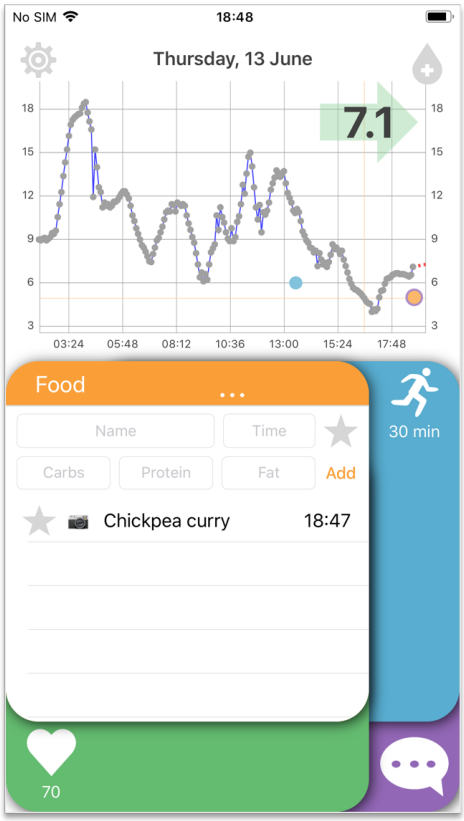
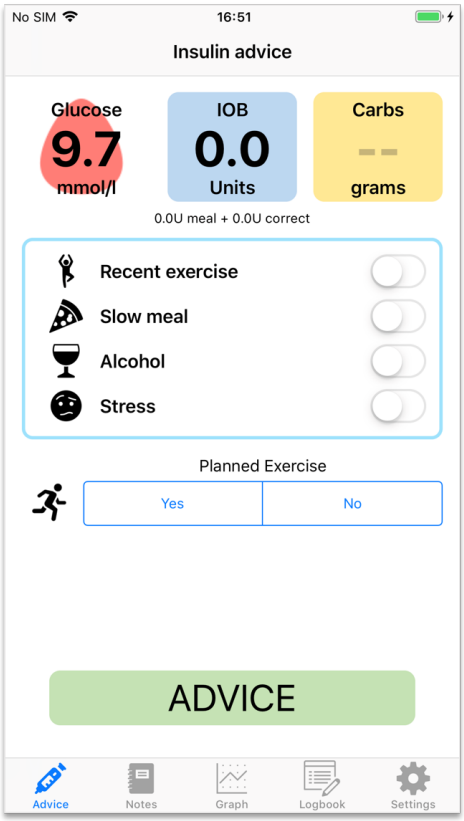
Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2019





Project Title:	**A Smartphone-based Machine Learning System for Diabetes Management**
Student:	**Ryan S. Armiger**
CID:	**01065807**
Course:	**4T**
Project Supervisor:	**Dr Kezhi Li**
Second Marker:	**Dr Pantelis Georgiou**

Abstract

Modern technology enables new opportunities to create decision support systems that empower people living with diabetes to improve their blood glucose control, aiming to reduce hypo/hyperglycemic events and the long-term complications of their condition. This report covers the development of 2 smartphone apps, with the future aim of combining their benefits into a single system. The ARISES app focuses on using machine learning to predict blood glucose levels and the Advanced Bolus Calculator for Diabetes (ABC4D) app explores improving personalised bolus calculation using case-based reasoning. Both systems collect user information throughout the day and physiological data is collected using a continuous glucose monitor and the Empatica E4 wristband, connected to a phone via bluetooth. The apps interface with a server to aggregate their data and allow remote monitoring of clinical trial participants. The algorithms implemented on the device are validated against their existing off-device behaviour and the remaining software has undergone either unit testing or user testing. Focus groups were consulted on the UI design, shaping the final interface to suit users. While some challenges exist to implementing these systems, smartphones appear well-suited to providing convenient real-time advice.

Acknowledgements

I would like to thank Dr. Kezhi Li for the guidance and supervision provided during this project. I am also grateful to Dr. Pau Herrero for the rewarding opportunity to contribute to the ABC4D project.

Contents

1 Introduction **7**

1.1 Project Motivation 7

1.2 Project Definition 7

1.3 Report Structure 9

2 Background **10**

2.1 Blood Glucose Prediction 10

2.1.1 Datasets 11

2.1.2 Missing data 11

2.1.3 Selecting Features 13

2.1.4 Overfitting 15

2.1.5 Model Topography 15

2.2 Bolus Calculation 16

2.3 Implementation Background 17

3 Requirements **19**

4 Analysis and Design **22**

4.1 Data Collection 22

4.1.1 Continuous Glucose Monitor 23

4.1.2 Empatica Wristband 24

4.1.3 Diary Entry 25

4.2 Data Storage 26

4.3 Data Processing 28

4.3.1 Blood Glucose Prediction 29

4.3.2 Bolus Calculation 30

4.4 Data Visualisation 31

4.4.1 Display 31

4.4.2 Domains 34

4.5 Data Uploading 34

4.6 User Experience 36

4.6.1 Performance 36

4.6.2 User Interface 36

5 Implementation **39**

5.1 Data Collection 39

5.1.1 Continuous Glucose Monitor 39

5.1.2 Empatica Wristband 40

5.1.3 Diary Entry 41

5.2 Data Storage 42

5.3 Data Processing 43

5.3.1 Blood Glucose Prediction 43

5.3.2 Bolus Calculator 44

5.4 Data Visualisation 45

5.4.1 Display 45

5.4.2 Domains 46

5.5 Data Uploading 47

5.6 User Interface 48

6 Testing **50**

6.1 Continuous Glucose Monitor 50

6.2 Empatica Wristband 52

6.3 Data Storage 53

6.4 Machine Learning Model 55

6.5 Bolus Calculator 55

7 Results **57**

8 Conclusion **59**

9 Further Work **61**

10 User Guide **63**

A Project Files **64**

A.1 Project GitHub page 64

List of Figures

1.1 ARISES (left) and ABC4D (right) main app screens 8

3.1 Original layout designs for ARISES and ABC4D apps 19

4.1 From left to right. Empatica E4 wristband, Dexcom G6 CGM, ARISES diary 22
4.2 ABC4D app with some data collection devices 23
4.3 Each ARISES Domain 26
4.4 Insulin entry system. Left: old system, Right: new system with bolus calculator recommendations 27
4.5 Empatica data (Top) compared to CGM data (Bottom) 28
4.6 Screenshot of graph from the device, displaying a prediction line . . . 29
4.7 Left: Settings required by the bolus calculator, Right: Some typical values 30
4.8 Old ARISES graph (Left) compared with new graph (Right) 31
4.9 Horizontal graph 32
4.10 Adjustable graph and diary sizes 33
4.11 Possible trend arrow positions 33
4.12 Health table with different filters set. Top Left: Last 7 days, Top Right: Last 7 days and Illness, Bottom Left: Last 7 days and Exercise, Bottom Right: Last 30 days and Illness 35
4.13 System for dynamically adjusting insulin and seeing the BG prediction for that value 36
4.14 Data displayed on the ABC4D web interface, collected during early testing 37
4.15 Layout update comparison. Left: old layout on the iPhone 5s; Middle: new layout on the iPhone 5s; Right: new layout on an iPhone 7plus . 37
4.16 Comparing old skeleton layout (Top) with new layout (Bottom) across a range of screen sizes. 38

5.1 Left: Time picker, Right: Keypad without decimal point 41
5.2 Showing moving of a view to prevent keyboard obscuring an input . . 42
5.3 ARISES database graph 43

5.4 Diagram showing objects fetched and functions required to calculate a bolus suggestion 45
5.5 Breakdown of views in layout 48

6.1 CGM emulators. Left: Constant output value, Right: Advanced profile output 50
6.2 Comparison of graphs for CGM data from Dexcom app (Left) and ABC4D app (Right) 51
6.3 Empatica data values stored by the E4 realtime app 52
6.4 EDA data stored by the Empatica over 10 seconds 53
6.5 Temperature data stored by the Empatica over 10 seconds 53
6.6 BVP data stored by the Empatica over 10 seconds 54
6.7 An example exercise log (Top) and the corresponding database entry (Bottom) 54
6.8 Bolus calculator testing results. Left: MATLAB output, Right: Swift implementation output 56

List of Abbreviations

ABC4D Advanced Bolus Calculator for Diabetes

AI Artificial Intelligence

ANN Artificial Neural Network

AWS Amazon Web Services

BG Blood Glucose

BVP Blood volume pulse

CGM Continuous Glucose Monitor

CNN Convolutional Neural Network

DCNN Dilated Convolutional Neural Network

EDA Electrodermal activity

IBI Interbeat interval

ICR Insulin to Carbohydrate Ratio

IOB Insulin on board

ISF Insulin Sensitivity Factor

ML Machine Learning

PH Prediction Horizon

ROC Rate of Change

T1D Type 1 diabetes

T2D Type 2 diabetes

TFLite TensorFlow Lite

UI User Interface

UTC Coordinated Universal Time

Chapter 1

Introduction

1.1 Project Motivation

Diabetes is a chronic disease becoming increasingly prevalent globally. The World Health Organisation estimates that in 2014 8.5% of the adult population were living with Type 1 (T1D) or Type 2 (T2D) diabetes [1]. In T1D, the pancreas does not produce the insulin required to properly regulate blood glucose (BG) levels, making insulin injections necessary for survival. In T2D, insulin sensitivity prevents the body from using the insulin produced effectively. While T2D can be reduced or prevented through healthy lifestyle choices, T1D cannot currently be prevented. In the short term, hypoglycemia, characterised by low BG levels, can cause issues ranging in severity from dizziness to causing a coma. HbA1c is a limited measure of a diabetics average BG levels over the last 3 months, which correlates well with their risk of long-term diabetes complications [2]. These complications are numerous and include life-changing conditions such as retinopathy, coronary heart disease, and stroke [3]. According to a 30 year study on diabetes complications, "An intervention that aimed to achieve glycemia as close to the nondiabetic range as safely possible reduced all of the microvascular and cardiovascular complications of diabetes" [4]. Managing to maintain BG levels close to a nondiabetic range is not a simple task, as a range of complex physiological factors can affect BG levels, sometimes in ways that are not yet well understood or that vary significantly between individuals[5]. As such, systems which can help educate or enable a user to make better decisions could have a significant effect upon improving the lives of those living with diabetes.

1.2 Project Definition

The goal of this project is to produce a smartphone app capable of assisting a Type 1 diabetic user to manage their condition through collecting some information and providing tools to support decisions. While many aspects of the system are

also applicable to T2D management, the incorporated machine learning systems are trained on T1D patient data and due to significant biological differences will likely not generalise well to T2D. This builds upon a previous ARISES project in which the basic user interface (UI) was implemented by continuing to improve the user experience and incorporating sophisticated models for BG level prediction and insulin bolus calculation. Some aspects of this project are shaped by a current focus on developing this system for use in clinical trials, enabling the effective testing of models and algorithms while collecting a rich dataset of lifestyle and physiological information to use to develop future systems.

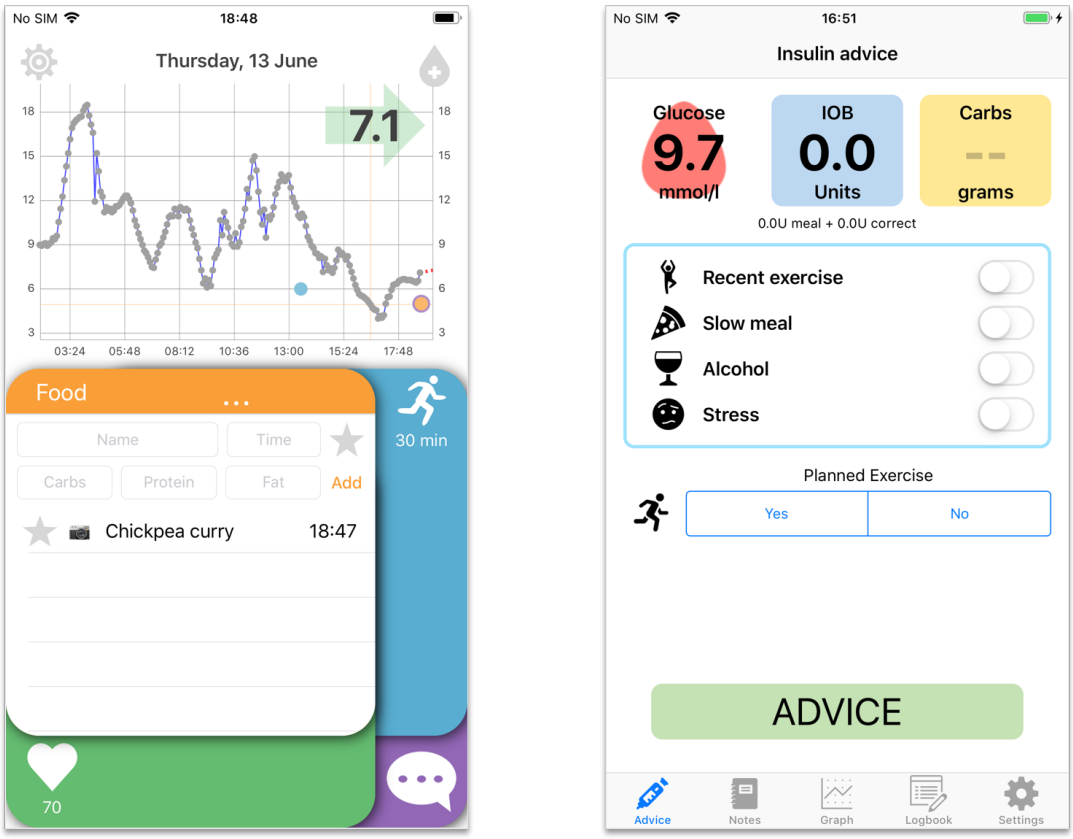


Figure 1.1: ARISES (left) and ABC4D (right) main app screens

Repeated reference will be also made to a similar app being developed at the same time, the Advanced Bolus Calculator For Diabetes (ABC4D), as due to their similarities a significant amount of work is transferable. The projects also vary in their implementation details, offering an interesting opportunity to compare their approaches to similar problems. Figure 1.1 shows the UI for the main screens of both apps.

One challenge to creating decision support systems is capturing all the factors which are most important to modelling the complex system. When designing software for this project this leads to several different systems for collecting user data,

from connecting modern sensors such as the Dexcom G6 CGM and Empatica E4, to asking a user to input some physiological information and track their lifestyle. Ensuring these separate systems are appropriately coordinated to store all this information and utilise the embedded algorithms and models is a significant part of this project.

1.3 Report Structure

This report will begin with some background on the state of existing algorithms and models for diabetes management, with a focus on prediction. This will include the specific prediction model used within the app, and some detail about bolus calculators. Following this, the requirements chapter will outline the initial specifications for the systems and the process of compiling them. The subsequent chapter on analysis and design will explore the realisation of these specifications into the system at a high level, explaining any trade-offs made. The following chapter give a much more detailed explanation of how these features are implemented. After this, the report will explain the steps taken to test each aspect of the system and a note on the results of development. Finally, the report will conclude with a comprehensive evaluation of the system as whole and summarise the future work required on this project.

Chapter 2

Background

'Artificial Intelligence Methodologies and their applications to diabetes'[6] gives descriptions of methods for various applications in the field of diabetes management, such as recommendation systems, bolas calculators, diabetes diagnosis and many more. In this particular project the primary challenges in diabetes management that we seek to assist with are blood glucose prediction and bolus calculation.

Other interesting problems in the field such as detection of diabetes and exercise recommendations systems are less relevant to the use case, as a diagnosis of diabetes is already confirmed and exercise recommendation systems are much more useful for Type 2 diabetes management. Improved blood glucose (BG) control is the main aim. Prediction and effective bolus calculators are very useful tools for pursuing that goal.

2.1 Blood Glucose Prediction

BG prediction is usually formulated to provide a prediction horizon (PH), a forecast of future blood glucose levels at a specific time, often 30 to 60 minutes. It is typically argued that a window of this length could provide a diabetic with enough warning to take steps to avoid many occurrences of hypo or hyperglycemia and enable them to improve their control in general. These prediction models face a number of challenges. Non-linear time series modelling has an inherent complexity, but in the context of blood glucose prediction a further level of complexity is added by its dependence on biological factors [7]. Modern methods attempt to incorporate some extra features, usually collected as life events or physiological states, to improve prediction beyond what is capable using only previous BG levels. A number of other challenges also limit the capability of these models, typically involving the quality and accessibility of data on which to train and assess them. The following section will cover some of these problems and typical solutions applied in several modern papers.

2.1.1 Datasets

The first barrier to developing a model is procuring a dataset. The dataset chosen immediately creates some limitations for development. There are few datasets containing CGM data and biological features, many research departments collecting data for their own use. This severely limits the ability to compare models, as results stated in papers are tested on vastly different datasets. The Ohio T1DM [8] dataset, released in July 2018 during a Blood Glucose Level Prediction Challenge, is available to researchers on request now and may become a standard for evaluation for some time. Due to its release as a challenge there are already a few papers describing models trained using it. This project also has access to the dataset collected as part of the ABC4D project.

The size of the dataset may limit the complexity of models, as more complex methods often require more training data to prevent overfitting [9]. Feature selection is limited to the feature set provided and any higher level features which can be derived, potentially stifling otherwise promising models. In several papers, for example, [10], [9], [7], a delay between raw and predicted values is visible during fast variations in blood glucose levels, usually after life events such as insulin injection. This is likely due to a lack of features containing sufficient information to model these biological events explicitly.

2.1.2 Missing data

For the convenience of a user, CGM values are typically transmitted by Bluetooth to a nearby device for storage. This provides a minefield of potential cases where CGM values are not stored. Some of these are rectified by good CGM design, such as storing un-transmitted values in a backfill and transmitting these upon reconnection. Backfill storage is typically limited though, for example in the Dexcom G6 it is only 3 hours. So if a user's phone runs out of battery or they are separated from it for more than 3 hours, some data may be lost. Other issues may also cause poor data, such as long sensor pairing times of new sensors, which need to be replaced frequently. As a result it's important to consider the effect that missing data might have on a specific model, as that can vary. For example, the DCNN described in Taiyu Zhu's thesis [11] uses the changes between current and future data points, causing discontinuities to have a significant effect. To resolve this, some level of imputation is used to replace missing values. It's also important to think about the effect missing data has on training and testing data individually. Missing training data will reduce the accuracy of a model, but imputing values could

introduce some bias [12]. It seems to be agreed that sections of training data with a large amount of missing points should simply be discarded however. [12] [11] Missing values in training data can be imputed using interpolation, to fit a point between known points, but this cannot be done in testing or real-world execution of the model as future points are unknown, meaning extrapolation must be done instead. While extrapolation provides some data values on which to predict it does not work well for significant lengths of missing data, as in 'A Deep Learning Algorithm For Personalized Blood Glucose Prediction' the "error is still high for the data after these regions. Because the predictive curve is calculated by adding the predicted differences onto the previous values, it heavily depends on the data 6 timesteps before". As data is fed into the model as a batch, once true future data values are collected, previous missing data which is still being used in the model could then be retroactively replaced with interpolated values.

The methods used for interpolation can vary in complexity but it is generally the case that when data is input into a complex system such as a neural network, simple first-order interpolation is similar in performance to other methods and in Taiyu Zhu's Thesis this performs best. In the GluNet paper[13], spline interpolation is used, for accuracy and simplicity. Interpolation and extrapolation are intuitively only applied to CGM and potentially other time series inputs, as discrete information such as carbohydrate intake cannot be reasonably inferred in any way. As such, where life events are missing, they might have a significant effect on the error, but this is hard to determine without purposefully omitting some events and comparing the accuracy.

As BG prediction is typically trained on an individual's data, when a large amount of data is missing it can have a negative effect on the accuracy for that individual, as if sections of data with missing values are removed then less data is left for the model to train on, reducing its ability to generalise. Some transfer learning techniques which are typically used to help a model generalise by allowing it to initially learn features from a similar but not identical problem, can be used to supplement a patients data with data from other patients. This can extend the size of the dataset and may improve accuracy, but a balance must be struck between the amount of data used to help the model generalise to the glucose prediction problem while still optimising to prediction for the individual.

In 'Dilated Recurrent Neural Network for Short-Time Prediction of Glucose Concentration' they "use 50% of the target subject's data plus 10% of other subjects' data to train the model first, and then train the final model based on the whole training set of the target subject.". In the GluNet paper [13], useful predictions

were reported even when testing a model that had not been training on any data from the target subject.

2.1.3 Selecting Features

The modern and increasing availability of continuous glucose monitoring (CGM) devices and methods is a boon for diabetics, minimally invasive systems with sensor readings that update in minutes provide them with a much greater picture of the variations of their blood glucose levels throughout a day. While this is a huge step forward, even with this information, the effect that various factors and their combinations have on blood glucose levels is too complex to simply be extrapolated by a user to predict future BG levels. [14]

These factors are numerous, and many are still poorly understood. The most obvious and significant variations are after meals, physical activity and insulin intake. Other factors such as the effect of stress or hormonal fluctuations are more difficult to predict. Not only are these factors inherently complex, they also vary in their behaviour in individual patients. As such some "personal parameters such as the correction factor, insulin-to-carbohydrates ratio, and physical characteristics such as the patient's weight"[5] could be used to personalise a model to a single patient.

Physiological models exist that use sub-models created using an understanding of processes such as glucose metabolism. These models require several physiological parameters to be set and tend to be difficult to adjust[5]. Many purely data driven models such as Neural networks tend not to involve physiological variables. Some hybrid models exist, where physiological parameters and models are used to feed a data driven model. This seems like a reasonable way to incorporate physiological variations that are well known by patients, such as insulin-to-carbohydrate ratio to create a new feature that generalises better to the cohort, without requiring a model to be trained on a large amount of an individuals data.

According to 'Deep Learning for Health Informatics'[15], "studies have shown that there are no universally hand-engineered features that always work on different datasets. Features extracted using data driven learning can generally be more accurate.". This relies on being able to design a system to extract the most relevant features. For other ML problems, such as classification problems, Autoencoders do a good job at this when measures are taken to prevent overfitting, extracting the most discriminative features, while lowering dimensional space. Convolutional Neural Networks also perform a level of feature extraction, as "Each layer in the network originates a high-level abstract feature" [15], due to the network learning

filters. This allows the network to understand some context of relationships between nearby inputs. Dilated Neural Networks work similarly, but learn filters based on a wider global context by using a dilation convolution. It seems intuitive that as far as is possible, data driven features should be used over hand-picked features in complex problems such as glucose prediction with many contributing factors that may interact.

According to 'A review of personalized blood glucose prediction strategies for T1DM patients' [5] some "studies state that the use of additional inputs makes the prediction task harder because formalising these inputs in mathematical terms and extracting useful signals from them is not easy" and "some works suggest that ingested carbohydrate information, along with injected insulin information might be redundant", leading some models to use the CGM signal as the only input. It is clear from many papers that some level of prediction can be done with only CGM as an input. [16] [11] [10]. While the information contained in ingested carbohydrates and injected insulin may eventually become redundant as their effect begins to show in the CGM values and trend, it is unlikely that there is no value in this information. Potentially the value is not sufficient to outweigh the issues surrounding formalising them as inputs, or the variation included by self-reporting of carbohydrate estimates. Given that the distortion between glucose in interstitial fluid can cause temporal delay longer than 10 min for CGMs that use interstitial fluid [10], there is almost certainly useful information that can be used to improve predictions by knowing at ingestion, or even in advance, the carbohydrate and insulin intake of a patient. In 'Artificial Neural Network Algorithm for Online Glucose', they propose that "Integrating into the model quantitative information concerning, for example, insulin injections and meal intakes could be beneficial to the prediction" [10] to facilitate tracking of rapid changes in glucose that the model struggled to predict as effectively.

They struggled to add these features as they were limited by the features of their dataset, but according to 'A smartphone-based platform integrating AI for Adaptive Glucose Prediction' [11] we have a dataset from the ARISES project containing various data fields including insulin events and carbohydrate intake and the OhioT1DM dataset also includes insulin and carbohydrate intake information along with other fields. The results in that paper show a comparison to an ANN and the DCNN used does not have as much delay shown during sharp upward BG trends. Whether this is due to the inclusion of other features or the difference in architecture is hard to say though. They also show a distinct increase in accuracy when using features such as Glucose and Insulin over the model with only CGM input. A paper

describing a NN model with a first-order polynomial predictor used in parallel also shows improvement when meal information is incorporated. [17] It seems likely that the relatively recent resurgence of neural networks allows meal information to be utilised without great efforts to effectively formalise it as an input.

2.1.4 Overfitting

Overfitting is a challenging problem in developing many machine learning models, and the causes of overfitting can vary from training for too many epochs to an insufficient amount of training data or allowing models to be too complex. Obviously a combination of these also occurs, such as complex models with an insufficient amount of training data to appropriately generalise. Some previous reference has been made to some methods for reducing overfitting, such as supplementing the dataset with extra data in transfer learning. Other common methods include regularisation through dropout, a process by which a defined percentage of the output features of a layer of a network are randomly turned off. The individual weights of a layer can also be penalised for being too large using different regularisation functions, typically l1 and l2 normalisation. This is often applied by some experimentation, but some intuition can be used, for example in a CNN features are more generic in earlier layers and more specific to the dataset in later layers. Based on the results it may be clear which would benefit more from further regularisation.

2.1.5 Model Topography

There is some discussion on the use of models that classify BG into established ranges and then allow patients and physicians to make decisions based on how life threatening those ranges are. While predicting life-threatening conditions is very useful, a vague range of possible values does not empower the individual to understand their condition in as much detail and adapt their lifestyle to improve their control in the same way as a specific value prediction. A range of values also prevents recommendations for countering that condition from being specific. While difficult to compare, I cannot imagine there is an increase in accuracy with these methods that outweighs the utility of a predicted value.

Interestingly, in 'A smartphone-based platform integrating AI for Adaptive Glucose Prediction'[11], the network is treated as a 256 bin classifier, with a softmax output layer to provide predictions of blood glucose change. Some trade-off between the number of bins is considered in 'A Deep Learning Algorithm For Personalized Blood Glucose Prediction' which says "The number of classes is chosen carefully

because a small number of classes cannot distinguish the difference while a large number of classes are not suitable for small training dataset".

For methods where quantisation is done on BG changes [11] [9], the likelihood that a glucose value changes by a larger amount is less than that if it varies by a smaller amount within the 5 minute timestep. Increasing the specificity of lower changes might improve accuracy a minor amount in values that are well predicted. This could be done through a data-driven method such as an autoencoder which would derive a transform that would optimise reconstruction from a lower dimension form, learning where the specificity is required. The majority of error in these models however, is in fast changing BG values that the model cannot predict, so overall this could not produce more than an incremental improvement. Something similar is done in the WaveNets paper [18] where a μ-law companding transform is done to an audio signal to improve accuracy in the dynamic range that is most significant in human speech.

'A review of personalized blood glucose prediction strategies' includes a variety of examples of models for predicting BG levels, but limited information to compare their effectiveness. All model examples provided are also from 2015 or earlier due to its publication in 2016. 'A smartphone-based platform integrating AI for Adaptive Glucose Prediction'[11], however, provides a description of a dilated CNN, which is superior to traditional ARMA models of the past, which cannot model the non-linearity of effects such as glucose-insulin interactions.

The model which is embedded in this app currently is a version of the model described in the GluNet paper. It is based on the Wavenet [18] and PixelCNN architectures [19], and "consists of several components, including data preprocessing, label transform/recover, multi-layers of dilated convolution neural network (CNN), gated activations, residual and skip connections."[13]. Tested *in silico* on UVA/-Padova simulator data and against the Ohio and ARISES datasets, it appears to provide state-of-the-art performance with reduced time lag and RMSE.

2.2 Bolus Calculation

Determining the amount of insulin required to offset BG levels with careful consideration of other life events, such as meal intake, can be complicated. The most standard formula for calculating a bolus can be seen below. This is taken directly from 'Method for automatic adjustment of an insulin bolus calculator'.[3]

$$B = \frac{CHO}{ICR} + \frac{G_c - G_{sp}}{ISF} - IOB \tag{2.1}$$

"where B is the recommended dose of insulin (IU); CHO is the total amount of carbohydrate in the meal (g); ICR is the insulin-to-carbohydrate ratio (g/IU), which describes how many grams correspond to one unit of fast acting insulin; Gc is the current capillary blood glucose level (mg/dL); Gsp is the blood glucose set-point (mg/dL); ISF is the insulin sensitivity factor (mg/dL/IU), and IOB is the insulin-on-board which describes the amount of active insulin remaining from previous injections."

While this is a relatively simple mathematical equation, for many diabetics this is too complicated or inconvenient to calculate at every meal and correction, leading to even more simplified equations and a level of intuition. The use of a verified bolus calculator could improve control by reducing under or overshooting a BG target. This simple formula also fails to account for any exercise done.

Developing effective bolus calculators is significantly easier with the use of a simulator to test on. The PADOVA Type 1 Diabetes Simulator [20] is FDA-accepted and reasonably suited to this purpose. According to 'Method for automatic adjustment of an insulin bolus calculator', it does lack intra-subject variability and uncertainty, but modifications can be made to simulate this.

As the basic formula tracks some individual physiological features such as insulin-to-carbohydrate ratio, it requires specifically setting to a user and may require adjusting based on changes in insulin requirements. citeautomatic adjusting Attempts to adjust these automatically have been shown to be safe in the ABC4D pilot study in 2016 [21]. The ABC4D app shown throughout this report is part of preparation for a 6 month study to determine the effect of the case-based reasoning adaptation algorithm within the app on user's HbA1c.

2.3 Implementation Background

Several frameworks exist for embedding ML models on iOS. Using Apple's own 'Core ML' framework would be the easiest and fastest way to implement predictions and would also take advantage of optimisations that accelerate their speed using the GPU. Keras, Tensorflow and Caffe models can be converted to be used with this framework, but the operations and models that you can convert are limited in complexity and customisation options. If this does not have enough features other frameworks such as Tensorflow Lite may be used, however these may have poorer system resource usage. Tensorflow Lite specifically, should allow for the use of the pre-trained frozen models currently used in the Android version of the ARISES app.

Lower-level implementations methods can also be used to embed ML models.

These include options such as other open-source frameworks, accessing C++ libraries by bridging the languages in the system, working with Apple's Metal Performance Shaders Graph API or using the Metal framework (similar to OpenGL) to interact with the GPU at a much lower level. All of these options have high development times due to innate complexity and the limited documentation on utilising the required languages and architectures for this purpose.

Chapter 3

Requirements

At the beginning of the previous project, building the original UI for the app, some basic specifications were provided, and these developed over time through discussions and focus groups. The original design provided by Professor Bob Spence for the ARISES app is shown on the left in Figure 3.1. This encompasses a few main design philosophies, primarily that there should be no navigational hierarchy so that all content should be accessible with one click and all important information should be visible at all times. For reference, the initial design for the ABC4D app is shown on the right of Figure 3.1, and can be seen to take a more traditional approach, using a stock tab bar for navigation.

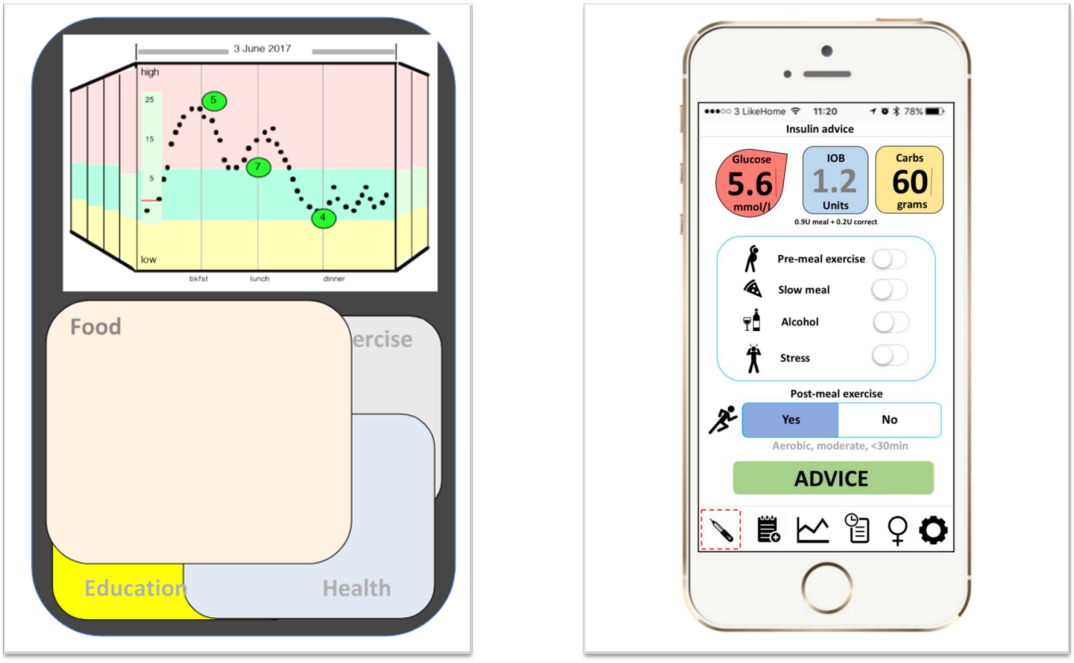


Figure 3.1: Original layout designs for ARISES and ABC4D apps

A very high level specification for features of the app stated the 4 domains would be: Food, to allow user to input meals and their nutritional information; Exercise, to input exercise duration and time; Health, to let the user view trends and enter miscellaneous information such as illness; and Education, to provide advice and links

to new information about diabetes management.

Focus groups with people living with diabetes provided extremely relevant advice that helped refine these design choices and features by providing feedback and suggestions on specific designs, as well as improving our understanding of their general needs and desires from a system like this. An example of specific feedback was to ensure that the most recent CGM value is large and visible as well as including a trend arrow.

One key takeaway was that when designing an app that we expect to be used upwards of 3 times a day, it's important to focus on minimising the inconvenience to the user. Anything that takes longer that strictly necessary or is not intuitive to perform will quickly become irritating due to such frequent use. As an example of this, several animations for navigating between domains with small delays were trialled, but rejected by the focus group. Similarly, it was repeatedly highlighted that any notifications and alerts should be kept to a minimum and be capable of being turned off. Some concern was also expressed about the amount of information that is being displayed and how to appropriately maximise the use of space without making it too crowded or difficult to see. This is especially important from an accessibility standpoint, considering the prevalence of diabetic retinopathy.

Another important point raised was that the uptake of these systems requires us to be able to earn the trust of users, as we are asking them to make decisions about their health based on our predictions and suggestions, so it is important that the user interface is also stable and bug free as this affects the perception of the entire system. It is also important that the suggestions made appear as transparent as possible, as diabetic users will have some expectations based on experience with their condition and may reject differing suggestions if it's not clear how they were made. Some examples of this are desiring breakdowns of bolus calculations and their strong interest in being able to see how potential decisions might affect the prediction dynamically.

A significant insight was that the premise behind that education domain was of limited value because, particularly in clinical trials, users of the app will have undergone specific education before receiving the app. Subsequently the domain was renamed advice and left as a space for potentially providing some identified trends in a user's data or warnings.

The scope of the project described in this report prioritises the functionality of the app over form, so advanced UI design choices such as alternative input methods are not explored. Instead, changes focus on the most impactful areas: implementing connections between devices, embedding the ML system, incorporating the bolus

calculator and UI changes that significantly improve usability. All new systems and changes were created with consideration of the design priorities highlighted by the focus groups.

Further to the user-oriented requirements, the planned use of the system in a clinical trial creates further requirements in how user's data is collected, stored and collated. It is desirable for this data to be uploaded and available remotely. This serves multiple purposes: it can be reviewed during the trial if a user is encountering a problem with the app, and at the end of the trial the data can be analysed to potentially determine some information about the success of the trial and develop new features and systems.

Chapter 4

Analysis and Design

The specification for this app can be explained as a pipeline of data, beginning with collection from sensors on the user, through storage then processing and culminating in app features that empower them to improve their BG control.

4.1 Data Collection

Data collection is done in three main ways. The most vital data is the continuous glucose monitor (CGM) information collected via Bluetooth from a Dexcom G6 [22], which provides the user's BG level and trend. Other physiological data is collected from the user via the Bluetooth Empatica E4 wristband. The user also enters some information regarding life events into the app via a diary. Figure 4.1 shows the sensors and diary entry domains. Figure 4.2 shows the ABC4D app with the data collection devices used in that project.

Figure 4.1: From left to right. Empatica E4 wristband, Dexcom G6 CGM, ARISES diary

Requirements for data collection are primarily focused on performance, ensuring that Bluetooth device connections are as stable as possible and that software is designed to handle disconnection events and any errors transmitted. The goal for performance will be replicating the stability of the apps provided by the manufacturers of the CGM and Empatica E4.

Figure 4.2: ABC4D app with some data collection devices

4.1.1 Continuous Glucose Monitor

The Dexcom G6 CGM has been fully integrated into the ARISES app, such that so long as the Dexcom app is receiving data and the ARISES app is open in the background, it should also be receiving data. The Dexcom app takes care of calibration and pairing of the CGM, which allows the app to only require the user to enter the serial number of their CGM to begin storing incoming values to the database. Running both apps simultaneously also provides a level of redundancy to ensure all data is saved. Another advantage of also running the Dexcom app is their comprehensive alarm system for alerting users of dangerous BG levels, favoured by Focus group participants due to how customisable it is, so the ARISES app does not implement it's own notifications.

The CGM connection has been well tested and the Dexcom app shows the same BG values and trend as the ARISES app. The ARISES app also handles the backfill from the transmitter, which is the storing of BG values on the CGM in cases where the signal between the CGM and device are lost, and later transmitted when the connection is re-established. While the storage capacity of the device's backfill is not well documented, it appears to be capable of storing and transmitting several hours of missing data, more than enough to handle most disconnection cases. In the case where the ARISES app is not open or in the background and the device uploads

backfill to the Dexcom app, then the ARISES app will not be able to retrieve this data as it only monitors data being sent to the Dexcom app and the device deletes the backfill after transmission. As such the user should always keep the ARISES app open in the background. There is no way to improve on this using the frameworks available, if the Dexcom app is used simultaneously.

4.1.2 Empatica Wristband

To connect the Empatica E4 wristband a user must enter the settings page, hold the button on the wristband to enter discovery mode and then select the device from the table. The user must have connection to the internet to be able to complete this process as Empatica require that connections to the device be authenticated against their server.

Once connected, the phone then stores a large amount of data received in real-time. This includes user temperature, accelerometer information, blood volume pulse, interbeat interval, electrodermal activity and any tags created by the user to mark events. Heart rate does not appear to be transmitted by the device, but this can be calculated using interbeat interval and blood volume pulse. Currently all this information is stored, but due to the large capacity of data, in the future a sub-sample of this data may be preferred. The device allows detection of whether it is on a user's wrist, so in the case of the device being off the wrist, the data is not stored.

Stability is important for these connections and significant effort was made to test and ensure this in the Empatica integration. Unfortunately, it seems that the development system was not created with long-term use in mind. The combination of requiring the user to manually set the device to discovery mode, server authentication, poor connection when low on battery and a very locked-down developer framework prevents automatic re-connection of the device. Even in their proprietary app, E4 realtime, there is no re-connection system, so once the device disconnects, the user must manually reconnect. Currently the best workaround available is to notify the user of a disconnection and ask them to reconnect. The device can remain connected for a reasonable number of hours, but will disconnect at distances greater than 6 meters or when the battery of the device gets low. A warning is also implemented for when the device is low on battery, prompting the user to charge the device.

The Empatica realtime app cannot be run simultaneously to provide redundancy in data storage due to the connection system and is not designed to store user data throughout a whole day. The current system is not ideal overall, and to prevent data

loss during clinical trials where the device is only being used to collect data rather than in prediction algorithms, asking participants to use the device in recording mode instead and uploading this data at the end of the day might be a better solution.

4.1.3 Diary Entry

Within the app's data entry systems, care has been taken to ensure only the expected type of data can be entered, so that entry fields expecting numerical values will only receive numerical values. This is done by carefully selecting and implementing the most relevant data input method, such as a keypad, keyboard or date picker for the specific field. This prevents incorrect or odd data entry from crashing the app. Some care has been taken to ensure values entered remain within specific expected ranges. In the ABC4D app alerts are used to notify the user if they attempt to input a value outside of an expected range. This could easily be transferred to the ARISES app, and simply has not been done yet due to it's lower priority than more important features. The domains of the app are shown in Figure 4.3.

The app allows the user to add meals and exercise at any time throughout the current day, including retroactively, to keep the data correct, which is useful for keeping the prediction model accurate. Currently there is no functionality for editing or deleting previously added meals and exercise. This is relatively simple but somewhat time-consuming to implement, so was not prioritised in development.

Insulin input is currently handled in a new way, as part of the bolus calculator system, where bolus suggestions are provided when a meal is added or a correction is requested. The comparison to the previous method of inputting insulin is shown in Figure 4.4. Insulin input is currently added at the time of input, but should probably be adjusted to allow users to retroactively add and edit the stored logs of insulin given. It should be less likely that a user forgets to input an insulin dosage, however, as hopefully they will be using the system to calculate the bolus. A user is not limited to only inputting the recommended bolus, and can override that value to ensure that users input the actual values they take and provide accurate information to the system. Focus groups participants highlighted that without this, users would simply lie to the system when they wished to take a different dose based on experience.

Figure 4.3: Each ARISES Domain

4.2 Data Storage

Data storage involves carefully persisting a large amount of information in various forms. CGM data is received every 5 minutes and contains a BG value and trend, allowing its collection over a large period of time to be stored on the device without using a significant amount of storage space. Information that the user inputs is also small, typically recording only a few events per day. The Empatica E4, however, provides a large collection of time-series sensor data, as seen in Figure 4.5, that cannot be stored for a long time due to limited device storage capacity.

The performance of a database can directly affects a user's experience. The UI displays a variety of information that is either directly fetched from the database or calculated using database values. As such, any delay in retrieving this data may be transferred into delays in the app. Currently no noticeable delay is visible when accessing data in any part of the app, even in the health section when fetching the

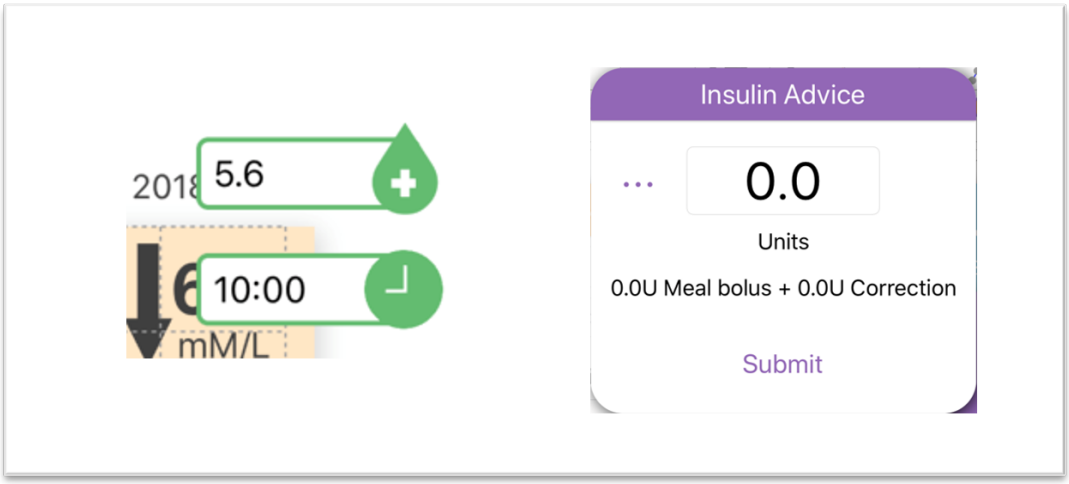


Figure 4.4: Insulin entry system. Left: old system, Right: new system with bolus calculator recommendations

previous 60 days of lifestyle data.

Required storage space is also important, as smartphones with increased storage are typically more expensive, an app that required a large amount of storage could reduce its accessibility to some and provide a poor experience to others by potentially limiting storage for other common smartphone features such as taking photos or playing music. As the Empatica E4 collects a large amount of information, this cannot all be stored on the device. Once uploaded to an online storage system this data should be deleted. Currently this is not implemented as no online storage system has been fully developed. In a production use of this, rather than for a clinical trial, the app could retain only the last couple hours of data for use in prediction, but in clinical trials it is desirable to be able to store and collect all this information.

One significant issue with the storage method initially implemented is it isn't well set up to handle time zones. This was noticed during ABC4D user testing, and causes a number of problems. As all dates are stored in universal time (UTC), when viewing this data later it becomes impossible to determine when time zones change causing all data to be offset from the local time it was collected and losing the context of that time. The original relational database system, wherein all objects are related to a specific day also becomes problematic when the beginning of a day shifts in different time zones. In the ABC4D app this has been rectified with significant database changes, storing local time as well as UTC and adjusting database fetching systems to run based on time relative to now, rather than based on a day. These changes can be transferred into the ARISES app, but this has not currently been done yet due to the scale of work required in comparison to the value of handling

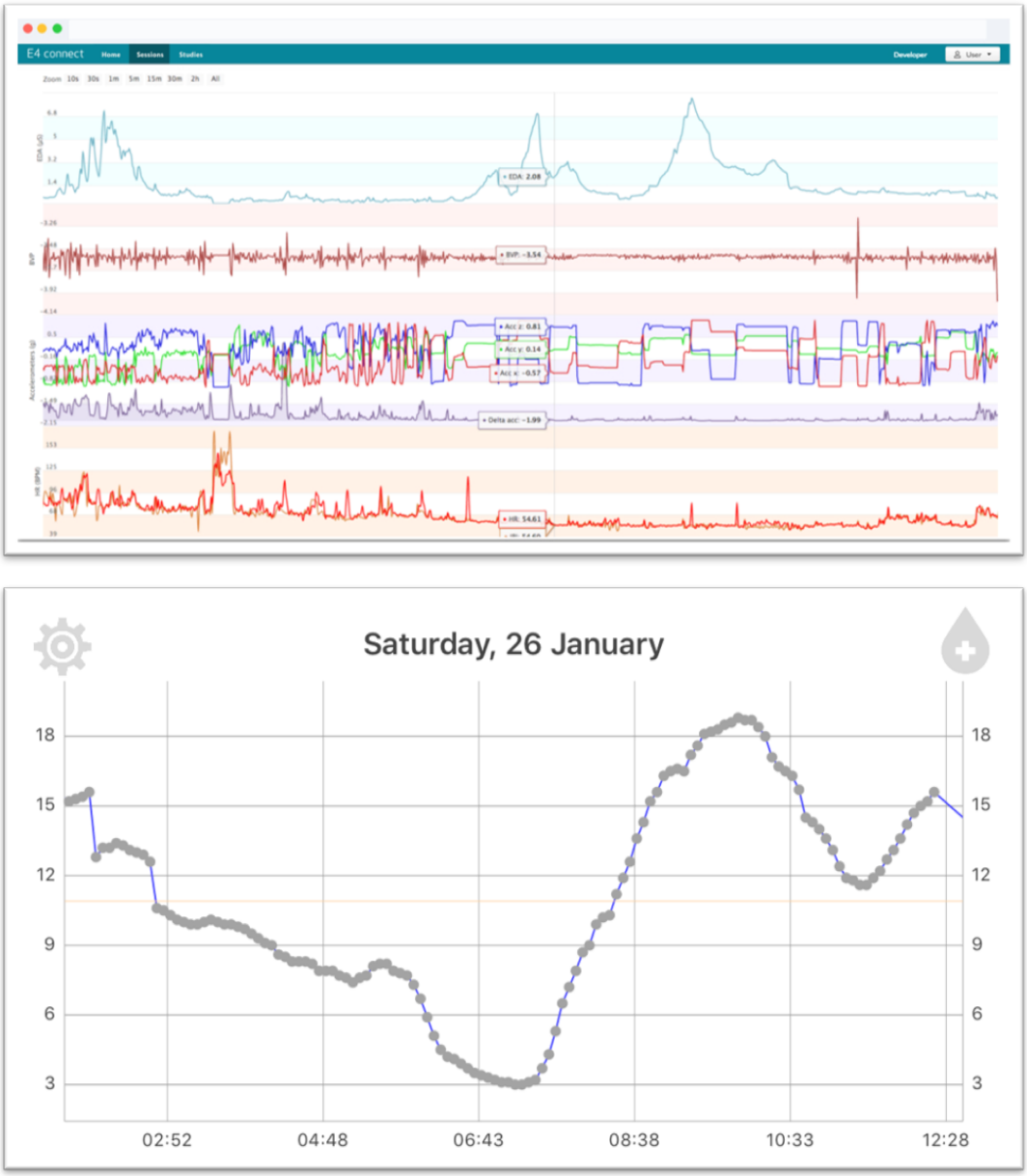


Figure 4.5: Empatica data (Top) compared to CGM data (Bottom)

the specific cases where clinical trial participants travel abroad, reducing its priority. A slightly different method for tracking time zone will be necessary in the ARISES app storing local time on each Empatica data object was tested and dramatically increases the amount of storage required.

4.3 Data Processing

Most of the power of this app as a tool for diabetes management lies in the embedding of sophisticated models that provide useful insights. Many other apps are capable of collecting and visualising information but the complexity of physiological factors and

their interactions limits the usefulness of human conclusions. This app specifically will incorporate a state-of-the-art deep learning model for BG prediction. A bolus calculator will also be added to provide insulin recommendations.

4.3.1 Blood Glucose Prediction

The overall aim is that the BG prediction model provides a wide enough prediction horizon (PH) to enable users to see future BG levels with enough time to act to rectify potential hypo or hyperglycemic events, even during the significant rates of change following life events such as insulin injection. The performance of the chosen model and its implementation on the smartphone is important, as while a model may have PH of 30 or 60 minutes, any delay on calculating that prediction reduces the effective PH.

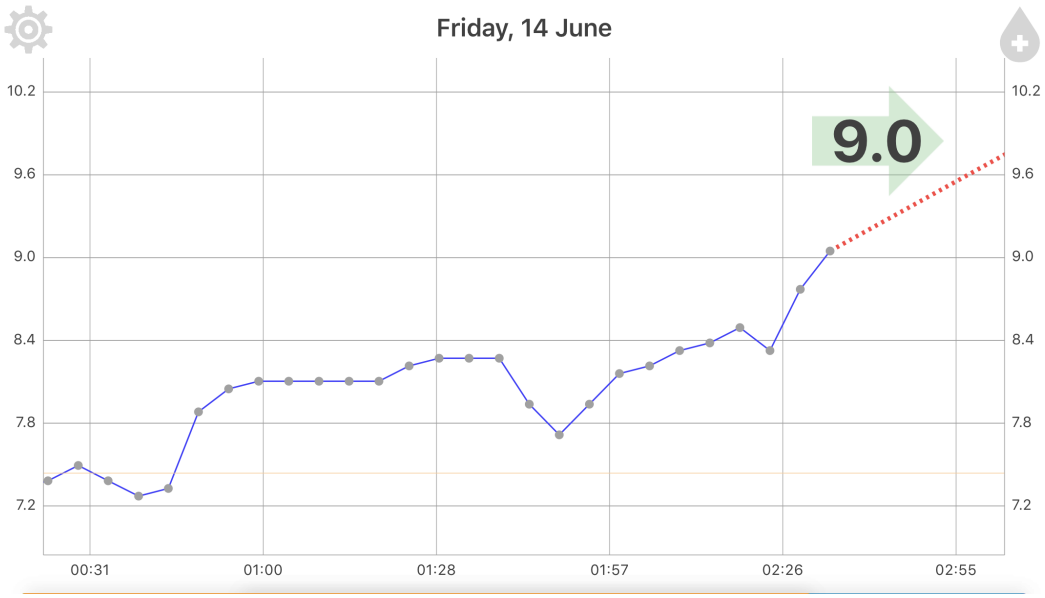


Figure 4.6: Screenshot of graph from the device, displaying a prediction line

The model which is implemented on the device is a version of the GluNet model described in the background chapter, with a 30-minute PH. An example of this prediction is shown in Figure 4.6. This version only provides a single value prediction, preventing useful confidence intervals from being displayed on the graph. This could easily be replaced with any model with identical input and outputs, by replacing a single file. Similarly, only minor changes would be required to implement a 60-minute PH model as well. If any CGM values are missing from the required window, linear interpolation is done to provide the required values. If a significant amount of information is missing, the user should be warned that the prediction is less reliable, but this alert is not yet implemented.

The model and its implementation runs fast enough to allow the user to dynamically change future carbohydrate or insulin values and see the effects in the prediction with no perceivable delay.

4.3.2 Bolus Calculation

Calculating an appropriate bolus is typically based on carbohydrate intake, blood glucose and activity level [23]. Most diabetics will be aware of a basic method for calculating a meal or correction bolus using their insulin-to-carbohydrate ratio and correction factor, however a verified bolus calculator may remove calculation error and is definitely more convenient. This project includes a bolus calculator with some more advanced features based on the calculator used in the ABC4D project, converted from the MATLAB implementation created by Dr. Pau Herrero.

This bolus calculator requires the user to input some specific settings, to personalise the calculation to their physiology. The settings page for the ARISES app can be seen in Figure 4.7. The ABC4D calculator makes adjustments based on exercise, insulin on board, the type of meal (e.g. Breakfast, Lunch), menstrual cycle and a postprandial variable BG target. The majority of these features are retained in the ARISES version, however to limit the required settings a user's menstrual cycle is not taken into account and some small implementation simplifications are made such as using fixed times for classifying meals as breakfast, lunch or dinner. This calculator is more than sufficient for the use case but could be adjusted based on more discussion about the specific factors and settings that it would be desirable for it to account for.

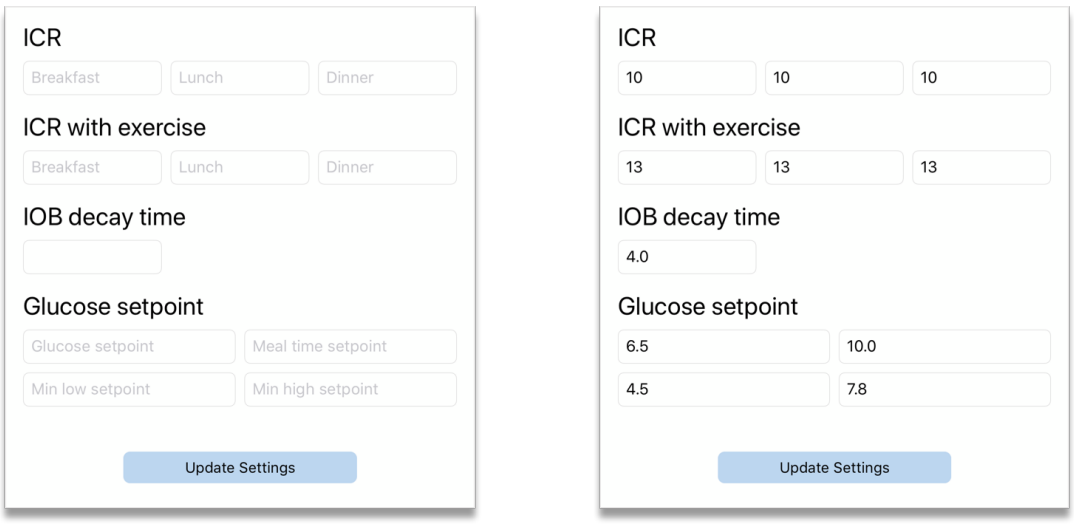


Figure 4.7: Left: Settings required by the bolus calculator, Right: Some typical values

4.4 Data Visualisation

In this project sophisticated prediction models are of little use if their conclusions cannot be effectively communicated to a user. As such, summarising a user's data and highlighting the most important information is key.

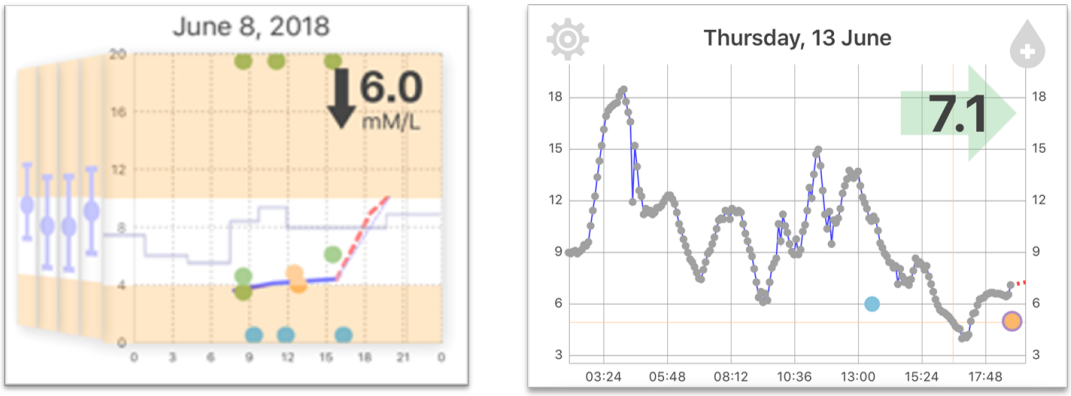


Figure 4.8: Old ARISES graph (Left) compared with new graph (Right)

4.4.1 Display

The app contains a graph displaying a user's BG levels for the current day. This replaces the previous graph, as shown in Figure 4.8, which had significant performance issues following an iOS update. The new graph also allows a user to zoom in on different time periods throughout the day. This is key to be able to view predictions, shown as a dotted line, at a reasonable size. A significant re-layout also allows the graph to be displayed in landscape mode, taking up the entire screen, as seen in Figure 4.9. Points plotted on this graph are shown with a dot rather than just a line so that it is clearly visible when data points are missing and the line is an interpolation. Previous days can be displayed by selecting the day after tapping the date at the top of the display, or using the health domain table to view a specific day.

One possible improvement would be showing a nondiabetic or target BG range with a different color, as the old graph did. While possible, this is somewhat difficult as because the graph can be zoomed and scrolled, the colour range must follow the new axis positions. Attempts made to do this were not as smooth as desired. The previous graph version had a bifocal display showing some contextual summary information about previous days. As this is not required for functionality, based on the amount of time to develop the extra graph and transforms to produce this effect, this was not implemented.

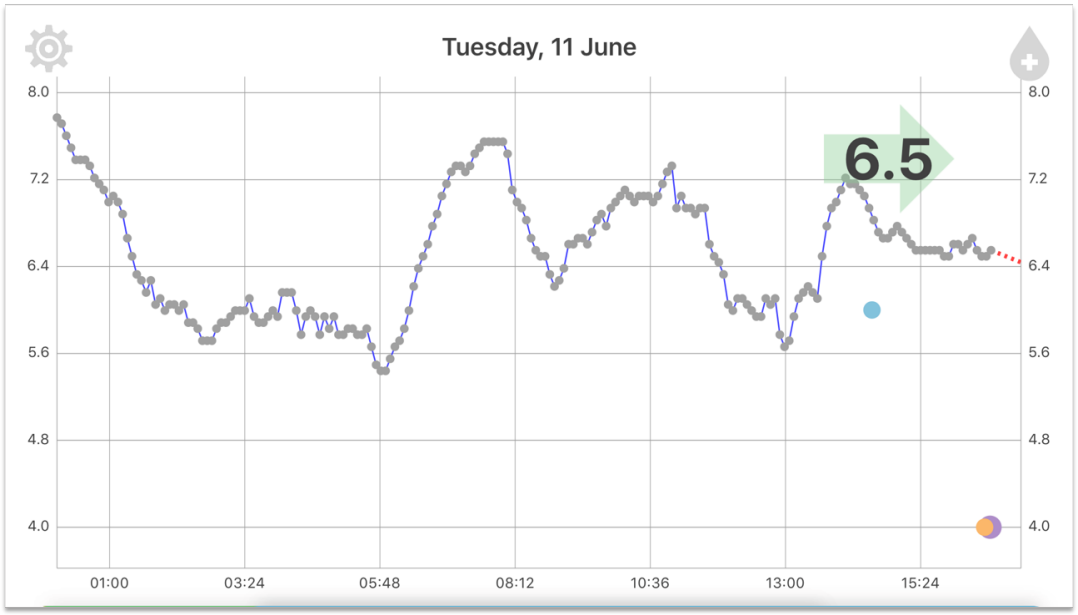


Figure 4.9: Horizontal graph

While it is important to view the information displayed on the main graph, this limits the space remaining to view information displayed in the domains, such as the health section. Particularly on smaller screens, this can become unpleasant to use. Focus group participants particularly noted that in places the app was dense with information. To help alleviate this, a new feature was developed allowing user's to adjust the size of the graph and diary domains shown. By moving their finger up and down on the three dots in the centre of the screen, the amount of screen space devoted to each part can be customised. This value is stored on the device so it remains in this position until changed. Examples of this adjustment are shown in Figure 4.10.

The user's most recent BG value is displayed in the top right corner of the graph, along with an arrow representing its current trend. The arrow position is always identical to that shown in the Dexcom app, except for the double up or double down arrows which are simply displayed as up or down arrows. The possible trend arrow positions are shown in Figure 4.11. In the ABC4D app, the most recent value is only displayed if it is less than 15 minutes old to prompt users to enter their own finger prick BG values to allow the system to provide accurate bolus calculations. Applying a similar process to the ARISES app would be helpful so that users are made aware that the displayed value is not recent, but has not been done. A user's IOB could also be displayed here, utilising the system implemented in ABC4D to calculate and display the IOB on the main view appearing. This was something some focus group participants were keen on seeing, although it was highlighted that

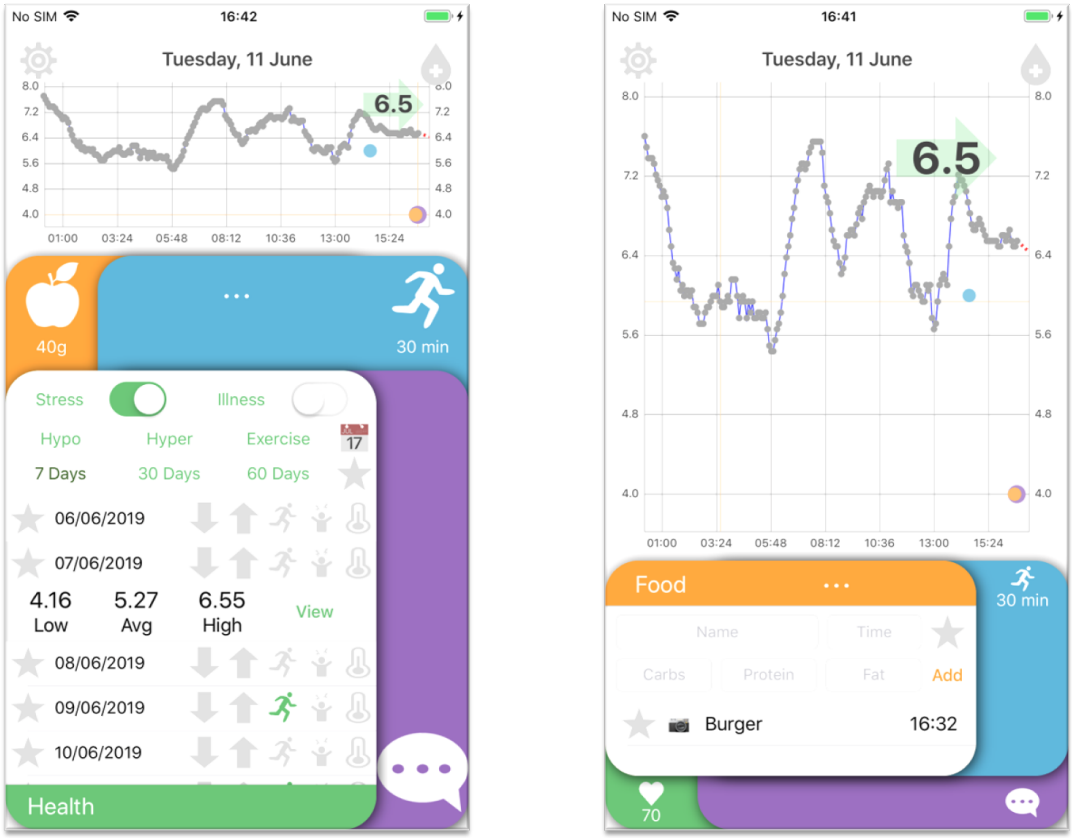


Figure 4.10: Adjustable graph and diary sizes

this is not something everyone will necessarily want and should be optional, by allowing a user to customise the amount of information they wish to see.

Figure 4.11: Possible trend arrow positions

When meals, exercise or insulin events are added, they will be displayed on the graph as a dot with their corresponding colour, to provide the user with a greater understanding of the effects of life events on their BG levels. Insulin is displayed at a position corresponding to the number of units taken. Meals are displayed as the value of carbohydrates divided by 10, to give some idea of the amount, while remaining within the typical CGM data range. Exercise is displayed based on intensity, with 6, 7 and 8 corresponding to low, medium and high intensity.

Small indicators on each tab show useful summary information relevant to that

page, for example the food tab displays the total carbohydrates consumed on that day. Other indicators are not currently connected to database information in the same way.

4.4.2 Domains

All of the domains now contain tables displaying logs of events such as food, exercise and insulin. The health domain contains a table, shown in Figure 4.12, which displays summary information from previous days which can be filtered based on specific tags such as incidents of hypoglycemia. Any number of tags can be combined to investigate their relationships. Tapping on a cell in the table expands it to display the low, average and high BG value for that day, as well as a button which allows you to view that day. Clicking this button brings that day's information onto the graph and domain tables, to be investigated in more detail. Similarly to the food and exercise tables, days displayed in the health table can be favourited, allowing easy access to interesting days if a user wishes to show a clinician, for example.

A key feature discussed in focus groups was the ability to explore the effects of different decisions dynamically, for example, seeing the effect of a certain meal on BG levels before choosing it. This has now been implemented in the carbohydrate field in the food domain, so when a user enters a value the prediction will instantly be adjusted to display the effect of that meal. Similarly, when inputting an amount of insulin taken, a user can explore other options than the recommended value by swiping their finger up and down on the three dots in the insulin view, as shown in Figure 4.13, instantly seeing the effect on BG prediction reflected in the graph. Currently these are the only factors which can be adjusted to affect the embedded model, but if a model was sensitive to other factors, this could easily be extended to display the effect of varying those.

4.5 Data Uploading

In the ABC4D project, a web interface, created by Dr. Bernard Hernandez, allows data for each participant to easily be monitored over the course of a trial. Figure 4.14 shows a graph from the server displaying some data collected and uploaded automatically during testing. Data is uploaded to this website using its APIs, and the system can even be used to update settings on a user's phone or send notifications. This is required for that project, to ensure that clinicians can monitor the adjustments that the system makes to a user's ICRs. In the ARISES project, by

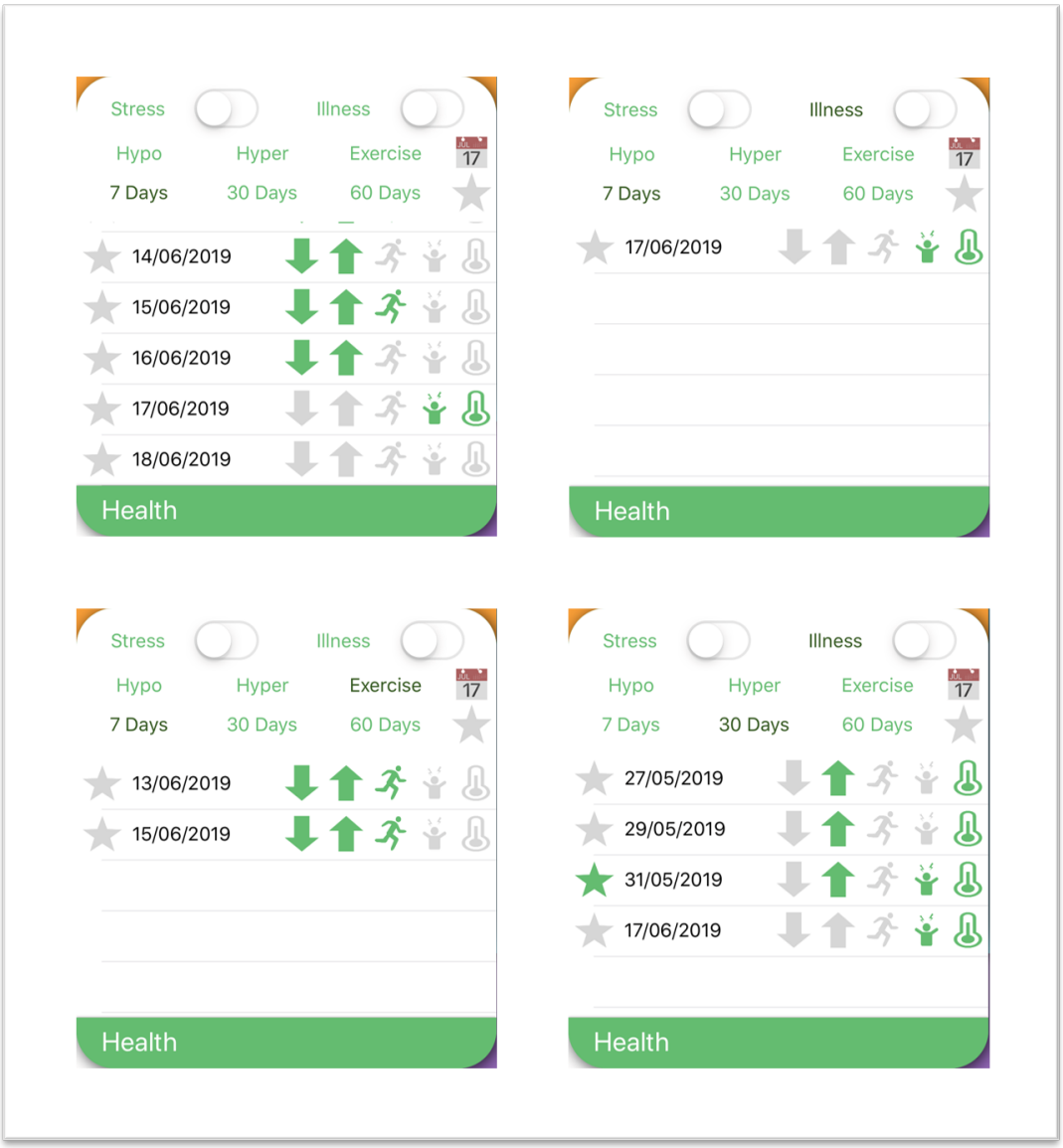


Figure 4.12: Health table with different filters set. Top Left: Last 7 days, Top Right: Last 7 days and Illness, Bottom Left: Last 7 days and Exercise, Bottom Right: Last 30 days and Illness

comparison, this level of visualisation and monitoring is not necessary so a simpler storage system can be used, to reduce the complexity of development.

A very limited method for uploading data is implemented, with the help of Taiyu Zhu, currently only capable of uploading meal data. It uses Amazon Web Services (AWS) S3 as the storage location, and uploads JSON objects. Much work is required to expand this system to be capable of storing all users data reliably throughout a clinical trial. Progress in the ABC4D app converting database objects, uploading data automatically at certain times of day, checking for an internet connection, parsing server responses and handling retries is likely transferable.

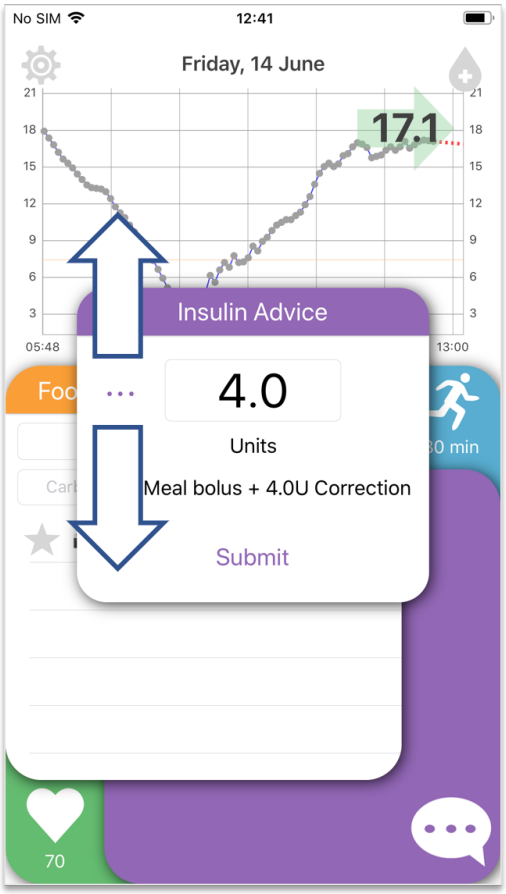


Figure 4.13: System for dynamically adjusting insulin and seeing the BG prediction for that value

4.6 User Experience

4.6.1 Performance

It is good practice to ensure that the UI of an app remains interactive, as a frozen screen is associated with app crashes and bugs, leaving a poor impression on the user. As a medical app that we ask users to put their faith in, this is best avoided. To prevent this, anything requiring a large amount of processing is placed on a separate thread, allowing the UI to run smoothly and without hindrance. Similarly, there are no known crashing bugs in the current app version

4.6.2 User Interface

Significant work has been done to improve the layout of the app on devices with smaller screens. Previously the ARISES app was designed and tested only on larger screen sizes, partially due to the available test device and partially because the density of elements in the design makes it increasingly difficult to layout the app on

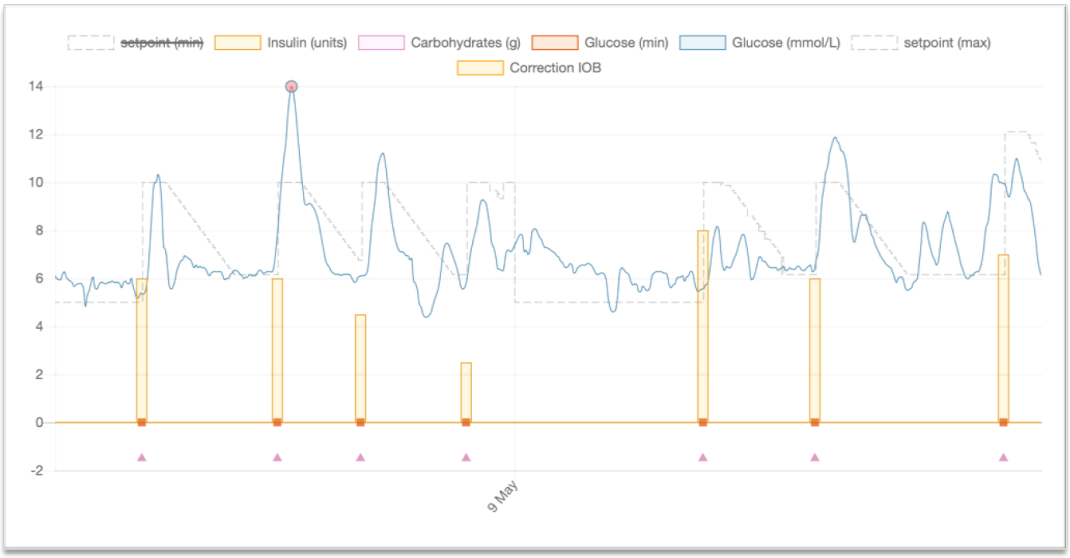


Figure 4.14: Data displayed on the ABC4D web interface, collected during early testing

smaller screens. As you can see from Figure 4.15, the improvement is significant, making the app much more usable and consistent in design.

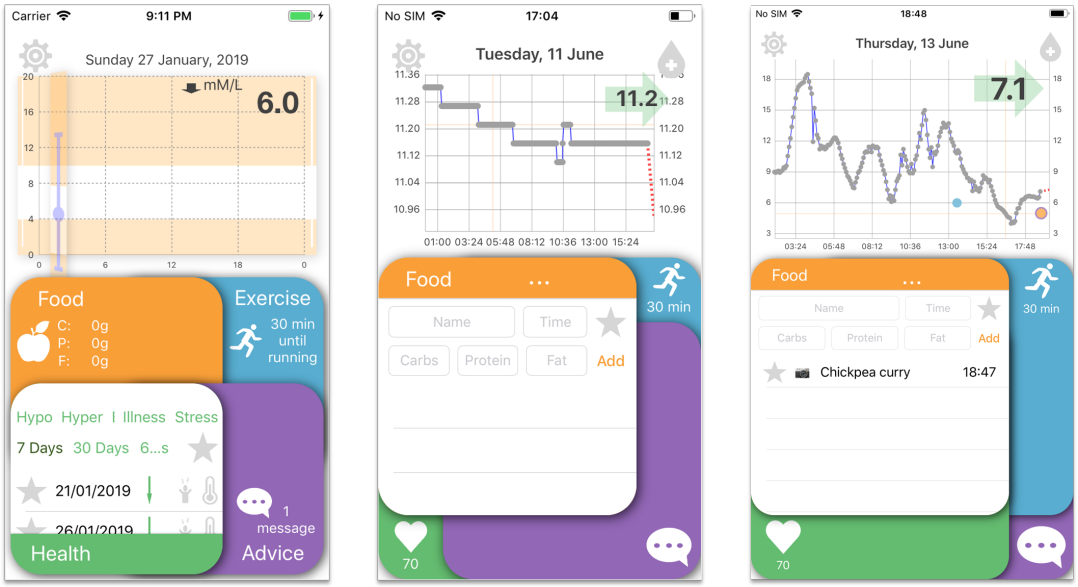


Figure 4.15: Layout update comparison. Left: old layout on the iPhone 5s; Middle: new layout on the iPhone 5s; Right: new layout on an iPhone 7plus

The new layout can be used by all phone sizes now as space is defined in proportion to the size of the screen the app is being displayed on, as shown in Figure 4.16.

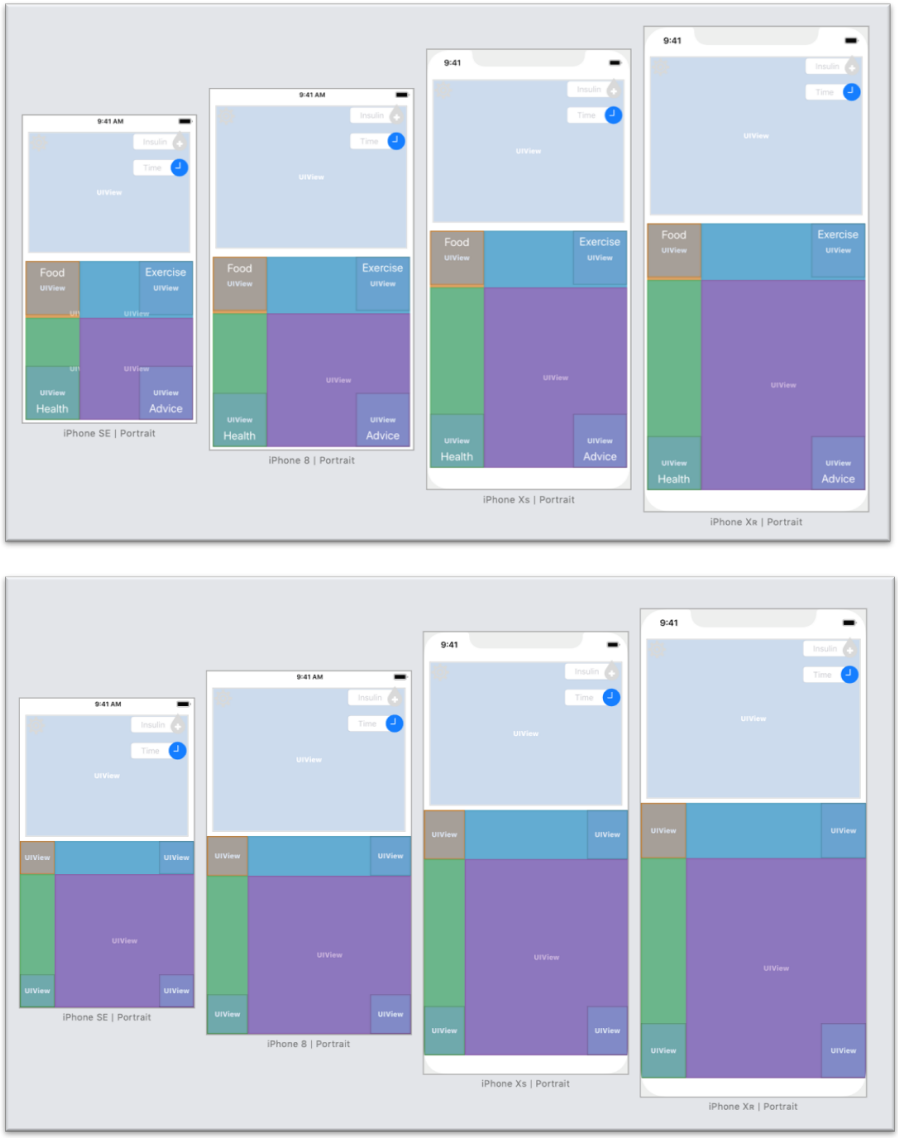


Figure 4.16: Comparing old skeleton layout (Top) with new layout (Bottom) across a range of screen sizes.

Chapter 5

Implementation

In this chapter the specific implementation details of previously described features will be highlighted and further explained. When referring to notifications later in this chapter, often this is describing the notification and observer pattern used to communicate between different parts of the app to update views, unless it is specifically stated as a notification to the user.

5.1 Data Collection

5.1.1 Continuous Glucose Monitor

The CGM implementation uses a framework called CGMBLEKit [24], as no official Dexcom developer tools or frameworks exist currently. Installing this framework requires a bridging header file and enabling the "Uses Bluetooth LE accessories" background mode option in the project settings.

This specific framework allows a number of options, such as between running the app simultaneously with the Dexcom app or without it, which requires extra steps to handle calibrations. We opted for the tandem approach, which allows simplifying the implementation so that all that is required from the user is a transmitter ID. Once entered into the settings page this is stored with a persistence method called user defaults. This allows the user to set this once and forget about it.

After transferring the required files, a number of methods must be implemented to execute the desired behaviour. When launching the app, the transmitter ID is retrieved from the user defaults settings and the units for the BG values received are set to mmol/l. If settings are added to allow a user to change units, it should not change this value, but instead trivially convert the units when displaying values instead. The most important of the required methods are `transmitter didRead glucose` and `transmitter didReadBackfill glucose`. These are called when the app is receives a `Glucose` object from device. For the backfill method, an array of `Glucose` objects is received. These `Glucose` objects contain a range of information, but the

variables we are most interested in are `glucose.readDate`, `glucose.glucose` and `glucose.trend`. These values are stored in the database in a new `GlucoseMO` object. It is important to note that `glucose.trend` does not provide the rate of change of recent BG values, as might be expected, but instead returns integer values corresponding to the position of the trend arrow displayed in the Dexcom app. Other methods that can be implemented handle potential errors that may be received.

The framework handles scanning and reconnecting to known devices, so even if the CGM loses connection it will reconnect seamlessly. The only issue with running the Dexcom app simultaneously is that it causes the rare possibility that if the ARISES app is closed the Dexcom app will collect the backfill data after a reconnecting, causing the backfill to be deleted by the device and preventing it from being uploaded to the ARISES app. If the Dexcom app was not running, the device would have to wait until the ARISES app collects the backfill.

5.1.2 Empatica Wristband

Implementing a connection to the Empatica E4 wristband requires using their iOS SDK [25]. This is because connecting a device requires inputting the API key that the device is connected to on their system, then authenticating that API and device against their server. The specific implementation details of this are hidden within their SDK and cannot be changed. As a result, an internet connection is required to connect a device, which is not ideal.

Current implementation is set up so that when a user opens the settings page, authentication is attempted and if successful a table will appear. The the user needs to set their wristband to discovery mode by pressing and holding down on the button. Then the table should show the device name and can be tapped to connect.

Because we are required to use the Empatica SDK, we are limited to the functionality they expose and cannot do much to fix any odd underlying behaviour. For example, on odd occasions the device presents itself in the table with no ID number, and cannot be connected, but there is nothing that can be done to debug this.

This also prevents any real progress in attempting to set up automatic reconnection. Efforts were made to store the device, notify the user on a disconnect and automatically connect to that device, when a disconnection was received. This still required the user to set the device to discovery, but could be done without opening the phone. When reconnecting however, errors would be raised about received packets being out of sync, and received data would be timestamped incorrectly. On occasion the connection would simply fail. This is not good enough for our purposes.

The device also displays poor connection stability when low on battery. So

a method is implemented to track the battery level and notify a user when it is low, prompting them to charge the device. Notifications are also sent to the user on disconnection of the device, to ask them to reconnect it manually. Sending notifications requires the user's permission, which is sought when they initially open the app and handled in the app delegate.

Once a connection has been made, a number of methods are implemented to handle receiving the streamed Empatica data. Within each of these methods the timestamp must be reformatted to a date from the time interval provided, which is in Unix time, the number of seconds since January 1st 1970. The important values are then repackaged into an appropriate object to be stored in the database. The heart rate method is implemented identically to other methods but appears to receive no data. This is potentially because the calculations done to get this from blood volume pulse and interbeat interval are not done on the device in streaming mode. As those variables are stored however, this can be calculated later.

One method allows checking if the device is on a user's wrist, so that condition is stored and checked in each method, to prevent storing data when the user is not wearing the device.

5.1.3 Diary Entry

To ensure correct data input, specific input methods are set on different text fields. Some examples are shown in Figure 5.1. In the carbohydrate text field, the number of characters is limited to 3 as part of the system for dynamic prediction and some other text fields could benefit from restrictions like this.

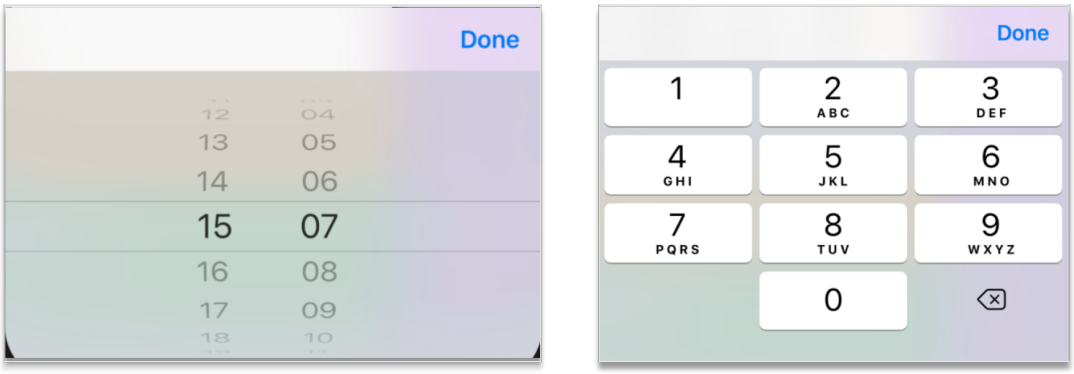


Figure 5.1: Left: Time picker, Right: Keypad without decimal point

To ensure convenient entry of values in each domain, observers are set to call functions to handle the appearance and disappearance of a keyboard or picker. This allows adjusting the view positions to prevent obscuring the text field which is being edited, as shown in Figure 5.2.

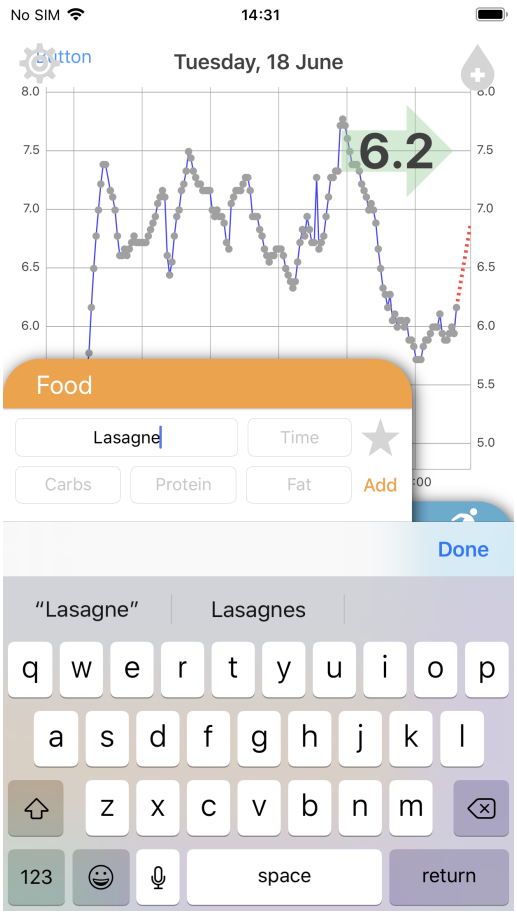


Figure 5.2: Showing moving of a view to prevent keyboard obscuring an input

5.2 Data Storage

The ARISES app contains a database implemented in Core Data, Apple's relational database framework that saves to an SQLite file by default. The database has been expanded to handle storing all the data collection required, which includes lifestyle information, bolus calculator settings and Empatica data. When a device is locked this SQLite file is encrypted and cannot be accessed. Other apps cannot access this database as it is stored in a sandbox, providing a reasonable level of security.

Figure 5.3 shows a graph of the database, describing the objects and values that can be stored. As can be seen from the graph, many objects share a relationship with a day object. This was done in the previous project as a way to fetch information about a specific day. Unfortunately this concept breaks down when considering travelling and the effect of time zones. Day objects are defined with a date corresponding to the beginning of that day, which is different in UTC when travelling. As such, multiple day objects can be created when travelling between time zones, and only those corresponding to the current time zone will be fetched correctly. Ideally

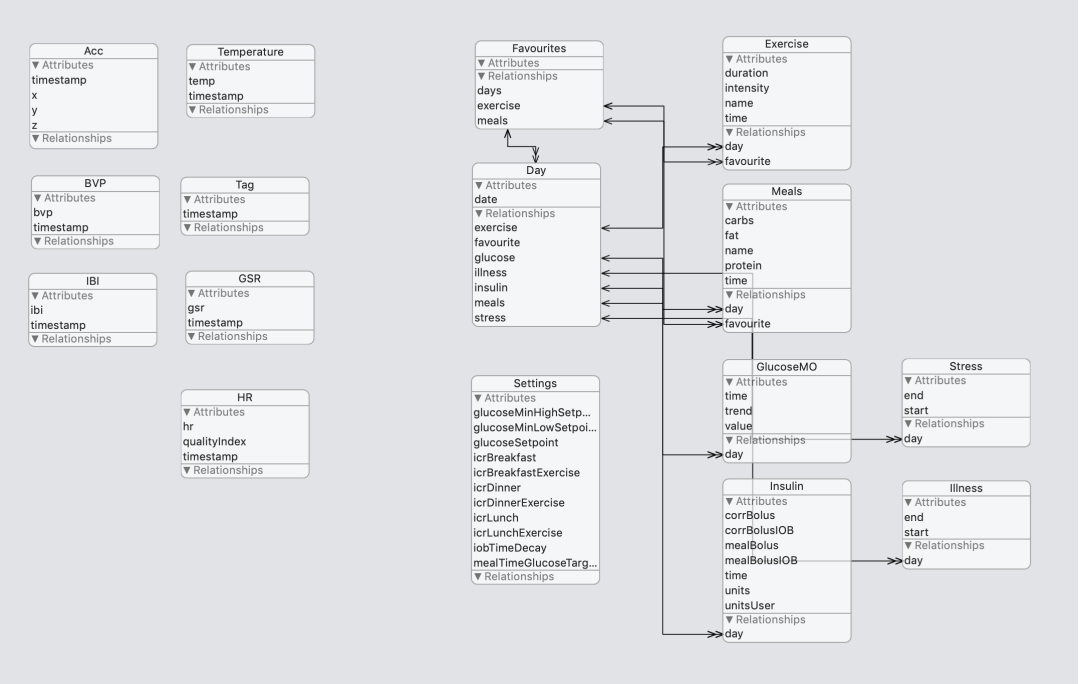


Figure 5.3: ARISES database graph

these relationships should be removed, and the database fetch functions redefined to fetch the relevant 24 hours of data for the current time zone, as has been done in the ABC4D app. This will require a significant adjustment to a number of other parts of the app, such as how the health section data is fetched.

Modifications have been made to the original ARISES database to ensure dates are stored correctly rather than as strings, fixing several previous issues surrounding sorting the results of a fetch request.

5.3 Data Processing

5.3.1 Blood Glucose Prediction

Implementing a machine learning model on a phone requires a few key steps. Firstly, the python implementation of the model needs to be converted into a format that can be ran by a machine learning framework on the device. Different frameworks have been described in the background section of this report. Ideally Apple's CoreML framework would be used because of the optimisations provided by access to the device's GPU. In this circumstance however, the model could not be converted due to limitations of the tensorflow to CoreML converter [26] surrounding the use of transpose layers. Instead, the TensorFlow lite (TFlite) framework [27] is used, which does have the benefit of a simpler implementation for inputs than the MLMultiArray

format required for CoreML systems. For the size of the network we are running, the CPU is more than sufficient for dynamic prediction without any noticeable delay, even on an iPhone 5s.

The TFlite framework is installed using the pods package manager. A class is set up to invoke and prepare the model for inputs. The required input data is of shape `[16, 4]`. This corresponds to the 4 channels of data: BG level, Carbohydrates, Insulin and a time index. The BG data is in mg/dl, carbohydrates is in grams, insulin in units and the time index is a value normalised between 0 and 1 representing the time within a day. This data is sampled at the previous 15 time steps of 5 minutes to create the input.

To collect this data from the continuous time data stored in the database, the previous 85 minutes of values for BG, carbohydrates and insulin are fetched and added to an array at an index corresponding to their nearest 5 minute interval. When BG values are missing, the missing values are produced using linear interpolation.

When the input is fed into the model, the output provides an index for an array of bins from -127 to 128, representing a prediction of the change in BG in mg/dl at a 30-minute PH. This is converted to mmol/l and added to the most recent BG value to get the prediction value. A notification is then sent containing this value, which is received by the graph to plot it appropriately.

5.3.2 Bolus Calculator

A basic outline of the information that is fetched from the database and the functions used to calculate the bolus suggestion is visible in Figure 5.4. The ouputs of the `CalculateBolus` function are: Saturated bolus, the recommended bolus rounded to 0.5 units; Meal Bolus, the component of the recommended bolus that accounts for the meal consumed; Correction bolus, the component of the bolus recommended to account for BG levels above the setpoint; Meal and Correction IOB, the new insulin on board values; and Adjustment Dose ROC, the component of the bolus recommended based on the BG rate of change.

`CalculateInsulinOnBoard` uses the IOB decay time from settings and the previous insulin record, which contains previous IOB values to calculate the current insulin on board for a user, based on the current time. `RetrieveICR` uses the current time of day to determine which ICR value to use for this meal. If an exercise log from within the last 8 hours exists, the exercise ICR value is retrieved instead. `CalculateVariableGlucoseTarget` takes information about the previous meal and a user's BG target information from the settings to calculate a variable glucose target to prevent over-medicating insulin by only correcting to to the postprandial

BG curve. `CalculateGlucoseRateOfChange` fits a first order polynomial with the last 4 BG values, which are received every 5 minutes. If any of these values are missing or the last BG value is not recent, the rate of change is set to 0 instead. `CalculateBolus` calculates the meal bolus based on the selected ICR and carbohydrate intake. Then a correction bolus is calculated based on the current BG level, IOB and targets. Currently this is then rounded to a precision of 0.5 units, as that is the precision of the insulin pen used in the ABC4D project, but this could be removed easily if desired.

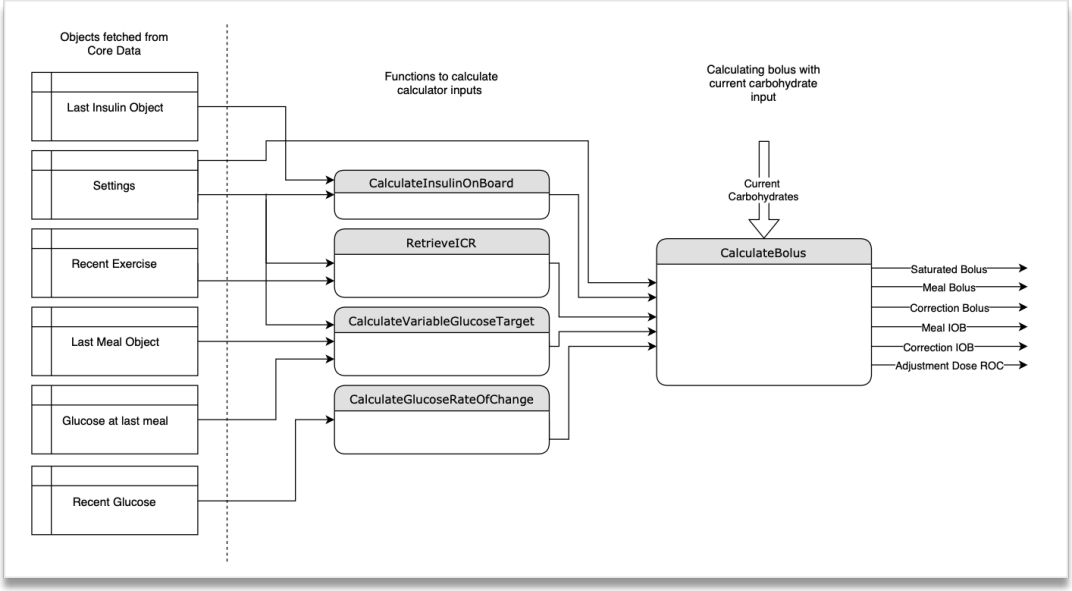


Figure 5.4: Diagram showing objects fetched and functions required to calculate a bolus suggestion

5.4 Data Visualisation

5.4.1 Display

The new graph implementation uses the Charts [28] library. A Line Chart View is placed where the graph is desired. Data that should be plotted is fetched and each object is mapped to a Chart Data Entry, defining the y and x value. It is important that the array of values to be mapped to this data set is provided in ascending order of dates, otherwise this will cause problems. To display time on the x axis, each object's date is converted into a specific format relative to a reference date. An `xValuesNumberFormatter` is defined to provide the format for values on the x axis, in this case "HH:mm" format. Each Chart Data Entry is appended to an array and a Line Chart Data Set is defined using these values. Then it is simple to choose some

plot settings such as line colour, width, radius of circles plotted etc. Features such as zooming into different ranges are provided by default unless otherwise specified.

To update the graph when new values are available a number of observers are set in the graph's view controller class, which are called when new meals, BG values, exercise or predictions are received. This is the system that enables the dynamic updating of predictions, as whatever part of the system calls the prediction, the graph will receive the notification and update the plot. The performance of this is good enough that even when dynamically displaying values the update is not noticeable.

These observers are also used to update the BG value and trend displayed. The trend arrow is animated to adjust to the new rotation position. This is done by calling `getRotation` which translates the relatively unintuitive trend value received from the CGM to the trend arrow rotations visible in the Dexcom app.

5.4.2 Domains

The food domain, when closed, displays a count of the current day's carbohydrate total. This is done using a computed variable on the current day object, which is updated when a new meal is added. The health domain offers the ability to filter through summaries of previous days by returning an array of day objects and filtering this array repeatedly based on certain tags. This is done using a similar computed variable which returns array of tags such as "Hypo" and "Hyper" depending upon the values of objects related to that day. Currently this is done in a view controller, but could likely be done more efficiently using combined fetch predicates. This is not necessary however, as the performance, even when filtering through 60 day objects is fine. In the previous project this was not working correctly due to storing the date of day objects as strings instead of dates, preventing them from being filtered correctly.

Displaying dynamic predictions is done by capturing the input value, appending that to the prediction inputs fetched from the database appropriately, then calling the prediction. In the case of the carbohydrate text field, each time a new character is added or removed, a method is called which converts this string to an integer and calls the prediction. With the dynamic insulin predictions, a gesture recogniser is used to track the position of the finger on the screen, converting that into a value to add to the current insulin value. The top of the screen is defined as +6 units and the bottom as -6 units. This provides a reasonable range and the gesture recogniser calls a function repeatedly, which calls the prediction function with the insulin value appended.

5.5 Data Uploading

In the ABC4D project, data is uploaded via the server's designated APIs, requiring authentication headers with specific user tokens to ensure data is stored and displayed in the correct way. In the ARISES project an AWS S3 bucket is being used, so this level of specificity is not required, however some method for distinguishing between which JSON files belong to which participants is necessary. Some sort of unique ID can easily be appended to any JSON files when converted however.

The AWS iOS SDK [29] is used to simplify implementation. Functions are provided to upload JSON files to a defined bucket. Further testing needs to be done to determine the responses and errors that may be raised using these functions and handle them appropriately.

Converting database objects into JSON objects is simple when leveraging the `Codable` protocol. When using this, simply define a struct in the format required for the JSON and ensure it conforms to the `Codable` protocol. This just requires using more primitive data types such as strings over Dates, so some date formatting may be required. Then all that is required is initialising the struct with the contents of the database object and from there conversion to JSON is a single line of code. `let JSON = try? JSONEncoder().encode(structObject)` This can throw, so should be wrapped in a guard statement for safety.

Once a higher-level function is defined to upload a day of data, a system is required to manage which days have been uploaded. With the existing time zone issues, it's important to fetch all days, not specifically one's with the same start of day, to avoid losing data.

In the ABC4D app the current management system is that each day has a variable stating whether it has been uploaded or not and the number of times this has been tried. Every day at midnight all objcts related to days which have not uploaded are retrieved, converted into JSON and then an upload is attempted. Before attempting to upload, the device's internet connection is checked, to ensure wifi is available and if it isn't then the upload is attempted again in 5 minutes. If the upload for a day fails, then the retries counter is incremented and after 5 retries the day is no longer fetched to be uploaded. This is to prevent newer objects not being uploaded because of time spent attempting to upload older objects. This is all done in the background, while a users phone is locked, by triggering this system when a BG value is entered, causing the Bluetooth to wake the phone.

In the ARISES app, some of these methods are applicable, such as tracking uploaded days. Other methods such as attempting to do this in the background,

while the phone is locked, may be much less feasible. This is primarily due to the large amount of Empatica data. When the device is woken by Bluetooth, there is a limited amount of time available to run code, and it is likely insufficient to upload objects of that size. As a result this will need to be run when the user opens the app, but this could also be automated. When converting Empatica data into the JSON format, certain objects may need to be batched into a number of smaller JSONs to allow uploading some objects at a time. This may require further tracking of uploaded objects but this depends on the capabilities of the AWS SDK, which still need to be explored.

5.6 User Interface

In app development, the position of objects in the device's view are defined by constraints, which are a set of rules about where the objects should be and their size. The new user interface, which displays neatly on all device sizes was implemented by completely redoing all of the constraints. This had to be done to replace the previous constraints, which were based on fixed distances between objects, with constraints based on maintaining sizes proportional to the screen size. Because the system is implemented using a skeleton of embedded views, as shown in Figure 5.5, it isn't too complicated to adjust the overarching layout, without affecting the layout of each embedded view. Each embedded view and indicator has also been redone to ensure space is used appropriately on all phone sizes.

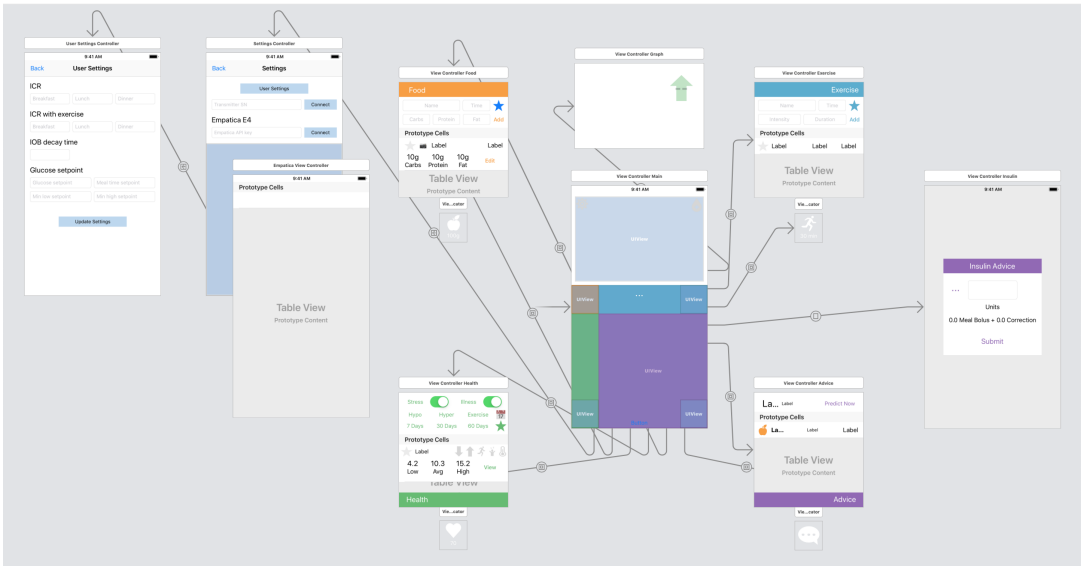


Figure 5.5: Breakdown of views in layout

Another advantage of this method was that it easily enabled the system for

adjusting the graph size. This is done by using a gesture recogniser to track the position of a user's finger on the screen, which updates the constraint controlling the percentage of screen which is occupied by the domains versus the graph. Because all the other constraints are relative and proportional to each other, they can smoothly change. Some limits are placed on the upper and lower end of this adjustment, as reducing either section's size any further becomes unusable.

This is similar to how the horizontal graph feature works. When the user turns the phone sideways, specific observers are triggered, which prepare the device for that orientation. These hide the domain views and adjust the constraints on the graph view to expand to the entire screen. When the user turns the phone back to portrait this process happens in reverse. Specific care needs to be taken which observers are used, to ensure a smooth transition as they each trigger at slightly different times during the rotation.

Chapter 6

Testing

To ensure all parts of the app function correctly, a significant amount of testing is required. As the app contains a variety of components, several different methods are used to test the app.

6.1 Continuous Glucose Monitor

The stability of the Bluetooth connections from the wristband and CGM must be tested to ensure it is as close as possible to the manufacturer's apps. These can be considered our upper bound for performance as the development process of a company is likely much longer and more thorough than this project allows.

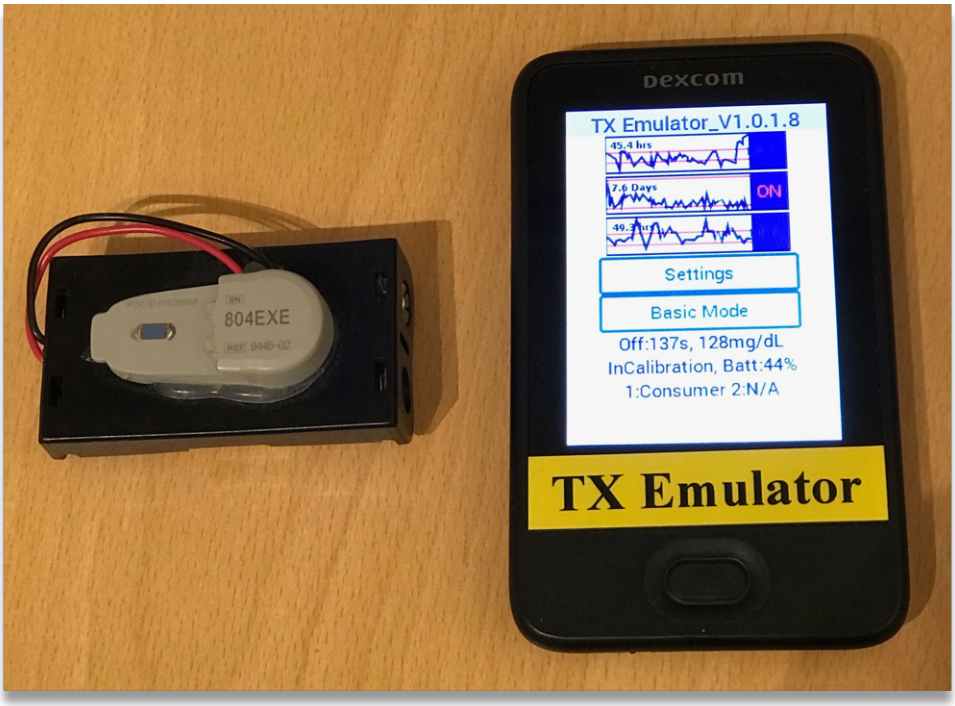


Figure 6.1: CGM emulators. Left: Constant output value, Right: Advanced profile output

For general testing during development, two types of emulators have been used,

shown in Figure 6.1. The emulator on the left produces a constant output of 11.2 mmol/l every 5 minutes. The emulator on the right contains a few extra features, outputting a realistic profile of BG levels. Another feature which has been useful is the ability to emulate signal loss and low battery state, allowing easier testing of reconnecting and handling data backfill.

The backfill system has been tested by using this emulator as well as turning off the phone, waiting to allow the backfill to store some data and then turning the phone back on. This simulates cases where a user's phone runs out of battery. In these cases, tests show that several hours of CGM data can be loaded from the backfill into the ARISES, ABC4D and Dexcom apps simultaneously.

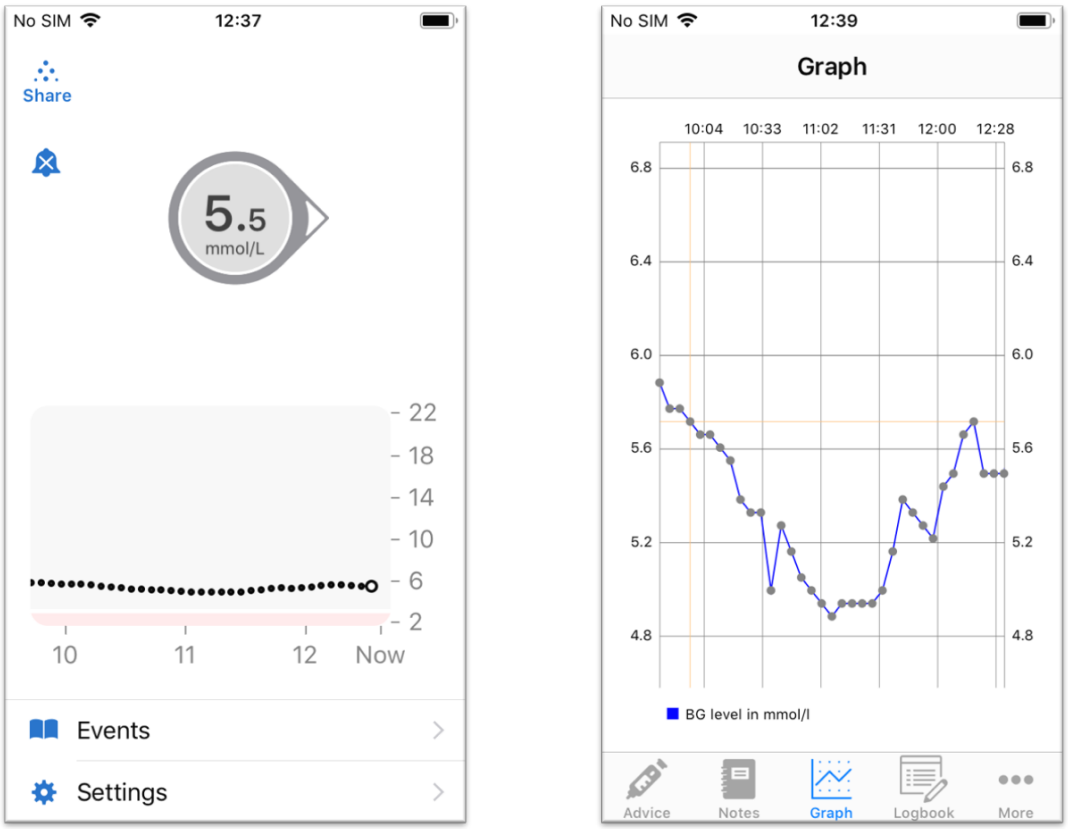


Figure 6.2: Comparison of graphs for CGM data from Dexcom app (Left) and ABC4D app (Right)

Figure 6.2 shows the CGM data collected over the same time period on the Dexcom app and ABC4D app. The CGM implementation in the ABC4D and ARISES app is identical. While it is not easy to see how similar this data is due to the lack of grid lines and size of the Dexcom graph, the data is clearly over the same range of 5-6 mmol/l and both end at values of 5.5 mmol/l. This comparison has been done for several months and is always consistent. The Dexcom app can also be set to display the previous hour of CGM data, allowing the user to scroll with their finger

and view each BG value, and when doing this it is clear that the values received are the same.

In addition, the ABC4D app was tested by two clinicians who wore a real Dexcom G6 CGM over the course of a week, engaging in a variety of typical activities.

As such the CGM connection has been quite rigourously tested. The same tests were done to verify the trend arrow is identical to the Dexcom app. During testing by the clinicians a specific case where an incorrect trend position was being displayed was found and rectified. Following this, all trend arrow positions were tested by temporarily adding a button to the UI to allow cycling through and verifying arrow positions are as expected.

6.2 Empatica Wristband

As the Empatica app cannot be run simultaneously, identical data cannot be compared. The data stored by the device in recording mode, which is sent to the Empatica Connect web service, shown in Figure 6.3, was compared to ensure received values are in the expected ranges and vary in a similar way.

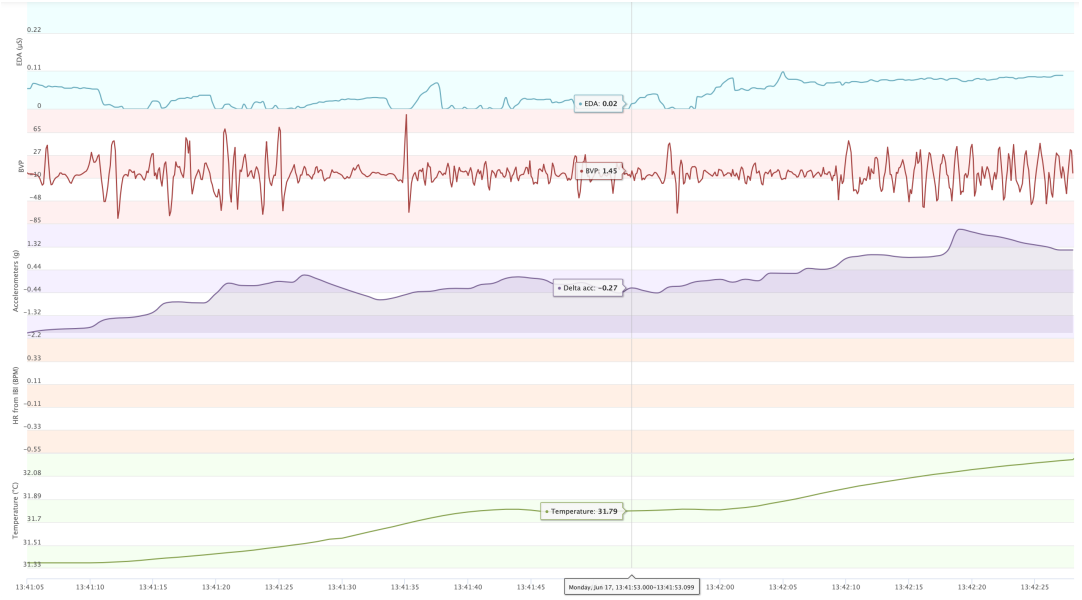


Figure 6.3: Empatica data values stored by the E4 realtime app

For comparison, Figure 6.4 shows 10 seconds of EDA data collected by the app, Figure 6.5 shows 10 seconds of temperature data and Figure 6.6 shows 10 seconds of BVP data. The x-axis shows the number of evenly spaced samples plotted. This provides an idea of how frequently the data is received. As can be seen in the graphs, EDA and temperature data appear to have the correct ranges and expected variations over time, however the BVP data shows much larger values than expected,

but similar response over time. This implies a unit conversion taking place before the values are displayed. This should be considered and explored if BVP is used to calculate user's heart rate. No heart rate values have been received from the device despite implementing the method identically to other received values.

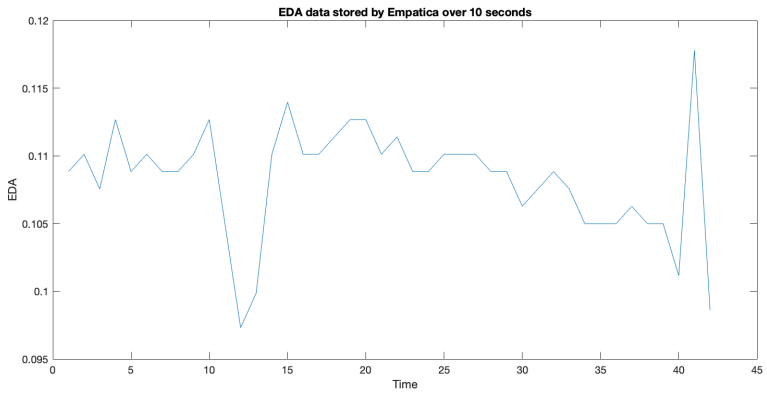


Figure 6.4: EDA data stored by the Empatica over 10 seconds

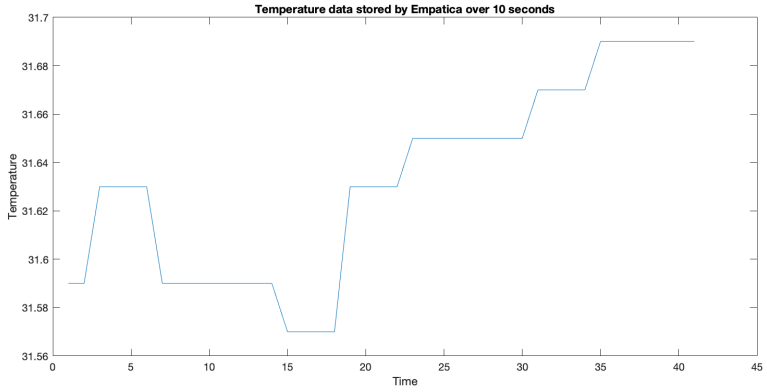


Figure 6.5: Temperature data stored by the Empatica over 10 seconds

Unfortunately the stability of the Empatica connection is not as reliable. The device disconnects at a distance of roughly 6 meters, or when the battery is low. Connection of the device lasts several hours typically before disconnecting for a currently indiscernable reason. Notifications set to trigger when the devices disconnects function correctly, but are easily missed in practice.

6.3 Data Storage

The database has been tested to ensure that input data is stored as expected. This is done by downloading the app's data container and opening the SQlite file for the database. Within it, all stored values are visible and can be compared to their inputs

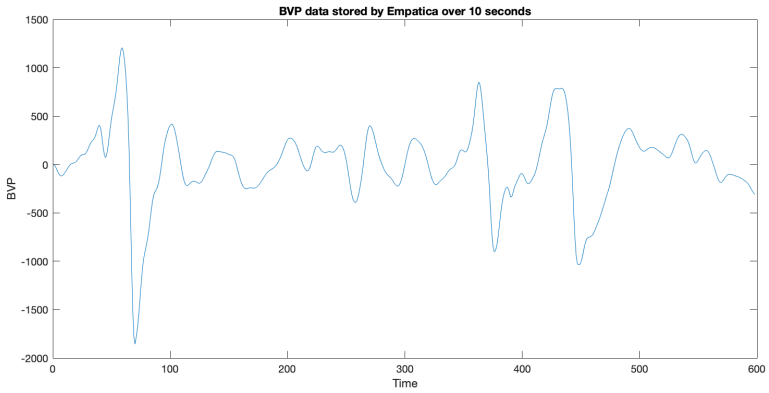
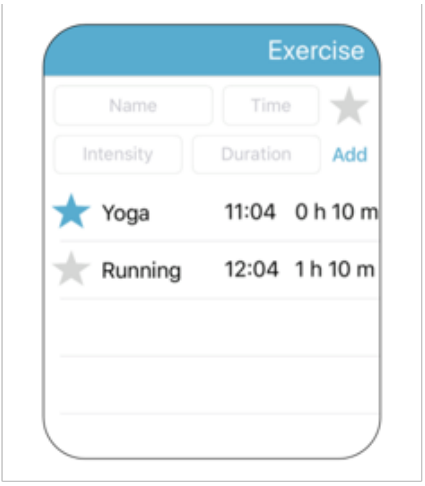


Figure 6.6: BVP data stored by the Empatica over 10 seconds

and expected values. Figure 6.7 shows two exercise logs that have been added to the diary, and their corresponding database entries.



ZFAVOURITE	ZDURATION	ZINTENSITY	ZNAME	ZTIME
Filter	Filter	Filter	Filter	Filter
1	0 h 10 m	Medium	Yoga	11:04
NULL	1 h 10 m	Low	Running	12:04

Figure 6.7: An example exercise log (Top) and the corresponding database entry (Bottom)

Prior to integrating the Empatica, the size of the database when storing roughly 2000 glucose objects from the CGM emulator was tested and took only 172 KB of memory. This corresponds to nearly a week of data, assuming consistent connection and that the CGM transmits data every 5 minutes. As such, storing all of a users CGM data on the phone during the length of a typical clinical trial is perfectly feasible.

In comparison, storing Empatica data for this duration is not at all feasible. Table 6.1 shows the large number of objects stored over a roughly 10 minute period of Empatica use. After 4 hours and 15 minutes, the database size had risen to 18.7 MB when accelerometer data was also stored. As can be seen, accelerometer data takes up a significant percentage of the total memory required.

Acc	0	19,458
BVP	36,115	38,918
EDA	2,256	2,430
Temp	2,256	2,432
Tag	1	1
IBI	80	85
Memory (MB)	1.2	1.8

Table 6.1: Table showing the number of objects stored and the size of the database after a 10 minute period, when integrating the empatica without (Left) and with (Right) accelerometer data stored

6.4 Machine Learning Model

When placed on the app, the behaviour of the model is identical to the TensorFlow implementation, other than the amount of time it takes to run. This was verified by exporting the predictions made on the test set of the Ohio dataset by the TensorFlow implementation, by the converted TFlite version and then by the model on the device. Every prediction provided is identical and a test has been provided in the project comparing these values. On an iPhone 5s, completing the test suite of 2589 predictions takes 19.19 seconds, whereas on an iPhone 7 plus the tests take 6.31 seconds to complete. As expected from these results, when dynamically calling the model to provide and plot predictions, there is no perceivable delay.

6.5 Bolus Calculator

As a primary part of the ABC4D app, the bolus calculator system has been thoroughly tested. When initially converted into swift from Dr. Pau Herrero's MATLAB code, a test program was provided using data from the UVA/Padova simulator. Figure 6.8, shows the outputs of that program in MATLAB and the corresponding outputs of the Swift version. The actual output tables were also compared to ensure the correct conversion.

Since then some changes have been made to the functioning of specific functions, such as `calculateVariableGlucoseTarget`, `calculateGlucoseRateOfChange` and

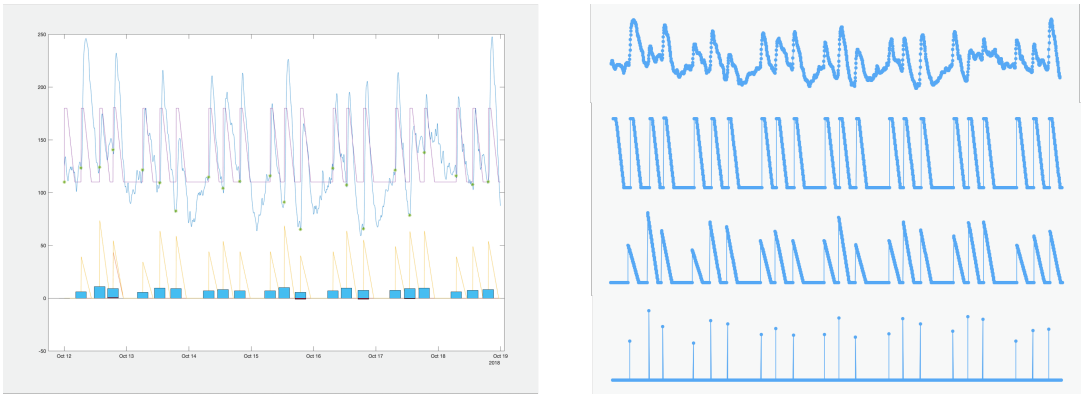


Figure 6.8: Bolus calculator testing results. Left: MATLAB output, Right: Swift implementation output

`retrieveICR`. As a result, specific tests for these functions were defined in the ABC4D app, comparing test data from the new MATLAB implementations, or expected values.

This was also tested during the trial by the clinicians, ensuring any issues which are clinically noticeable, but obscure to a less informed developer, are spotted and fixed. This testing is ongoing and any further fixes can be transferred to the ARISES app due to near-identical implementation.

Chapter 7

Results

The results of this development will not truly be seen until it is placed into the hands of users. During a later focus group, users were provided the app on an iPhone 5s to gauge their response to the current user interface. While at that point, features such as prediction and bolus calculation had not been implemented, the rest of the app was visibly similar to its current state. The response was positive. Previously during that same focus group concern had been raised when viewing concept designs that the graphs seemed busy and that it was important to use the full space as smaller objects are difficult for people with poorer eyesight to see. To later receive positive feedback on how usable the real version of the app is, even on the smallest screen size available, is useful validation for the new UI changes.

How well the app works for users in other areas of development is harder to quantify, but it has aimed to implement and improve the most important features highlighted in focus groups. An evaluation of the success in achieving these features is provided in the conclusion of this report.

Evaluating the app against requirements captured in focus groups and the provided specification, the ARISES app compares favourably.

When evaluating its ability to gain the trust of a user, the strong stability of the CGM connection, a vital part of the system, is a good result. Other systems such as bolus calculators breakdowns and useful summary tables to display stored information also provide some transparency to the app. The poor stability of the Empatica connection, however is a significant blow against this aim, but there is little that can be done to improve this.

Strong progress has been made improve how easy the system is to use, maximising the space available and adding features such as the adjustable and horizontal graph. Even on small screens the system is significantly more convenient to use. Further progress is required to allows users to edit stored data retroactively, as incorrectly entering information that cannot be changed is likely to become an annoyance.

Powerful new features, not available elsewhere have been added, such as the

prediction system and dynamic exploration of possible decisions. These features were highly desired during focus groups and their implementation brings significant value to the app.

The app also retains the core design philosophies specified, ensuring that all information is available on the main view. The most important information is visible at all times. While some navigation aspects have been added to create space for user settings, this system has been designed to allow a user to set and forget these settings. This means that in typical use there is no navigational hierarchy, as desired.

Before use in a clinical trial, there is remaining work that needs to be completed. The system for uploading user data is still primitive. Once this has been completed a system is required to erase stored Empatica data as it is unfeasible to store on the device. The issues which arise during travelling to different time zones should also be rectified, to prevent data loss when participants travel, potentially quite likely if a trial is ran over the summer. This specific issue could not easily be avoided due to inheriting the technical debt from the previous project, and devoting time to fix this specific case within this project would have been done at the cost of other important features.

Chapter 8

Conclusion

Evaluating the app against requirements captured in focus groups and the provided specification, the ARISES app compares favourably.

When evaluating its ability to gain the trust of a user, the strong stability of the CGM connection, a vital part of the system, is very beneficial. Other systems such as bolus calculators breakdowns and useful summary tables to display stored information also provide some transparency to the app. The poor stability of the Empatica connection, however is a significant blow against this aim, but there is little that can be done to improve this.

Strong progress has been made to improve how easy the system is to use, maximising the space available and adding features such as the adjustable and horizontal graph. Even on small screens the system is significantly more convenient to use. Further progress is required to allows users to edit stored data retroactively, as incorrectly entering information that cannot be changed is likely to become an annoyance.

Powerful new features, not available elsewhere have been added, such as the prediction system and dynamic exploration of possible decisions. These features were highly desired during focus groups and their implementation brings significant value to the app.

The app also retains the core design philosophies specified, ensuring that all information is accessible from the main view. and the most important information is visible at all times. While some navigation aspects have been added to create space for user settings, this system has been designed to allow a user to set and forget these settings. This means that in typical use there is no navigational hierarchy, as desired.

Before use in a clinical trial, there is remaining work that needs to be completed. The system for uploading user data is still primitive. Once this has been completed a system is required to erase stored Empatica data as it is unfeasible to store it all on the device. The issues which arise during travelling to different time zones should also be rectified, to prevent data loss when participants travel, potentially

quite likely if a trial is ran over the summer. This specific issue could not easily be avoided due to inheriting the technical debt from the previous project, and devoting time to fix this specific case within this project would have been done at the cost of other important features.

In conclusion, compared to other apps on the market, this app provides a number of superior features, such as the state-of-the-art prediction algorithm, but is not as polished as production apps. As a tool for use in clinical trials, the app is custom-tailored to the requirements, allowing collection of the specifically desired data and testing of promising algorithms, models and user interface. In this regard, some further work is however required to allow easier collating of this data from participants.

Chapter 9

Further Work

The remaining work which is most vital to complete is the networking code to allow safe and reliable data uploading of participants data. This is not especially complicated but it takes time to implement the correct JSON formats for each object as there is a lot of boilerplate code required. Considering all potential error cases for the upload system and handling them appropriately will also take time to implement and test.

Following this, the next priority is handling the time zone issue. This pervades a range of the app's features, so ensuring that all existing functionality remains will require good testing, and adjustments in a number of places. This is not an insignificant task, but should be done to ensure all the data is collected appropriately during a clinical trial.

A lesser, but significant topic of future work is adjusting the diary entry systems to allow editing of previously entered values. Creating the new functions to enable editing these objects is not complicated, but new popover views will likely need to be created to allow the user to edit values. Careful consideration needs to be taken to ensure these changes appropriately affect the following stored data. For example, editing an insulin log stored early in the day may cause the IOB already stored in later insulin logs to be incorrect. Checking all of the cases where these retroactive changes affect the current state of the system could require significant testing. A compromise could be made by allowing editing within a short time window of inputting information, to allow users to correct recent mistakes, without having to consider further effects of changes.

It is also desirable that the Empatica connection is improved, but effort was made to accomplish this to no avail. The required SDK and device connection process significantly limits the progress that can be achieved here. As the current implementation has similar performance to the e4 realtime app, developed by Empatica, it doesn't appear that much more can be done.

Beyond this, a number of small improvements can be made with less investment of development time:

- Exercise duration should be stored as a numeric value of minutes instead of as a string.
- Stress and illness are currently stored as logs of when a user turns the switch on and off. With more clarity on the format desired, this could be improved.
- The food tab has the only working indicator, providing summary information when the tab is closed. Other indicators should be implemented.
- IOB could be displayed on the display next to BG levels, and in an expanded cell of the insulin table.
- The bifocal display sidebands could be reintroduced.
- Clicking on dots representing meals etc. on the graph could display information about that meal.

Chapter 10

User Guide

The project can be downloaded from the GitHub page available in section A.1. This can be installed on any device which is linked to an Apple developer account with membership in the Apple Developer Program, or that is part of the iOS Developer University Program. This installation requires a Mac with the newest version of Xcode installed.

The app can be remotely distributed to devices. The devices must be included in a provisioning profile created on the Apple develop website. You can then send them a link that installs the app when clicked from the device. This requires archiving the app, then following the development distribution method. The received files should then be uploaded to a server to allow them to be accessed remotely and then a link can be distributed. The link will be of the form:

```
itms-services://?action=download-manifest&url=https://www.
    websiteadress/ARISES.plist
```

where `websiteaddress` is replaced with your hosted location. This has been used frequently in the ABC4D project to provide updates during testing.

Once the app is installed, the set up process is simple. Enter the transmitter ID in the app's settings page, after connecting the Dexcom G6 to the Dexcom app. Add and save the required user settings. Then press and hold the Empatica to set it to discovery mode, before clicking on the device which appear in the table on the settings page. This process connects all the devices and provides all the information required for the app to function. For the app to receive data from devices, it must remain open in the background.

Appendix A

Project Files

A.1 Project GitHub page

https://github.com/RyanArmiger/ARISES

Bibliography

[1] G. REPORT and O. DIABETES, "Global report on diabetes."

[2] S. I. Sherwani, H. A. Khan, A. Ekhzaimy, A. Masood, and M. K. Sakharkar, "Significance of hba1c test in diagnosis and prognosis of diabetic patients," *Biomarker Insights*, vol. 2016, no. 11, pp. 95–104, Jul 3, 2016. [Online]. Available: http://insights.sagepub.com/significance-of-hba1c-test-in-diagnosis-and-prognosis-of-diabetic-pati-article-a5741

[3] P. J. M. N. P. C. Herrero, Pau|Pesl, "Method for automatic adjustment of an insulin bolus calculator: In silico robustness evaluation under intra-day variability," *Computer Methods and Programs in Biomedicine*, vol. 119, no. 1, pp. 1–8, 2015. [Online]. Available: https://www.clinicalkey.es/playcontent/1-s2.0-S0169260715000279

[4] D. M. Nathan, "The diabetes control and complications trial/epidemiology of diabetes interventions and complications study at 30 years: overview," pp. 9–16, 2014. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/24356592

[5] S. Oviedo, J. Vehí, R. Calm, and J. Armengol, "A review of personalized blood glucose prediction strategies for t1dm patients," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 33, no. 6, p. n/a, Jun 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.2833

[6] M. Rigla, G. García-Sáez, B. Pons, and M. E. Hernando, "Artificial intelligence methodologies and their application to diabetes," *Journal of Diabetes Science and Technology*, vol. 12, no. 2, pp. 303–310, Mar 2018. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/1932296817710475

[7] J. Chen, K. Li, P. Herrero, T. Zhu, and P. Georgiou, "Dilated recurrent neural network for short-time prediction of glucose concentration," in *27th International Joint Conference on Artificial In-telligence and the 23rd European Conference on Artificial Intelli-gence (IJCAI-ECAI 2018)*, 2018.

[8] Abstract, "School of electrical engineering and computer science ohio university athens, ohio, 45701, usa fmarling,bunescug@ohio.edu."

[9] T. Zhu, K. Li, P. Herrero, J. Chen, and P. Georgiou, "A deep learning algorithm for personalised blood glucose," in *Blood Glucose Level PredictionChallenge,the 27th International Joint Conference on Artificial In-telligence and the 23rd European Conference on Artificial Intelli-gence (IJCAI-ECAI 2018),International Workshop on KnowledgeDiscovery in Healthcare Data.*, 2018.

[10] C. Pérez-Gandía, A. Facchinetti, G. Sparacino, C. Cobelli, E. J. Gómez, M. Rigla, A. de Leiva, and M. E. Hernando, "Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring," *Diabetes technology & therapeutics*, vol. 12, no. 1, pp. 81–88, Jan 2010. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/20082589

[11] T. Zhu, "A smartphone-based platform integrating ai for adaptive glucose prediction," Ph.D. dissertation.

[12] Bremer, Gough, . Cryer, and 2003, "Automatic blood glucose prediction with confidence using recurrent neural networks."

[13] K. Li, C. Liu, T. Zhu, P. Herrero, and P. Georgiou, "Glunet: A deep learning framework for accurate blood glucose forecasting."

[14] G. Scheiner, "Cgm retrospective data analysis," *Diabetes Technology & Therapeutics*, vol. 18, no. S2, p. 22, Feb 1, 2016. [Online]. Available: https://www.liebertpub.com/doi/abs/10.1089/dia.2015.0281

[15] D. Ravi, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep learning for health informatics," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 4–21, Jan 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7801947

[16] G. Sparacino, F. Zanderigo, S. Corazza, A. Maran, A. Facchinetti, and C. Cobelli, "Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 5, pp. 931–937, May 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4155016

[17] C. Zecchin, A. Facchinetti, G. Sparacino, G. D. Nicolao, and C. Cobelli, "Neural network incorporating meal information improves accuracy of short-time prediction of glucose concentration," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1550–1560, Jun 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6157604

[18] A. D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," Sep 12, 2016. [Online]. Available: https://www.openaire.eu

[19] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *CoRR*, vol. abs/1606.05328, 2016. [Online]. Available: http://arxiv.org/abs/1606.05328

[20] C. D. Man, F. Micheletto, D. Lv, M. Breton, B. Kovatchev, and C. Cobelli, "The uva/padova type 1 diabetes simulator," *Journal of Diabetes Science*

and Technology, vol. 8, no. 1, pp. 26–34, Jan 2014. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/1932296813514502

[21] M. Reddy, P. Pesl, M. Xenou, C. Toumazou, D. Johnston, P. Georgiou, P. Herrero, and N. Oliver, "Clinical safety and feasibility of the advanced bolus calculator for type 1 diabetes based on case-based reasoning: A 6-week nonrandomized single-arm pilot study," *Diabetes Technology & Therapeutics*, vol. 18, no. 8, pp. 487–493, Aug 1, 2016. [Online]. Available: https://www.liebertpub.com/doi/abs/10.1089/dia.2015.0413

[22] "Dexcom g6." [Online]. Available: https://www.dexcom.com/g6-cgm-system

[23] S. Schmidt, K. Nørgaard, and DMSc, "Bolus calculators 2014, vol. 8(5) 1035–1041," 2014.

[24] "Cgmblekit github page." [Online]. Available: https://github.com/LoopKit/CGMBLEKit

[25] "Empatica ios sdk api reference." [Online]. Available: https://developer.empatica.com/ios-sdk-api-reference-100.html

[26] "Tensorflow to coreml converter github." [Online]. Available: https://github.com/tf-coreml/tf-coreml

[27] "Tensorflow lite for ios." [Online]. Available: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/swift

[28] "Charts ios github page." [Online]. Available: https://github.com/danielgindi/Charts

[29] "Aws ios sdk github." [Online]. Available: https://github.com/aws-amplify/aws-sdk-ios