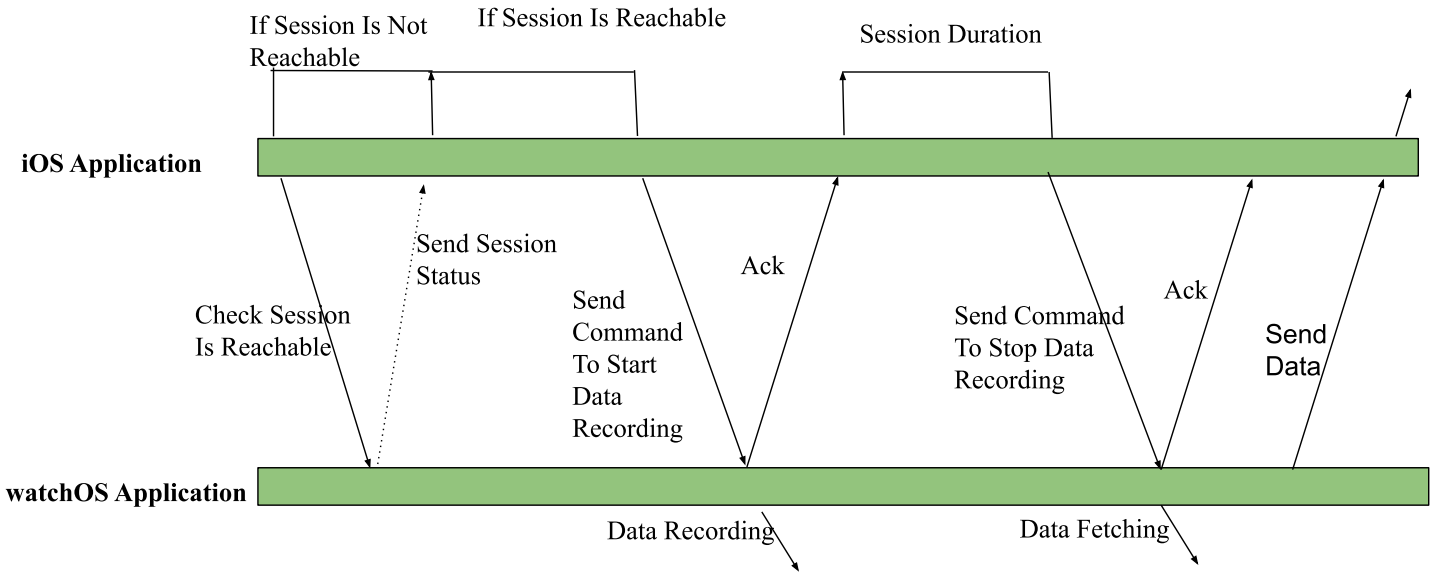


Figure 9: Custom Watch Connectivity Protocol

8.4 Testing

The testing of the app was done with unit tests, manual testing using the user interface, and a beta testing with TestFlight. I have already discussed some of the most important revelations from the testing process in section 8.3.3. In this chapter, we further reveal insights that were obtained and discuss the test cases

8.4.1 Unit Tests

Unit tests are powerful testing methods and I was able to test some parts of my code with unit tests thanks to the MVVM design. I have used unit tests to check the correct implementation of the authentication process and new activity addition process. These tests can be found under the HAR_APP_Unit_Tests file. I implemented a relatively limited number of unit tests because most of the correctness of the app could only be demonstrated with manual testing where I could obtain the produced sensor data and analyze it for correctness.

8.4.2 Manual Testing Using The User Interface

Manual testing was the primary way of testing. For this process, I have systematically written down all possible user interface interactions in a table which can be found below. For these tests, either the watchOS app or the iOS app was run in the debug mode on x code. The changes in User Interface were observed and cross checked with the test cases in the table. Further, test cases that involved a change in the characteristics of produced data such as a change in the sampling rate was observed by analyzing either the console outputs or analyzing the exported CSV file. The outputs verifying the correctness of test cases can be found in Appendix B for all test cases that can be feasibly shown.

Case No	What Case is Being Tested	Pre-Conditions	Expected Outcome	Actual Outcome
1	User is Able to Enter Their First Name	User is in Login Page	User can tap on the “First Name” field, a keyboard should pop up in order to allow the user to enter their first name.	User can tap on the “First Name” field, a keyboard should pop up in order to allow user to enter their first name.
2	User is Able to Enter Their Last Name	User is in Login Page	User can tap on the “Last Name” field, a keyboard should pop up in order to allow the user to enter their last name name.	User can tap on the “Last Name” field, a keyboard should pop up in order to allow the user to enter their last name name.
3	User is Able to Enter Their Birthday	User is in Login Page	User can tap on the “Birthday” field, a date wheel should pop up in order to allow the user to enter their birthday.	User can tap on the “Birthday” field, a date wheel should pop up in order to allow the user to enter their birthday.
4	User Can Log In	User is in Login Page and the mandatory fields are selected	User can tap on the “Login” button in order to be logged in and directed to the main page.	User can tap on the “Login” button in order to be logged in and directed to the main page.
5	User Can Log In	User is in Login Page	User can tap on Sign in with Apple button to log in using their apple credentials and they will be directed to the main page.	User can tap on Sign in with Apple button to log in using their apple credentials and they will be directed to the main page.
6	Login Testing: No Empty Field	User is in Login Page and some mandatory fields are empty	An alert is displayed noting that the first name isn’t entered.	An alert is displayed noting that the first name isn’t entered.
7	Login Testing: User is less than 18 years of age	User is in Login Page and the birthday entered shows that user is less than 18 years of age	An alert is displayed noting that the user can not use this app because they are less than 18 years of age.	An alert is displayed noting that the user can not use this app because they are less than 18 years of age.
8	User Can Tap On An Activity	User is logged in	User can tap on an activity on the main page and they will be directed to the activity detail view.	User can tap on an activity on the main page and they will be directed to the activity detail view.
9	User Can Tap On the Record New Activity Button	User is logged in	User can tap on the record new activity button and they will be directed to a sheet where they can input a new activity name.	User can tap on the record new activity button and they will be directed to a sheet where they can input a new activity name.
10	User Can Enter A New Activity Name	User is in the pop up sheet after tapping record new activity button	User can tap on the “Activity Name” field, a keyboard should pop up in order to allow the user to enter the new activity name.	User can tap on the “Activity Name” field, a keyboard should pop up in order to allow the user to enter the new activity name.
11	User Can Save The New Activity Name	User is in the pop up sheet after tapping record new	User can tap on the “Save New Activity” button to be navigated back to the main page where a	User can tap on the “Save New Activity” button to be navigated back to the main page where a new trophy

		activity button, and the user fills in the activity name field	new trophy icon at the bottom is visible with the name of the new activity.	icon at the bottom is visible with the name of the new activity.
12	Reject new activity saving	User is in the pop up sheet after tapping record new activity button, and the user doesn't fill in the activity name field	User can tap on the Save New Activity button, but this won't have any consequences.	User can tap on the Save New Activity button, but this won't have any consequences.
13	Change First Name	User is in Account & Preferences Tab	User can tap on their first name to change their first name if they tap on save changes after they make the change.	User can tap on their first name to change their first name if they tap on save changes after they make the change.
14	Change Last Name	User is in Account & Preferences Tab	User can tap on their last name to change their last name if they tap on save changes after they make the change.	User can tap on their last name to change their last name if they tap on save changes after they make the change.
15	Don't change User Details	User is in Account & Preferences Tab	If users don't tap on save changes, their modifications won't be saved.	If users don't tap on save changes, their modifications won't be saved.
16	Change Session Duration	User is in Account & Preferences Tab	Users can use the slider in the Account & Preferences tab to change the session duration.	Users can use the slider in the Account & Preferences tab to change the session duration.
17	Change Sampling Rate Per Second	User is in Account & Preferences Tab	Users can use the slider in the Account & Preferences tab to change the sampling rate.	Users can use the slider in the Account & Preferences tab to change the sampling rate.
18	Toggle accelerometer sensor	User is in Account & Preferences Tab	Users can use the slider in the Account & Preferences tab to change the sampling rate.	Users can use the slider in the Account & Preferences tab to change the sampling rate.
19	Toggle magnetometer sensor	User is in Account & Preferences Tab	Users can toggle the accelerometer off to stop the accelerometer sensor from gathering data.	Users can toggle the accelerometer off to stop the accelerometer sensor from gathering data.
20	Toggle gyroscope sensor	User is in Account & Preferences Tab	Users can toggle the magnetometer off to stop the magnetometer sensor from gathering data.	Users can toggle the magnetometer off to stop the magnetometer sensor from gathering data.
21	Toggle GPS sensor	User is in Account & Preferences Tab	Users can toggle the GPS off to stop the GPS sensor from gathering data.	Users can toggle the GPS off to stop the GPS sensor from gathering data.
22	Toggle Heart Rate Sensor	User is in Account & Preferences Tab	Users can toggle the heart rate sensor off to stop the heart rate sensor from gathering data.	Users can toggle the heart rate sensor off to stop the heart rate sensor from gathering data.
23	Reject Starting Data Collection	User is in Activity Detail View and Watch App isn't	Users are presented with an alert notifying them that the apple watch isn't connected or the watch	Users are presented with an alert notifying them that the apple watch isn't connected or the watch app isn't

		active	app isn't opened.	opened.
24	Start Data Collection	User is in Activity Detail View and Watch App is active	Users can tap on "Start Activity" to start data collection, the navigation buttons will disappear, button will change to "Stop Activity" and timer will start counting down.	Users can tap on "Start Activity" to start data collection, the navigation buttons will disappear, button will change to "Stop Activity" and timer will start counting down.
25	Stop Data Collection	User is in Activity Detail View and Watch App is active	Users can tap on "Stop Activity" to stop data collection, the button will change to "Start Activity" and the timer will reset. An export button will become visible when data is ready to be exported.	Users can tap on "Stop Activity" to stop data collection,, button will change to "Start Activity" and timer will reset. An export button will become visible when data is ready to be exported.
26	Stop Data Collection	User is in Activity Detail View and Watch App is active	Users can wait until time hits 0, the button will change to "Start Activity" and the timer will reset. An export button will become visible when data is ready to be exported.	Users can wait until time hits 0, the button will change to "Start Activity" and the timer will reset. An export button will become visible when data is ready to be exported.
27	Export Data	User collected data	User can tap on the export button to export the collected data	User can tap on the export button to export the collected data
28	View Data	User exported data	User can go to their local storage to view the collected data	User can go to their local storage to view the collected data

8.4.3 Beta Testing With TestFlight

Beta testing was crucial in learning about the issues pertaining to the watch connectivity framework. To do this beta testing, I published my app on the TestFlight framework. Nine students from St Andrews University were recruited to be testers. Testers downloaded the TestFlight app on their iphone, which enabled them to download my app on their iPhone and Apple Watch devices. Testers performed six different activities. Each activity was recorded at a sampling rate of 30 Hz and the duration of each recording session was set at three minutes. This was a real user study that was sanctioned by the University.

During beta testing I discovered that when the debug mode in x code is turned on, the watch app is forced to stay active which hides connectivity issues. I solved this issue by developing the protocol I discussed in section 8.3.3.

9. Evaluation

In order to evaluate my app, I have further investigated my app using pandas, used the profiler tool of x code in order to assess the future extensibility of my app, and carried out a user survey to assess usability.

9.1 Pandas Investigation

I investigated my dataset with pandas in order to see whether my app can address data quality issues that other applications suffer from. The small code snippet for pandas was obtained from an online resource. Specifically I wanted to see whether my app suffered from variability in specified sampling rate and actual recorded sampling rate due to OS level issues. Further, I wanted to demonstrate the correctness of my app where when the sampling rate is changed in the user interface, the actual sampling rate of the data is also changed.

To this end, I have recorded an activity for thirty seconds four times. In the first 2 recordings, the sampling rate was at 100 Hz. In the last 2 recordings, the sampling rate was at 30 Hz. The recordings were repeated in order to account for outliers in data. Theoretically, a sensor that has a 100 Hz sampling rate should have produced 3000 recordings if the session duration is 30 seconds, and a sensor that has a 30 Hz sampling rate should have produced 900 recordings in the same session duration of 30 seconds. My app produced the following recordings:

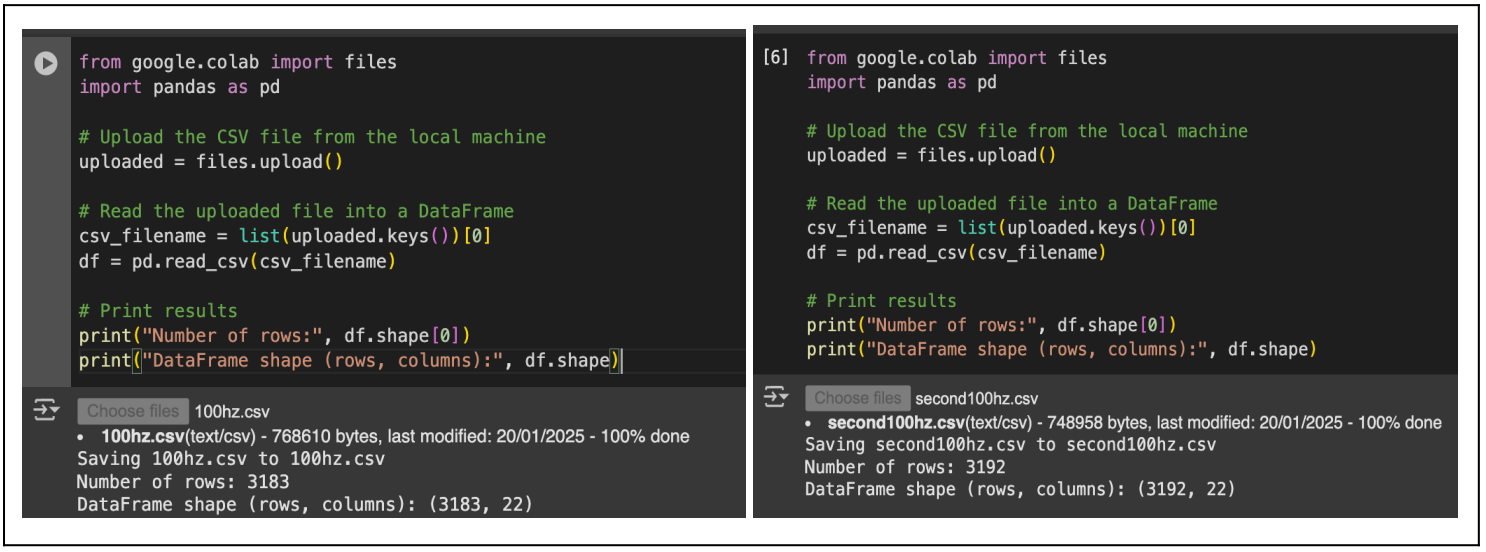


Figure 10: Number of recordings (rows) that have been produced with a 100 Hz sampling rate

Figure 11: Number of recordings (rows) that have been produced with a 30 Hz sampling rate

When further investigated, I noticed that some recordings have missing values. The missing values are introduced due to involved data transfer steps. When these missing values are pruned the 100 Hz recordings have 3083 and 3094 recordings respectively, and 30 Hz recordings have 1036 and 1053 recordings respectively. When these values are averaged out it can be said that a 100 Hz sampling rate produces 3088 recordings in 30 seconds and 30 Hz sampling rate produces 1044 recordings. While these values slightly deviate from theoretical values, it can be said that this deviation is negligible. Therefore, it can be concluded that my app solves the data quality issues some other publicly available datasets suffer from.

9.2 User Survey

The usability of my app is crucial as one of our aims in this research is to deploy a data collection instrument that can be used as a research tool. In order to evaluate the usability of my app, I carried out a user survey following the beta testing of my app. The user survey questions were taken from the standardized system usability scale (SUS) format. In this survey format, some questions are positively worded and some questions are negatively worded which helps an unbiased perspective of a system's usability. Figures 12 to 21 contains the questions and answers to the survey where 1 means strongly disagree and 5 means strongly agree .

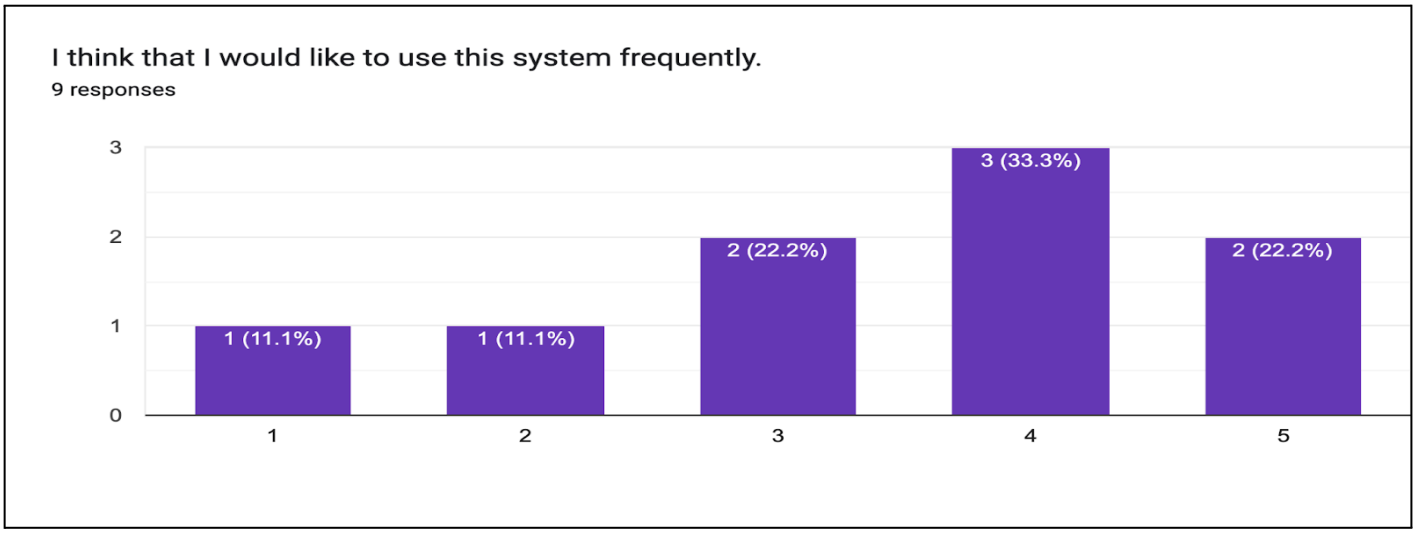


Figure 12

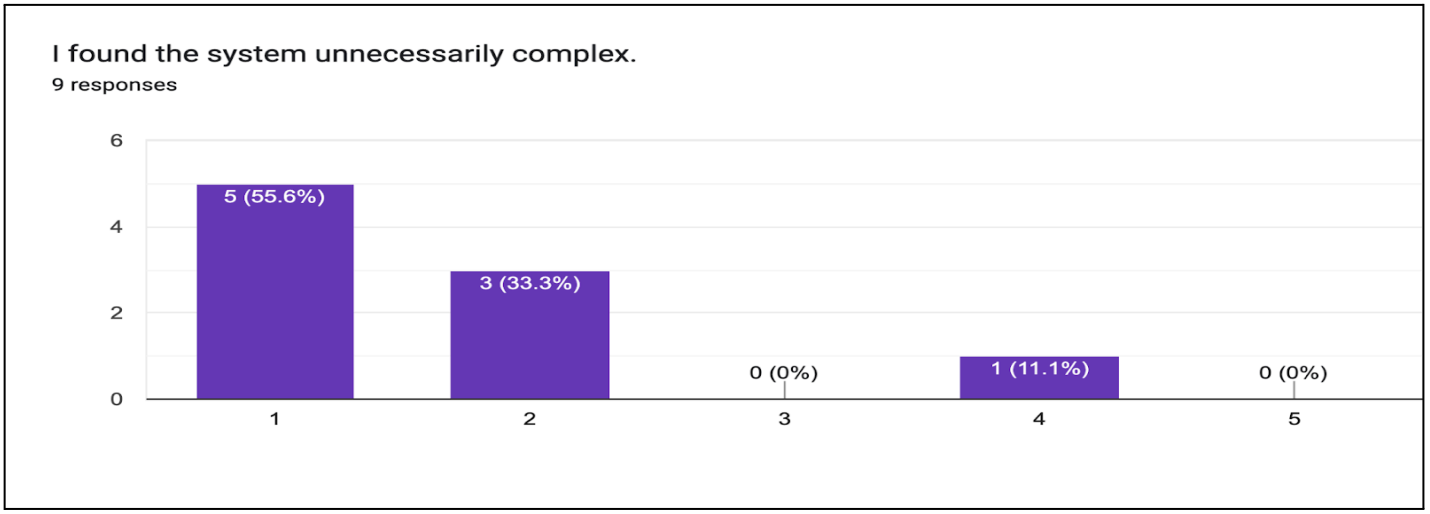


Figure 13

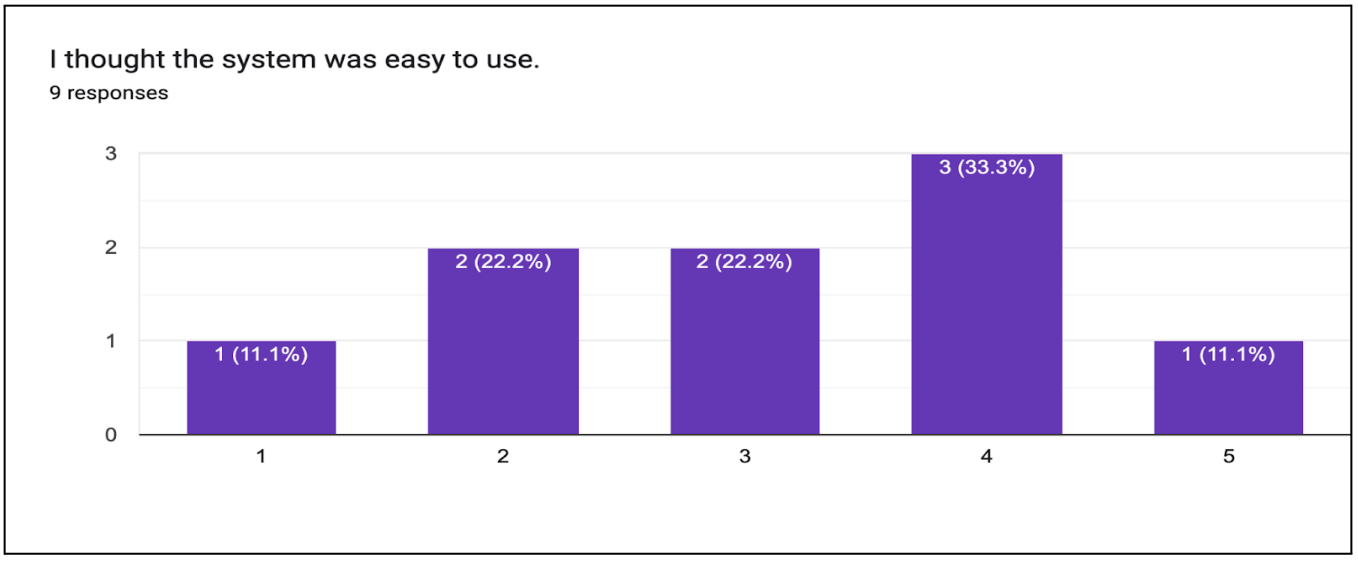


Figure 14

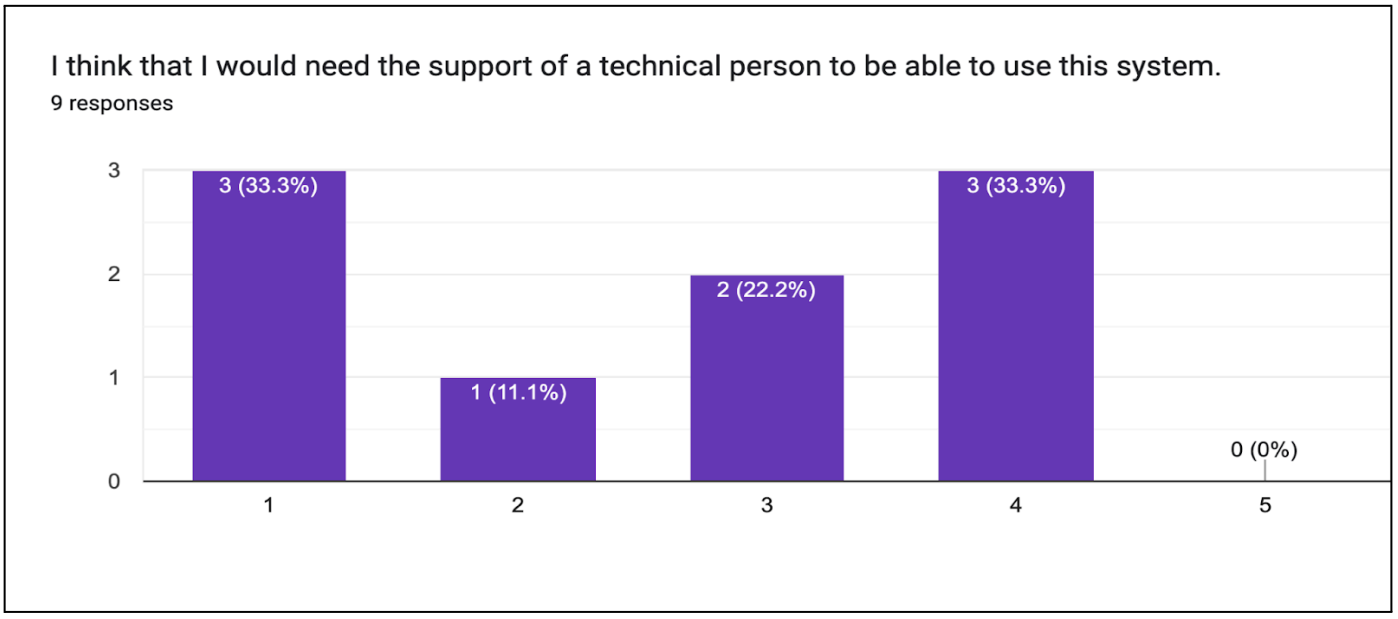


Figure 15

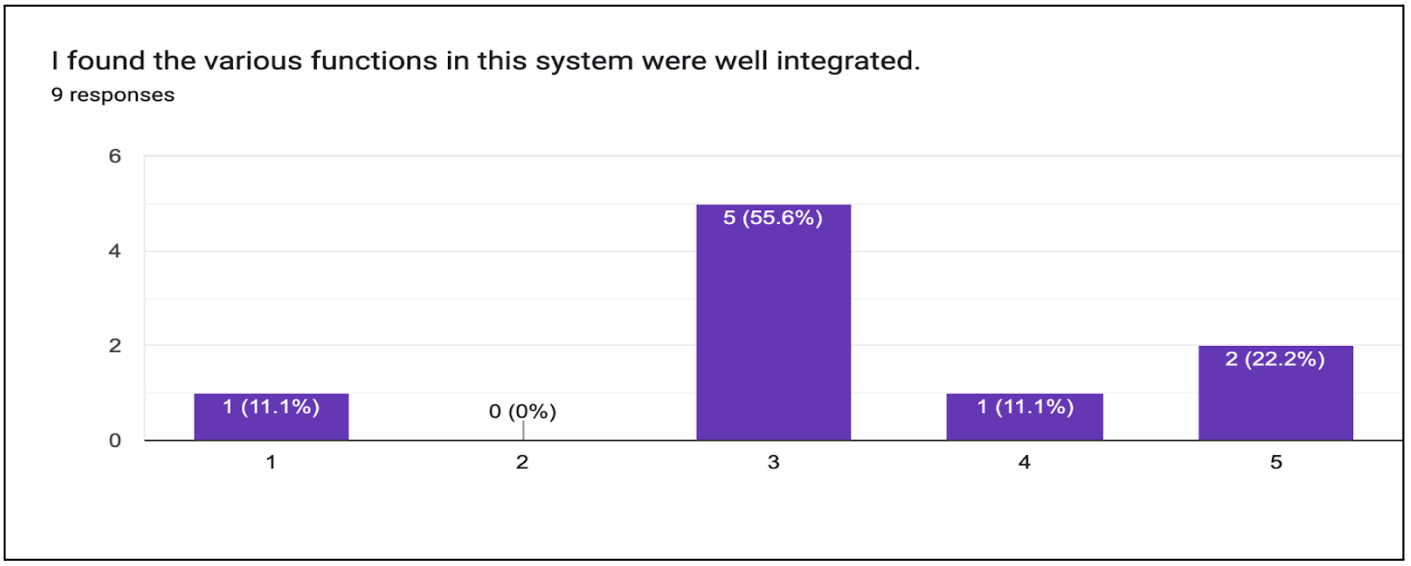


Figure 16

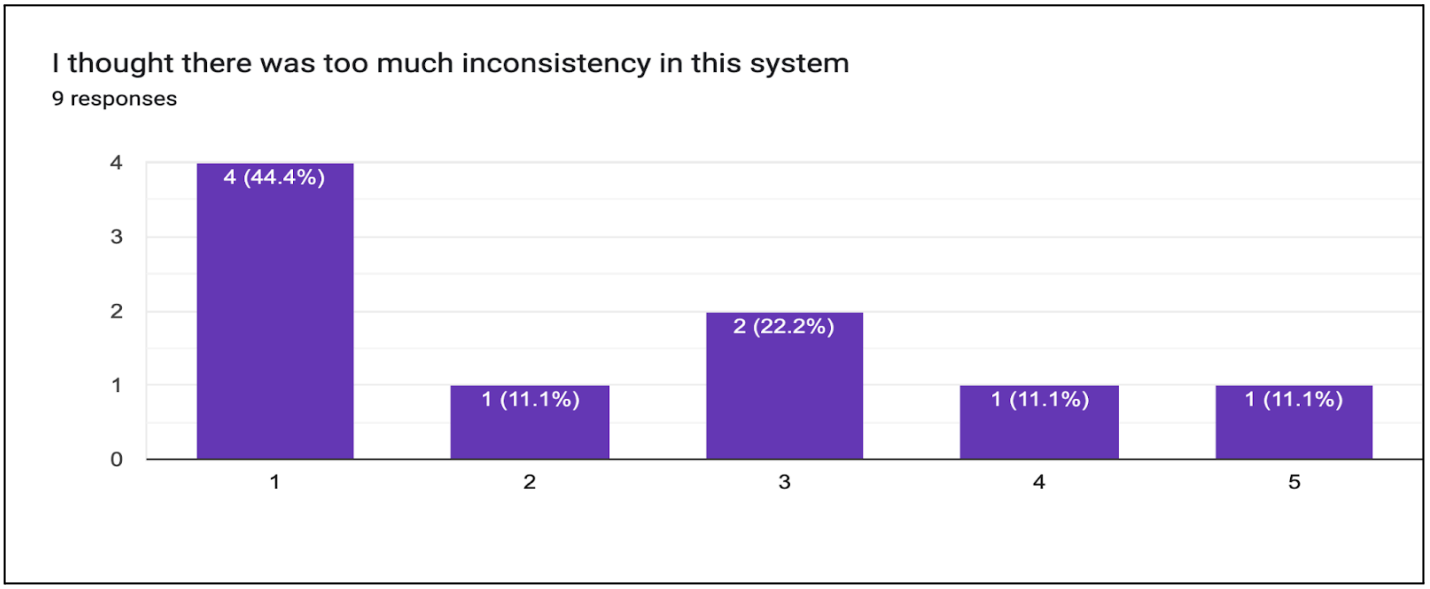


Figure 17

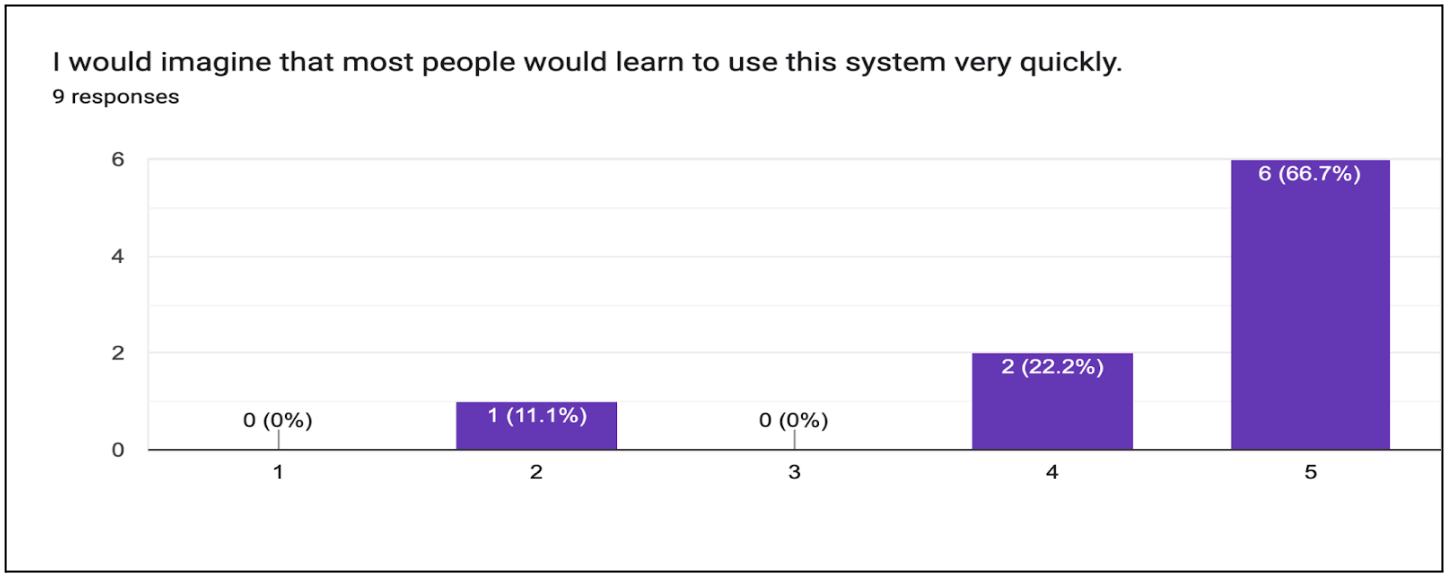


Figure 18

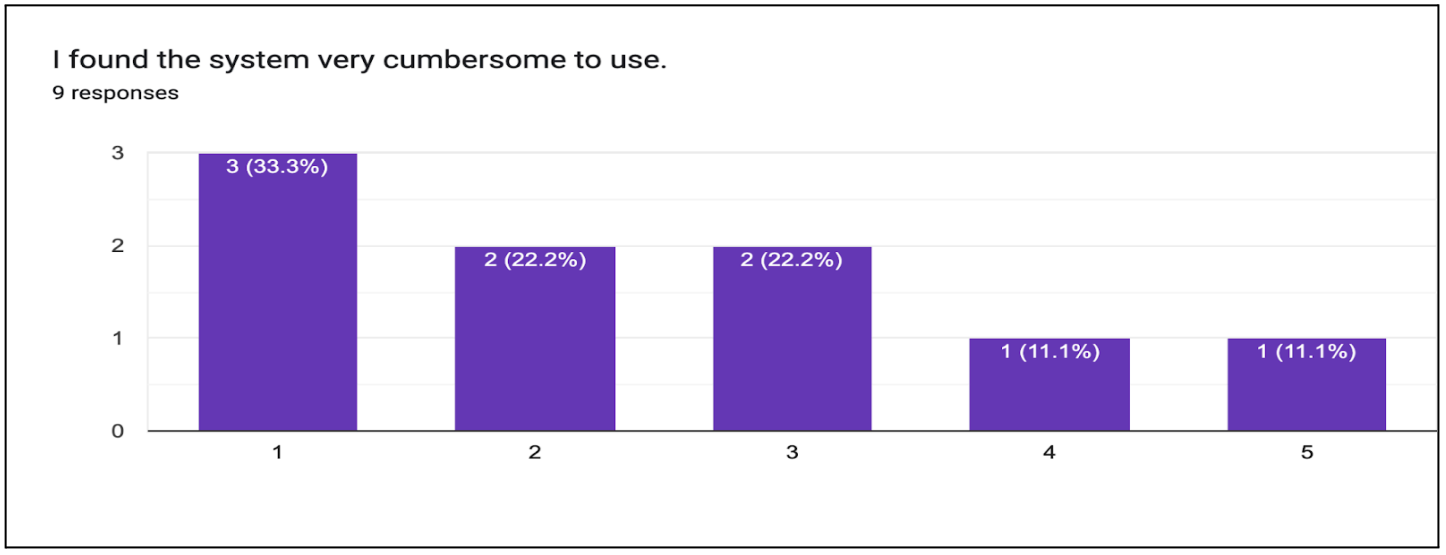


Figure 19

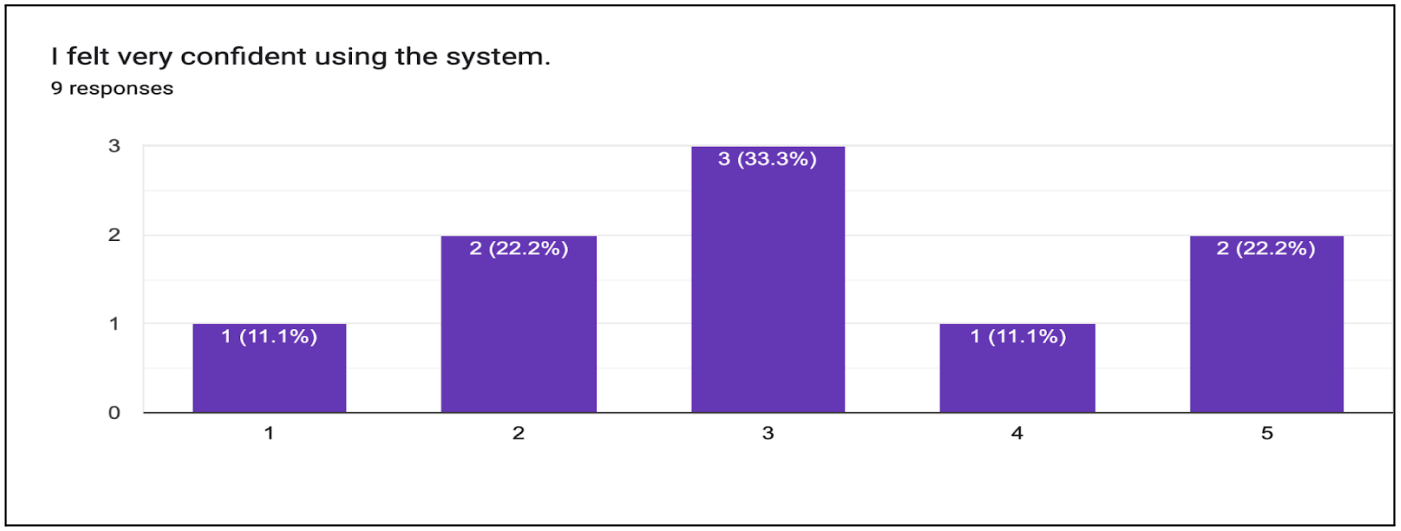


Figure 20

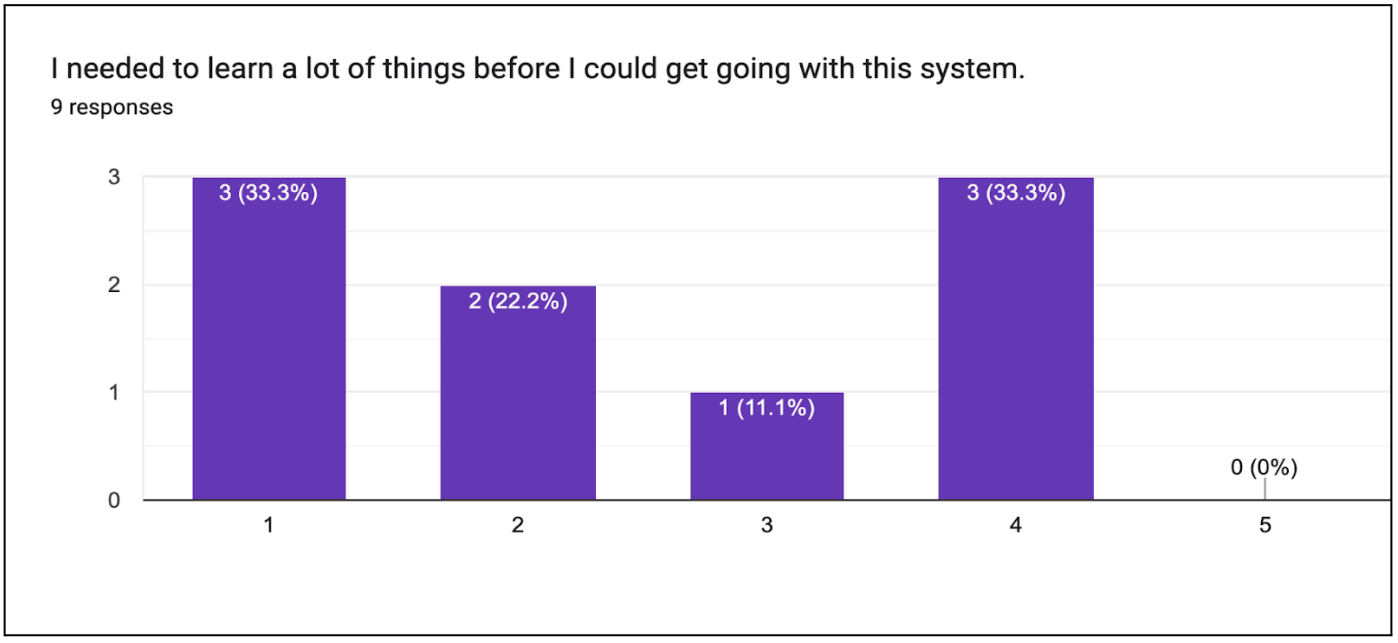


Figure 21

While this graph does not provide a complete image, it can still be said that the system is perceived to be usable to a good extent. This is evidenced by figure 13, 14,15,16,18, and 21. On the other hand figures 20, 19, 17, and 12 suggest that users found the system inconsistent which can be easily explained by the watch connectivity issues. For more accurate results, this user survey should be repeated with the final product. Unfortunately, the dissertation timeline and testers schedule didn't allow this.

9.3 Profiling Performance With Instruments

As discussed in the introduction, one of our goals was to explore whether this HAR system can be developed to become a HAR system that contains a DL model that can do activity inference. This requires the sensors in the apple watch to be used. Wearable devices have limited battery sizes, therefore, it is the bottleneck of the system, and it is crucial to know how much energy a specific sensor uses. I investigated this using the CPU Profiler of instruments framework.

While the instruments framework doesn't contain any way of knowing how much energy the watch app is consuming, the energy expenditure of the watch app can be thought to be parallel to CPU% usage as data collection is a CPU constrained task rather than a GPU constrained task. While model inference requires GPU usage, this model can be trained on a server and the inference can be performed on the iOS device which has relatively a large battery size when compared to the apple watch. I specifically focused on the apple watch in order to investigate the most constraining factor of the overall software system. Figure 22 shows an in screen image of the Instruments framework. The recordings I have made will be provided alongside the code of the app.

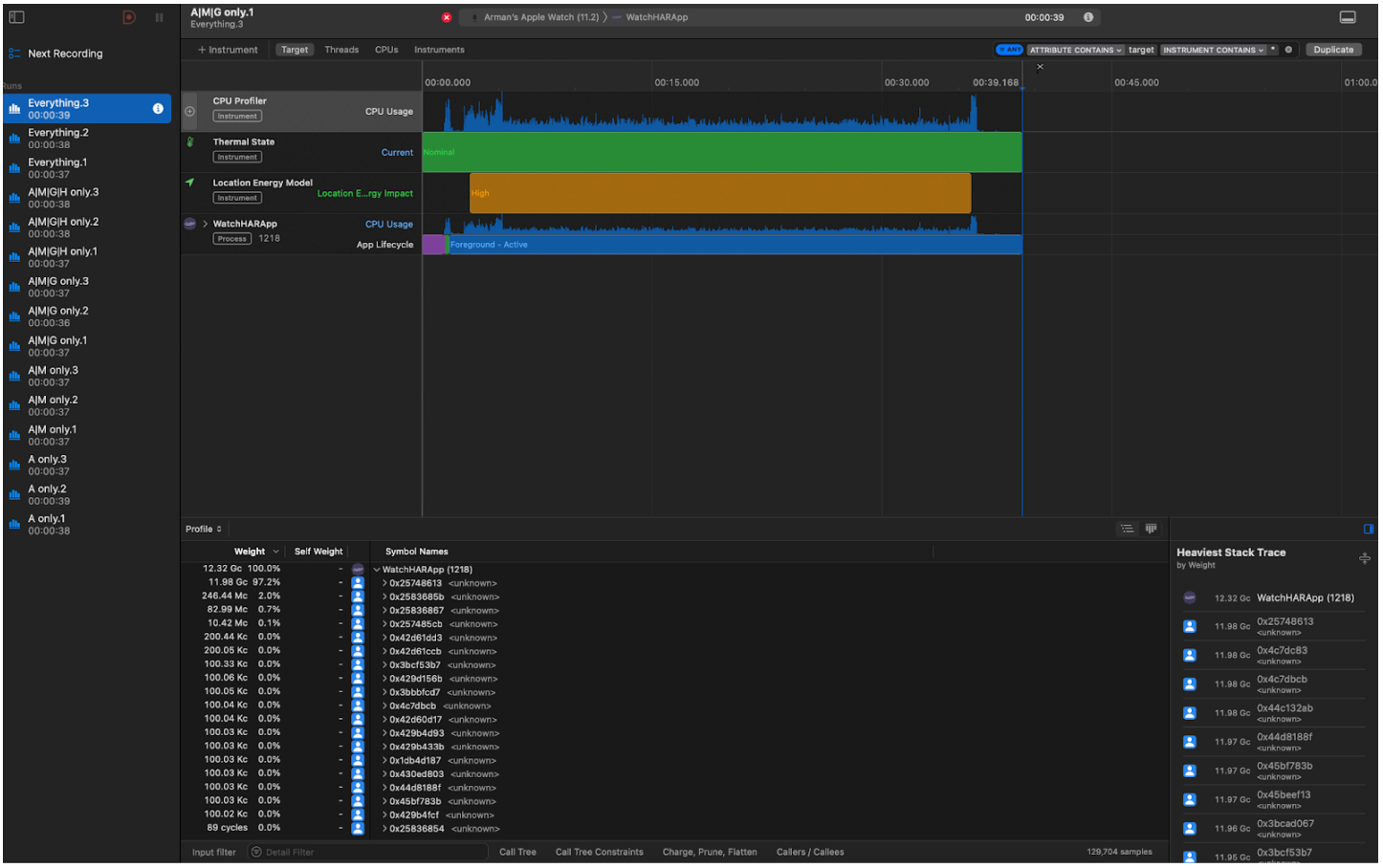


Figure 22: Instruments running CPU Profiler

The Instruments framework doesn't make CPU% usage directly available. However, I obtained support from Apple for my project and Apple engineers relayed to me that the CPU% usage can be approximately calculated by multiplying the number of samples by the per-sample duration, and finally dividing by total trace or time range duration. The CPU usage I obtained from my recordings can be found in the table below.

Sensors That Are Active	First Run CPU Usage %	Second Run CPU Usage %	Third Run CPU Usage %	Average CPU Usage %
Accelerometer Only	20	19	21	20
Accelerometer Magnetometer	25	27	24	25
Accelerometer Magnetometer Gyroscope	30	29	29	29
Accelerometer Magnetometer Gyroscope Heart Rate Sensor	31	30	29	30
Accelerometer Magnetometer Gyroscope Heart Rate Sensor GPS	34	31	33	33

It can be evaluated that this system can be evolved to become a fully functioning HAR system that can do activity recognition. My experiment also reveals that there isn't a significant energy usage penalty arising from multiple sensors being utilised. That is to say that there is significant value in investigating sensor fusion techniques to improve DL model inference performance.

10. Conclusion

10.1 Project Summary

This project provided an overview of the background of the human activity recognition research field. The insights obtained from this literature review enabled this project to introduce the system description, system requirements, system design, and system implementation of a mobile software system that can address the challenges HAR researchers face.

10.2 Successes and Drawbacks

The project can generally be considered a success. The user study proved that this app can be used to construct a general human activity dataset that contains high quality data and automates the data labeling process which are major challenges that inhibit the development of the field. While the usability app couldn't be evaluated to the full extent, the results still show that the app has an acceptable usability.

A drawback in our evaluation is the lack of an open sourced application which could be used as a reference point to evaluate how good our software system is when compared to other systems.

10.3 Future Work

The current system meets most of the requirements we set out in our requirements engineering process. However, there are still many opportunities for future work on the project that involves implementing a DL model on the device. Moreover, there is an urgent need to construct a generalized dataset using our system so that research can be reproducible.

Appendix A: Attribute Driven Design Process

Architecturally Significant Requirements

The Architecturally Significant Requirements were selected according to whether they had high architectural impact which are as follows: 01,02,11,13,14,15,17,19,22,23,24,25,26,27,28,29,30, D1, D2, D3, D4, D5, D6, D7

ADD Iterations

ADD Iteration: System Element 1

Chosen Element: Login Manager Element
Architectural Drivers:

- 11: Registration
- 13: 0 Auth Authentication
- 14,15: Authentication
    - Age Restriction Check
    - User Input Validity Check

First Depth First ADD Iteration of System Element 1

Child Components

Child Element: Authentication Service
Description: Performs validation on user input against age requirements and mandatory fields to enable logging in to access the main page.

Child Element: User Details Store
Description: Allows the app to hold crucial information about the user which enables other functionalities such as automating data labeling.

Child Element: 0 Auth Service
Description: Allows the user to log into the system more easily by using apple credentials.

Second Depth First ADD Iteration of System Element 1

Child Element: Load User Details Service
Description: Allows the app to obtain user details from the 0 Auth Service and communicates with the user details store to store the data.

Child Element: Login Interface
Description: Allows the user to interact with the app and input various user details

Architectural Pattern

	User Story Ids			
Styles/Patterns	**11**	**13**	**14**	**15**
Layered Architecture	✓	✓	✓	✓
Pipeline Architecture	✓	✓	✓	✓
MVVM Architecture	✓	✓	✓	✓

ADD Iteration: System Element 2

Chosen Element: Watch Connector
Architectural Drivers:

- 01: Apple Watch Connectivity
- 02: Automatic Utilisation Of A Default Sensor Suit

First Depth First ADD Iteration of System Element 2

Child Element: Message Sender
Description: Communicates with the apple watch to send commands that indicate sensors should start recording or they should stop recording.

Child Element: Message Receiver
Description: Communicates with the apple watch to receive collected data.

	User Story Ids	
Styles/Patterns	**01**	**02**
Layered Architecture		✓
Pipeline Architecture		
MVVM Architecture	✓	✓

ADD Iteration: System Element 3

Chosen Element: User Interface
Architectural Drivers:

- 17: Essential Activity Data Collection

- 19: Activity Specific Interface
- 22: Data Collection Controls
- 23: Data Collection Session Reusability
- **(D2)** - The application will be implemented with Swift and SwiftUI.
- **(D4)** - The project implementation has to be cost-effective.
- **(D5)** - The project will be implemented by a single developer.
- **(D6)** - The project is heavily time-constrained.

First Depth First ADD Iteration of System Element 3

Child Components

Child Element: Activity List
Description: These are the default set of commonly performed daily activities researchers are most likely to collect sensor data for.

Child Element: Activity Detail Page Interface
Description: This page will be used throughout a data collection session and it will give instructions on how to start a data collection session. This page will also display the duration of the session. Lastly, this page will be used to export the collected data once a data collection session is over.

Second Depth First ADD Iteration of System Element 3

Child Element: Export Button
Description: This button will be used to export data

Child Element: New Activity Creation Button
Description: This button will be used to create a new activity type

Child Element: New Activity Creation Interface
Description: This button will be used to enter the details of the new activity type

Architectural Pattern

	User Story Ids			
Styles/Patterns	**17**	**19**	**22**	**23**
Layered Architecture	✓		✓	✓
Pipeline Architecture				✓
MVVM Architecture	✓	✓	✓	✓

ADD Iteration: System Element 4

Chosen Element: Apple Watch App

Architectural Drivers:

- 24: Data Collection Session Status
- 25: Live Data Information
- 26: Data Collection Session Controls
- 27: Data Collection Session Controls
- **(D3)** - Sensor data must originate from a wearable device.
- **(D1)** - The application will be a mobile system.

First Depth First ADD Iteration of System Element 4

Child Components

Child Element: iPhone Connectivity
Description: Communicates with the iPhone app to determine when to start sensor recordings and send back the data that has been collected

Child Element: User Interface
Description: Displays the live data and whether there is an active data collection session.

Second Depth First ADD Iteration of System Element 4

Child Element: Message Sender
Description: Communicates with the iPhone app to send collected data.

Child Element: Message Receiver
Description: Communicates with the iPhone app to determine whether sensors should start or stop recording.

Child Element: Data Collection Service
Description: Manages swift frameworks that will be used to collect data

Third Depth First ADD Iteration of System Element 4

Child Element: Sensor Manager
Description: Turns on the sensors that will be used to record activity data which the user has specified.

Child Element: Data Store
Description: Stores the data that is being collected temporarily which will be read when the data collection session is ended.

Architectural Pattern

	User Story Ids			
Styles/Patterns	**24**	**25**	**26**	**27**
Layered Architecture	✓	✓	✓	✓
Pipeline Architecture	✓	✓		
MVVM Architecture	✓	✓	✓	✓

ADD Iteration: System Element 5

Chosen Element: Preferences Tab
Architectural Drivers:

- 28: Changing information about the user.
- 29: Changing sensors that are used to record data
- 30: Changing data collection duration
- D7: The application must be easily usable.

First Depth First ADD Iteration of System Element 5

Child Element: Input Fields
Description: These fields enable the user to provide new values for various user details.

Child Element: Sensor Toggles
Description: These toggles are used to turn on or off various sensors.

Child Element: Duration Slider
Description: This slider enables the user to change the duration of a sensor data collection session.

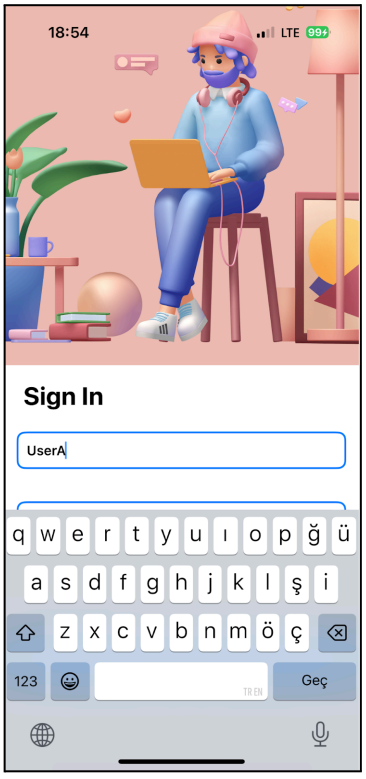
Second Depth First ADD Iteration of System Element 5

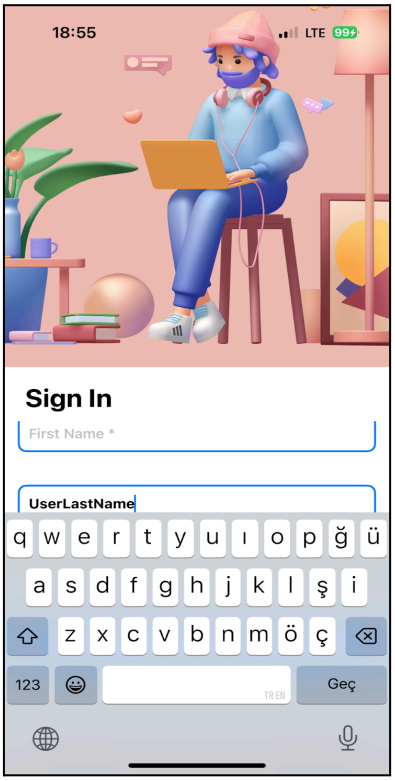
Child Element: “Save Changes” button
Description: This button enables the user to save the changed details. It exists to increase the usability of the user interface.

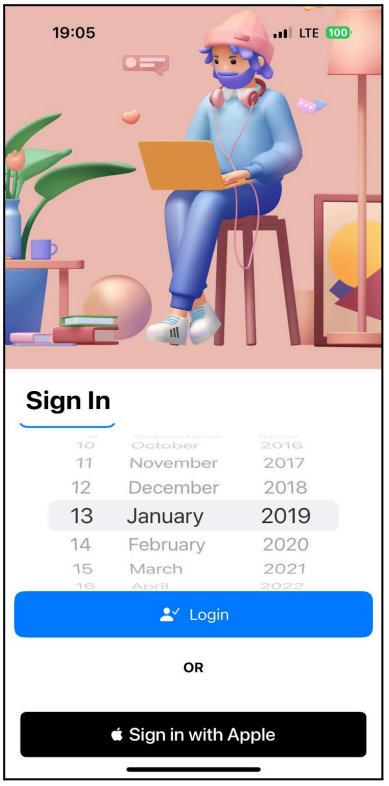
Architectural Pattern

	User Story Ids		
Styles/Patterns	**28**	**29**	**30**
Layered Architecture	✓	✓	✓
Pipeline Architecture	✓	✓	
MVVM Architecture	✓	✓	✓

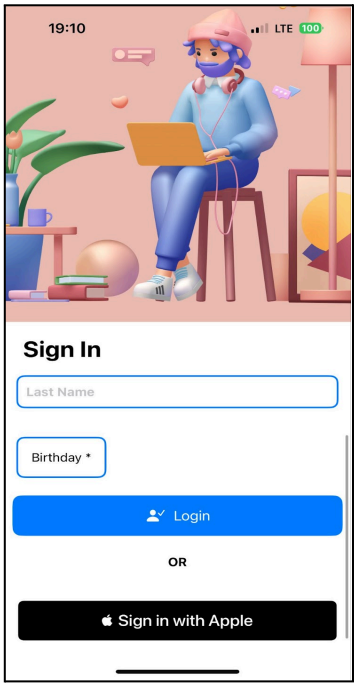
Appendix B: Proof Of Testing

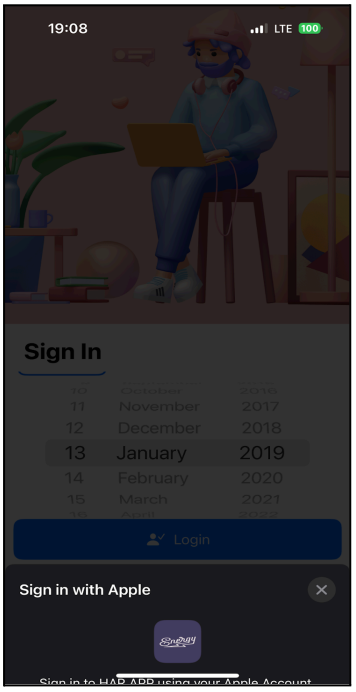




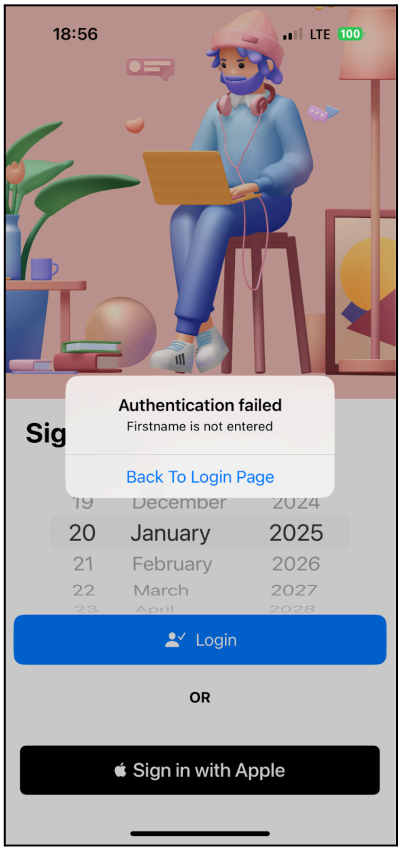


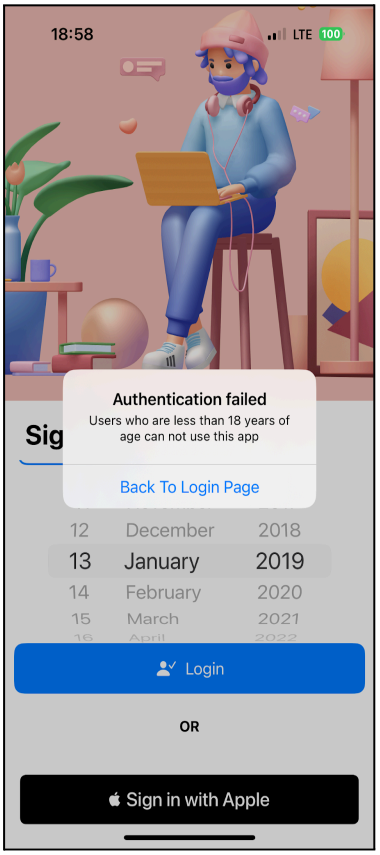
Test Cases 1,2,3 Respectively





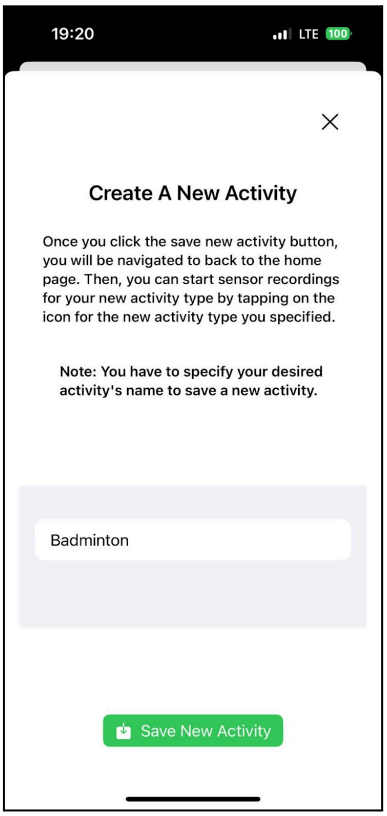
Test Case 5



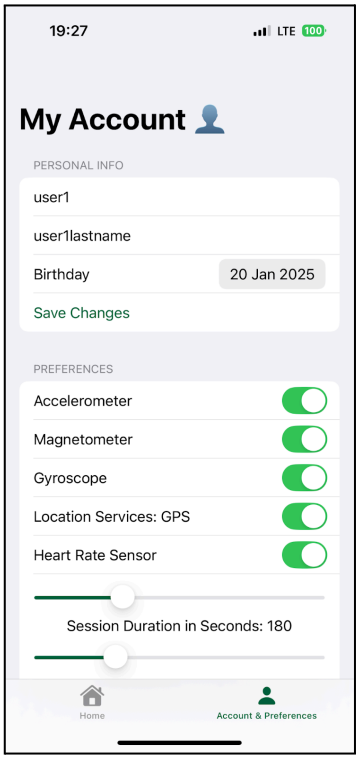


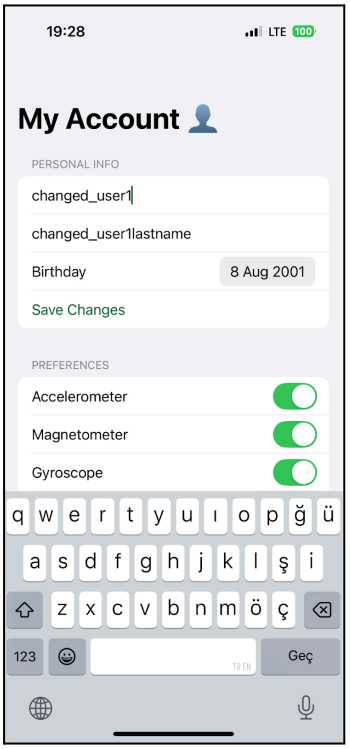
Test Cases 6 and 7 respectively

Test Case 8

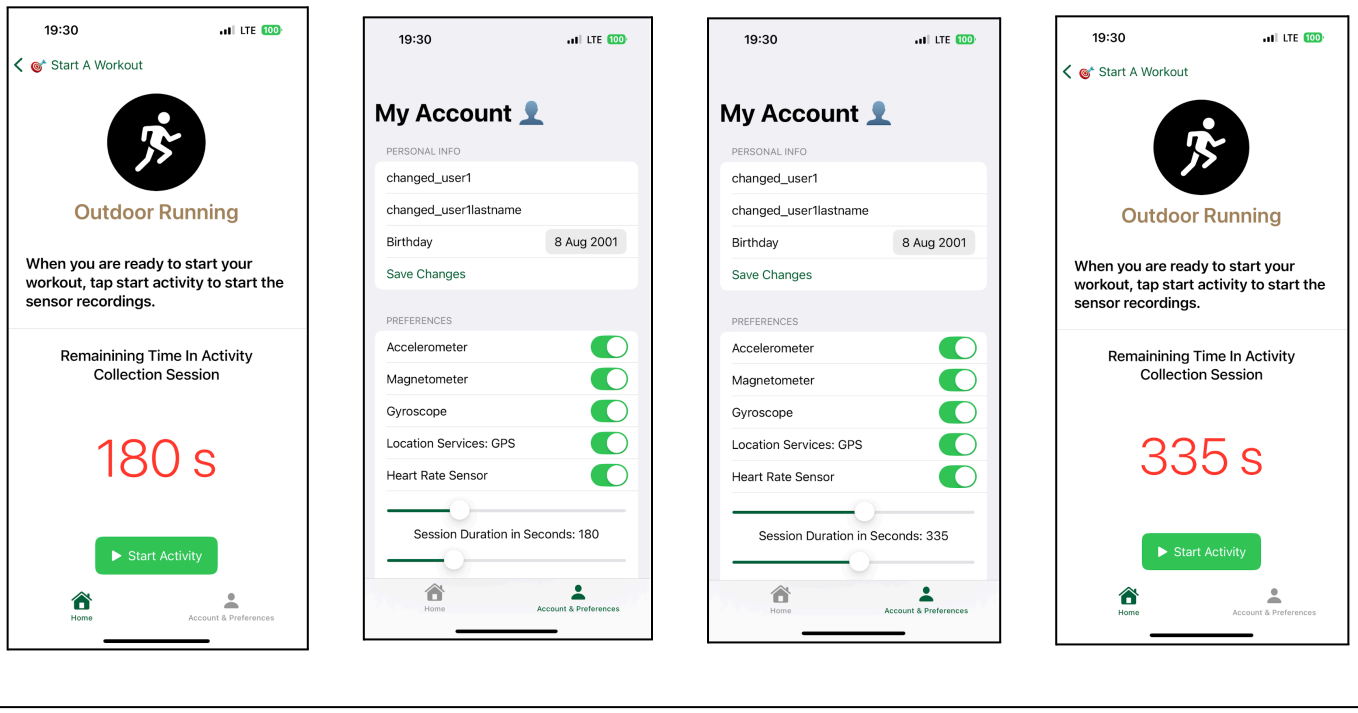


Test Cases 9, 10, 11

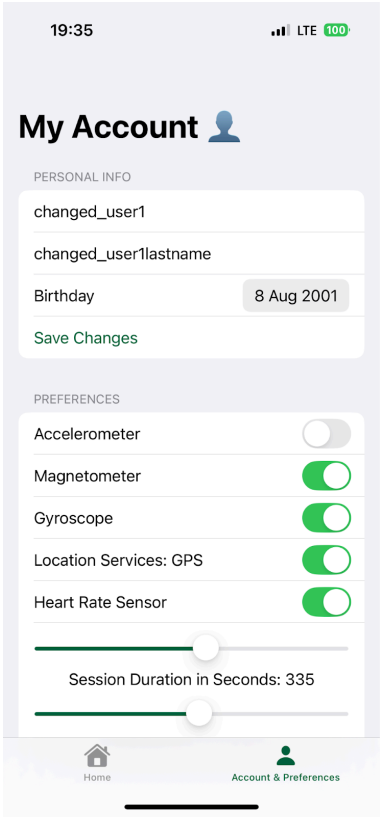




Test Cases 13,14

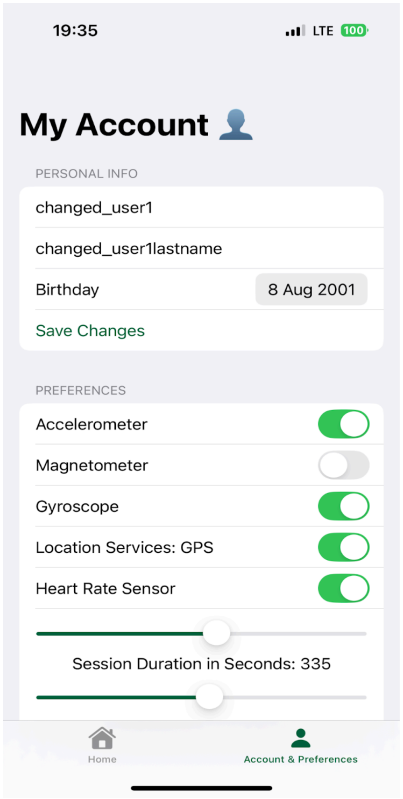


Test Cases 16,17



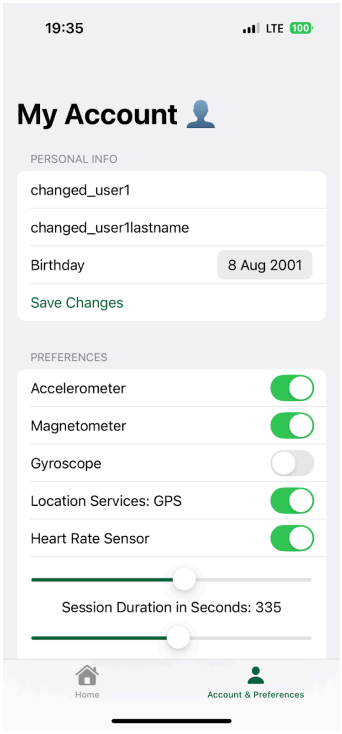

```
{
  "courseAccuracy" : "-1.0",
  "gyroX" : "0.38662",
  "magX" : "-26.23386",
  "accelZ" : "N\/A",
  "speedAccuracy" : "-1.0",
  "heading" : "105.70506286621094",
  "timestamp" : 1737326517.9870539,
  "gyroY" : "3.07288",
  "accelX" : "N\/A",
  "altitudeAccuracy" : "30.0",
  "speed" : "-1.0",
  "accelY" : "N\/A",
  "latitude" : "36.851639533266265",
  "coordinateAccuracy" : "6.276386648050005",
  "magZ" : "-0.23097",
  "headingAccuracy" : "89.0150146484375",
  "altitude" : "28.61454094922638",
  "longitude" : "28.243947705218076",
  "course" : "-1.0",
  "gyroZ" : "-0.59796",
  "heartbeat" : "72",
  "magY" : "-48.83765"
},
```


Test Case 18



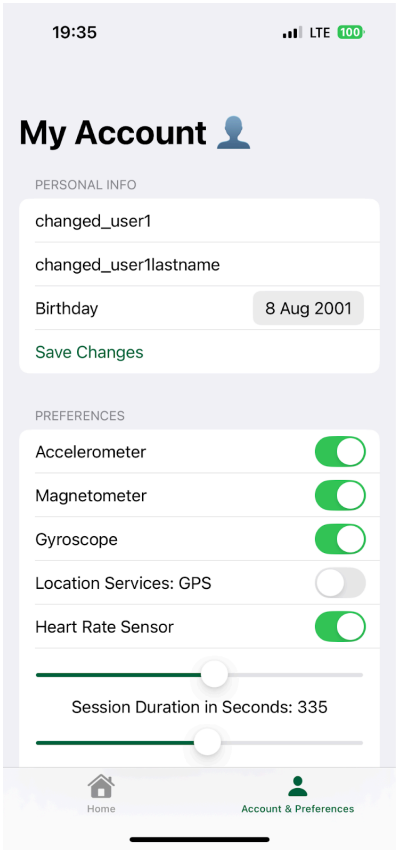

```
{
  "headingAccuracy" : "89.9000015258789",
  "gyroX" : "0.40363",
  "speedAccuracy" : "-1.0",
  "coordinateAccuracy" : "6.964130244200212",
  "accelY" : "-0.00104",
  "speed" : "-1.0",
  "heading" : "88.7718276977539",
  "timestamp" : 1737326427.27861,
  "longitude" : "28.24394770990913",
  "magX" : "N\/A",
  "gyroZ" : "-0.15804",
  "altitudeAccuracy" : "30.0",
  "gyroY" : "-2.98794",
  "magZ" : "N\/A",
  "accelZ" : "0.01003",
  "latitude" : "36.85163951939591",
  "heartbeat" : "74",
  "accelX" : "0.00225",
  "courseAccuracy" : "-1.0",
  "course" : "-1.0",
  "altitude" : "28.61454094922638",
  "magY" : "N\/A"
},
```


Test Case 19



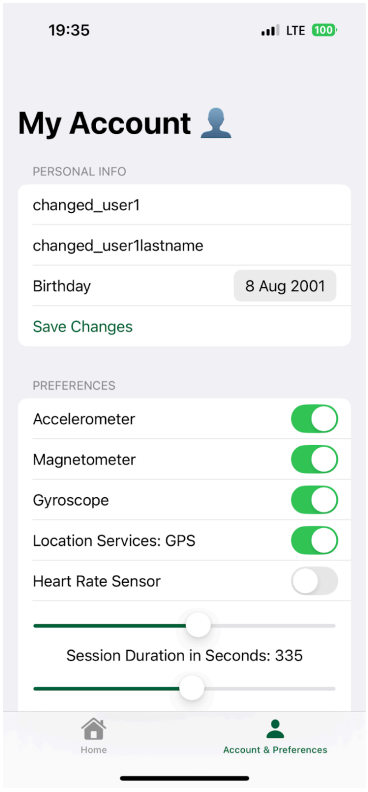

```
{
  "courseAccuracy" : "-1.0",
  "magX" : "-42.25980",
  "speed" : "-1.0",
  "heartbeat" : "74",
  "accelZ" : "0.00857",
  "magZ" : "-13.46660",
  "gyroZ" : "N\/A",
  "gyroX" : "N\/A",
  "altitudeAccuracy" : "30.0",
  "latitude" : "36.85163949165764",
  "longitude" : "28.243947705383192",
  "accelX" : "0.00321",
  "speedAccuracy" : "-1.0",
  "accelY" : "-0.00734",
  "altitude" : "28.61454094922638",
  "coordinateAccuracy" : "7.418850444971194",
  "heading" : "87.15364837646484",
  "course" : "-1.0",
  "headingAccuracy" : "89.9000015258789",
  "magY" : "-87.24454",
  "gyroY" : "N\/A",
  "timestamp" : 1737326380.030571
},
```


Test Case 20



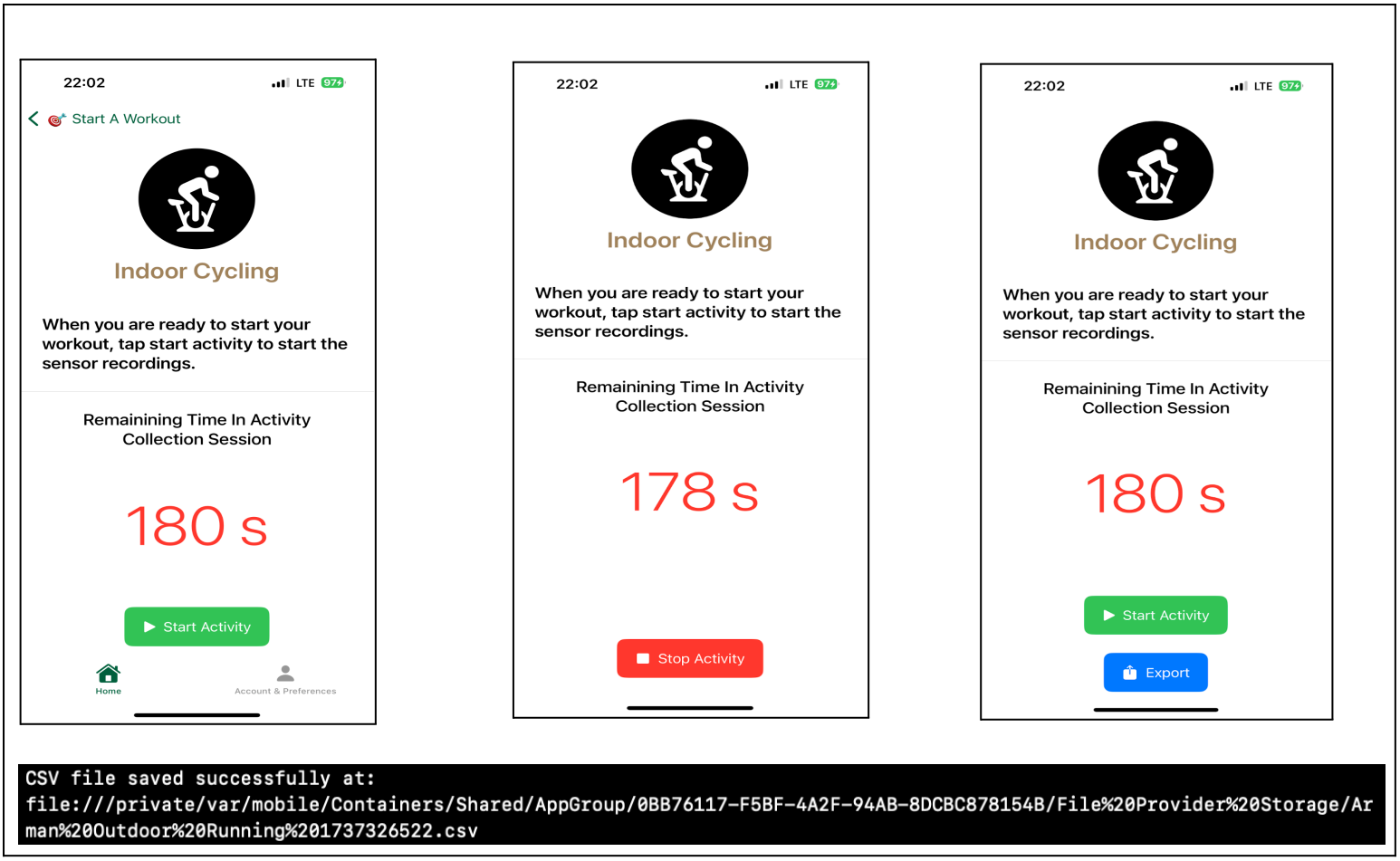

```
{
  "headingAccuracy" : "-1.0",
  "magY" : "-130.16142",
  "magZ" : "15.25305",
  "coordinateAccuracy" : "-1.0",
  "speed" : "-1.0",
  "longitude" : "-1.0",
  "gyroX" : "0.32593",
  "speedAccuracy" : "-1.0",
  "accelZ" : "0.00845",
  "accelX" : "0.00254",
  "altitudeAccuracy" : "-1.0",
  "latitude" : "-1.0",
  "timestamp" : 1737326282.0920548,
  "accelY" : "-0.00511",
  "gyroZ" : "-0.11626",
  "courseAccuracy" : "-1.0",
  "magX" : "-41.00153",
  "gyroY" : "-2.85998",
  "course" : "-1.0",
  "heading" : "-1.0",
  "heartbeat" : "74",
  "altitude" : "-1.0"
},
```


Test Case 21

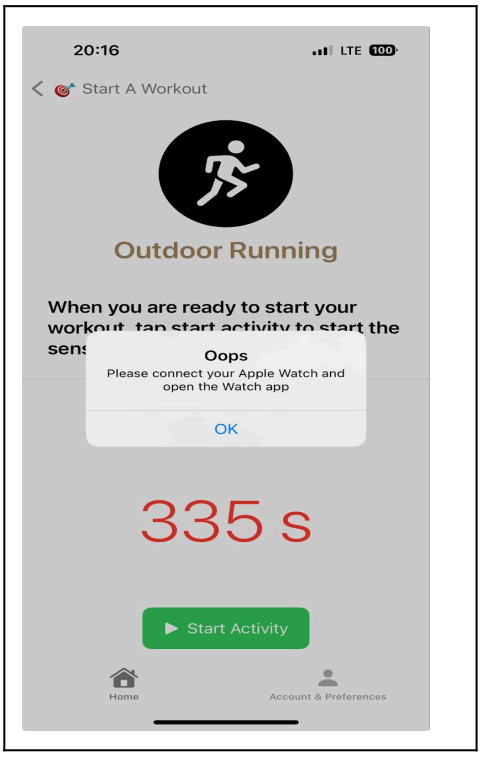



```
  "heartbeat" : "-1",
  "headingAccuracy" : "89.9000015258789",
  "course" : "-1.0",
  "gyroY" : "-2.86356",
  "heading" : "80.7494888305664",
  "coordinateAccuracy" : "9.482236737137052",
  "timestamp" : 1737326131.9589949,
  "gyroX" : "0.24970",
  "magZ" : "55.92972",
  "accelY" : "-0.00932",
  "accelX" : "-0.00012",
  "gyroZ" : "-0.16154",
  "speed" : "-1.0",
  "courseAccuracy" : "-1.0",
  "speedAccuracy" : "-1.0",
  "altitude" : "28.615403070808412",
  "accelZ" : "0.00915",
  "latitude" : "36.85163941354165",
  "altitudeAccuracy" : "30.000000000000007",
  "magY" : "-73.56525",
  "magX" : "-127.45700",
  "longitude" : "28.243947714465946"
},
```


Test Case 22



Test Case 24, 25, 27



Test Case 23

Appendix C: Approval Letter

University of St Andrews | FOUNDED 1413

School of Computer Science Ethics Committee

08 November 2024

Dear Erica

Thank you for submitting your ethical application which was considered by the School Ethics Committee.

The School of Computer Science Ethics Committee, acting on behalf of the University Teaching and Research Ethics Committee (UTREC), has approved this application:

Approval Code:	CS18177	**Approved on:**	07/11/2024	**Approval Expiry:**	07/11/2029
Project Title:	Personalised human activity recognition system on mobile devices				
Researcher(s):	Juan Ye, Arman Özkaya, Fiona Killalea				
Supervisor(s):					

The following supporting documents are also acknowledged and approved:

1. Application Form
2. Participant Information Sheet
3. Participant Consent Form
4. Participant Advertisement
5. Sample Questions
6. Study Facility Permission

Approval is awarded for 5 years, see the approval expiry date above.

If your project has not commenced within 2 years of approval, you must submit a new and updated ethical application to your School Ethics Committee.

If you are unable to complete your research by the approval expiry date you must request an extension to the approval period. You can write to your School Ethics Committee who may grant a discretionary extension of up to 6 months. For longer extensions you must submit an ethical review application form.

If you need to make changes to your project, you must submit an ethical amendment application.

You must report any serious adverse events, or significant changes not covered by this approval, related to this study immediately to the School Ethics Committee.

Approval is given on the following conditions:

- that you conduct your research in line with:
  - the details provided in your ethical application
  - the University's Principles of Good Research Conduct
  - the conditions of any funding associated with your work
- that you obtain all applicable additional documents (see the 'additional documents' webpage for guidance) before research commences.

School of Computer Science Ethics Committee
Dr Olexandr Konovalov (Convenor), Jack Cole Building, North Haugh, St Andrews, Fife, KY16 9SX
T: 01334 463273 E: ethics-cs@st-andrews.ac.uk

University of St Andrews | FOUNDED 1413

School of Computer Science Ethics Committee

You should retain this approval letter with your study paperwork.

Yours sincerely,

Hazel Wallace

SEC Administrator

References

[1] S. Zhang *et al.*, “Deep Learning in Human Activity Recognition with Wearable Sensors: A Review on Advances,” *Sensors*, vol. 22, no. 4, p. 1476, Feb. 2022, doi: 10.3390/s22041476.

[2] V. of T. Consumer, “Put a ring on it: Understanding consumers’ year-over-year wearable adoption patterns | Rock Health,” *Rock Health | We’re Powering the Future of Healthcare. Rock Health Is a Seed and Early-stage Venture Fund That Supports Startups Building the Next Generation of Technologies Transforming Healthcare.*, Aug. 05, 2024. https://rockhealth.com/insights/put-a-ring-on-it-understanding-consumers-year-over-year-wearable-adoption-patterns/

[3] M. Straczkiewicz, P. James, and J.-P. Onnela, “A systematic review of smartphone-based human activity recognition methods for health research,” *Npj Digital Medicine*, vol. 4, no. 1, Oct. 2021, doi: 10.1038/s41746-021-00514-4.

[4]G. Weiss. "WISDM Smartphone and Smartwatch Activity and Biometrics Dataset ," UCI Machine Learning Repository, 2019. [Online]. Available: https://doi.org/10.24432/C5HK59.

[5] A. Bulling, U. Blanke, and B. Schiele, “A tutorial on human activity recognition using body-worn inertial sensors,” *ACM Computing Surveys*, vol. 46, no. 3, pp. 1–33, Jan. 2014, doi: 10.1145/2499621.

[6] J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra. "Human Activity Recognition Using Smartphones," UCI Machine Learning Repository, 2013. [Online]. Available: https://doi.org/10.24432/C54S4K.

[7] M. Shoaib, S. Bosch, O. Incel, H. Scholten, and P. Havinga, “Fusion of smartphone motion sensors for physical activity recognition,” *Sensors*, vol. 14, no. 6, pp. 10146–10176, Jun. 2014, doi: 10.3390/s140610146.

[8] C. Chen, R. Jafari, and N. Kehtarnavaz, “UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor,” *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 168–172, Sep. 2015, doi: 10.1109/icip.2015.7350781.

[9] A. Stisen *et al.*, “Smart Devices are Different,” *SenSys ’15: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, Nov. 2015, doi: 10.1145/2809695.2809718.

[10] K. Altun, B. Barshan, and O. Tunçel, “Comparative study on classifying human activities with miniature inertial and magnetic sensors,” *Pattern Recognition*, vol. 43, no. 10, pp. 3605–3620, Apr. 2010, doi: 10.1016/j.patcog.2010.04.019.

[11] O. Banos *et al.*, “Design, implementation and validation of a novel open framework for agile development of mobile health applications,” *BioMedical Engineering OnLine*, vol. 14, no. Suppl 2, p. S6, Jan. 2015, doi: 10.1186/1475-925x-14-s2-s6.

[12] A. Reiss. "PAMAP2 Physical Activity Monitoring," UCI Machine Learning Repository, 2012. [Online]. Available: https://doi.org/10.24432/C5NW2H.

[13] “Human activity recognition in the context of industrial human-robot interaction,” *IEEE Conference Publication | IEEE Xplore*, Dec. 01, 2014. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7041588

[14] S. Iftikhar *et al.*, “AI-based fog and edge computing: A systematic review, taxonomy and future directions,” *Internet of Things*, vol. 21, p. 100674, Dec. 2022, doi: 10.1016/j.iot.2022.100674.

[15] M. D. Kadenic, K. Koumaditis, and L. Junker-Jensen, “Mastering scrum with a focus on team maturity and key components of scrum,” *Information and Software Technology*, vol. 153, p. 107079, Sep. 2022, doi: 10.1016/j.infsof.2022.107079.

[16] “Manifesto for Agile software development.” https://agilemanifesto.org/

[17] L. Bass, P. Clements, and R. Kazman, *Software Architecture in practice: Software Architect Practice_c3*. Addison-Wesley, 2012.

[18] Michaelstonis, “Model-View-ViewModel - .NET,” *Microsoft Learn*, Sep. 10, 2024. https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

[19] Zeba, “MVVM in iOS Swift - Zeba - Medium,” *Medium*, May 17, 2024. [Online]. Available: https://medium.com/@zebayasmeen76/mvvm-in-ios-swift-6afb150458fd

[20] A. Turner, “How many people have smartphones worldwide (2025),” *BankMyCell*, Jan. 04, 2025. https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

[21] Android Open Source Project, “Android Open Source Project,” *Android Open Source Project*. https://source.android.com/#:~:text=Android%20is%20an%20open%20source,source%20project%20led%20by%20Google.

[22] “Green Color: Hex Code, Palettes & Meaning | Figma,” *Figma*. https://www.figma.com/colors/green/

[23] “Featured | Apple Developer Documentation,” *Apple Developer Documentation*. https://developer.apple.com/documentation/