SOFTWARE REQUIREMENTS SPECIFICATION

for

Glucometer App

Version 0.9.0

Prepared by : Wong Chu Feng

Trilogy Technologies Pte Ltd

21 July 2023

Revision History

Name	Date	Description	Version
Glucometer connectivity integration	10 May 2023	The app currently includes a splash screen and establishes BLE connection to the glucometer, enabling data exchange and monitoring between the app and the glucometer.	0.5
Add Personal Information page and bug fix	22 May 2023	Added a page for users to key in their personal information which includes their name, age, gender, height, and weight. Fixed a bug that prevented the saving of entered personal information.	0.5.1
Export and Storage options	23 May 2023	Users can now choose to export and save biodata as .xls or .csv into their mobile phone storage. Fixed an issue where the exported file could not be saved.	0.6
Add email function	23 May 2023	Users can choose to send the saved data via email	0.6.1
Basic graph display	25 May 2023	Added the most basic graph that reads data from the local database and plots a graph to display all the data.	0.7
Graph package update	26 March 2023	Updated graph package from charts_flutter to syncfusion_flutter_charts due to deprecation.	0.8
System compatibility	1 June 2023	Migrated app from Android 8.0 to Android 10.0, including necessary dependency updates.	0.8.0.1
Graph display and data format fixes	8 June 2023	Fixed DateTime format to ensure proper graph display. Resolved a problem with time format when inserting data into the database.	0.8.1
iOS Development	12 June 2023	Built iOS development app using Xcode 13.	0.8.2
Dashboard enhancement	15 June 2023	Created a new, visually appealing dashboard to display biodata	0.8.3
BLE compatibility improvement	22 June 2023	Fixed problem where newer devices cannot deteect BLE device.	0.8.4
Data filtering and graph visualisation	28 June 2023	Users are now able to filter the data and view their data on the graph by the day, week, month, or year.	0.8.5

Refactor code	29 June 2023	This version change focuses on code refactoring, primarily to improve readability of the code base. This reorganization streamlines the development process, increase code efficiency and ensures a more robust foundation for future enhancements to any component in the app.	0.8.5.1
Graph colour highlighting	30 June 2023	Graph now visually highlights glucose levels above the threshold.	0.8.5.2
Multi-language support	11 July 2023	Added multi-language support for English, Mandarin, Malay, Hindi and Tamil.	0.9.0

Contents

1 Introduction

1.1 Purpose

The Glucometer App serves as a comprehensive digital companion, enabling users to record and track essential health metrics such as heart rate, cholesterol level, uric acid level, glucose level and SpO2 level.
With user-friendly features like data storage and data sharing with healthcare professionals and visual trend analysis through graphs, this app paves the way to revolutionise and elevate the current state of teleconsultation and remote health monitoring between patients and doctors.
By providing a convenient and efficient platform for tracking and sharing essential health metrics, it bridges the gap between individuals and healthcare providers, enabling more informed and personalised remote consultations.

1.2 Document Conventions

- Glucometer
  - A blood glucose meter is a (medical) device used to measure the concentration of glucose in a person's blood. It is commonly used by diabetic patients to monitor their glucose levels.
  - Specific to Trilogy Technologies' glucometer, it can measure not only glucose levels but also uric acid, heartrate, SpO2, and cholesterol levels.
  - The Arduino code defines the healthy range to be $< 140 mg/dl$ or $< 7.8 mmol/l$.
- SpO2
  - The measurement of oxygen saturation in a person's blood. It generally should range from 96% - 99%.
- Cholesterol
  - The fatty substance found in our bodies which can cause health issues in excess.
  - The Arduino code defines the healthy range to be $< 200 mg/dl$ or $< 6.6 mmol/l$
- Uric Acid (UA)
  - A waste product produced by the body's metabolism of certain foods. Elevated UA levels can lead to health issues.
  - The Arduino code defines a healthy range for both men and women.
  - Range for men: $[4.0, 8.5]$ in mg/dl or $[0.24, 0.51]$ in $mmol/l$
  - Range for women: $[2.7, 7.3]$ in mg/dl or $[0.16, 0.43]$ in $mmol/l$
- MAX30100 PulseOximeter
  - Heart rate sensor module that measures heart rate.

- Adafruit SSD1306
  - Organic light-emitting diode (OLED) graphic display.
- Bluetooth Low Energy (BLE)
  - A wireless communication technology designed for energy-efficient data exchange between devices. More commonly used in devices that transmits data over short distances.
  - Unlike traditional Bluetooth, BLE is more optimised for low-power, less data-intensive applications where energy efficiency is crucial, making it ideal for devices that require intermittent data exchange or have limited power.
  - No direct connection is required resulting in significant battery power savings. Instead, the app listens for the broadcast from the glucometer and hones onto the signal emitted from the BLE device to receive the data.
  - Note: BLE connection does not work on an emulator.

This SRS document is adapted from the IEEE standard for Software Requirements Specification.

1.3 Intended Audience and Reading Suggestions

This SRS document serves as a blueprint for developers and testers.
Developers can understand the purpose of this project and find any necessary information about the requirements or functionality of the app through this document. A clear description of the development process will be described and made easy for any to follow along.
Extending from this understanding to testers, they are able to identify the test cases that need to be executed to ensure the app can carry out its desired function. Test scenarios and test cases are to help validate the functionality and performance of the app.

1.4 Project Scope

Glucometer App is a user-friendly free-of-charge application that targets diabetic patients or those at risk and encompasses a range of features designed to empower users in monitoring and managing their glucose levels and aiding their healthcare providers.
This app serves as a digital health companion, enabling users to effortlessly view their glucose levels. With a user-friendly interface, users can securely store their data or conveniently email it to their healthcare provider for analysis and consultation, all done within the app. The app also incorporates an interactive graph, allowing users to visually track and analyse their glucose level over time, fostering a better understanding of their health status and facilitating informed decision-making regarding lifestyle choices or treatment plans.
However, this app will not provide medical advice, nor will it replace the guidance of a healthcare professional.

2 Overall Description

2.1 Product Perspective

The Glucometer app fits into a broader healthcare ecosystem, playing a pivotal role in empowering individuals to manage their health effectively. Within this perspective, the app interacts with various entities, including Trilogy Technologies' glucometers for data recording and input, user devices such as smartphones for data storage and communication channels for data sharing with healthcare professionals.
Furthermore, it aligns with prevailing privacy regulations and data protection measures to safeguard sensitive health information. By considering the app's place within a larger ecosystem, it ensures a seamless user-centric approach to health management.

2.2 Product Functions

- App must record and monitor heart rate, glucose level, cholesterol level, uric acid level and SpO2 levels.
- Users must be able to save their personal information including but not limited to name, age, gender, height and weight.
- App must provide the function to export and save biodata as .xls or .csv files to mobile phone storage.
- App must provide the function to email biodata saved as .xls and .csv files.
- App must allow users to filter and view their recorded data on a graph by day, week, month, or year.
- The graph must be visually appealing and provide color coding to highlight glucose levels above the set threshold.
- App must offer seamless connectivity with compatible glucometers for data input.
- App must be compatible on both Android and iOS platforms.
- User experience must be enhanced with visually appealing and intuitive dashboard to view their biodata while recording with the glucometer.

2.3 User Classes and Characteristics

- Regular users
  - Frequency of use: Regular and frequent use of the app
  - Subset of product functions: Utilise all product functions for comprehensive health monitoring and data management.
  - Technical expertise: Varying technical proficiency levels.

  - Security level: Standard user privileges with access to only their own personal data.
  - Education level/Experience: Diverse educational backgrounds.

- Healthcare professionals
  - Frequency of use: Periodic use for reviewing patient data and providing medical advice.
  - Subset of product functions: Access and analyse user-submitted data for medical evaluation and consultation.
  - Technical expertise: High level of proficiency in digital health tools and analysing health-related data.
  - Security level: Enhanced access privileges to view and analyse multiple users' data.
  - Education level/Experience: Qualified healthcare professionals with relevant medical training and expertise.

- Technical support team
  - Frequency of use: Occasional use for troubleshooting and providing technical assistance.
  - Subset of product functions: Focus on addressing technical issues, software updates and user support.
  - Technical expertise: Advanced technical knowledge to troubleshoot app-related problems.
  - Security level: Access to system logs and user support resources.
  - Education level/Experience: Technical professionals with expertise in app development and support.

2.4 Operating Environment

The app will support both Android and iOS platforms.

2.5 User Documentation

A demonstration video and app screenshots alongside this SRS documentation will be uploaded so prospective users are able to understand how to use the app.

2.6 Assumptions and Dependencies

- Users must be connected to Wi-Fi or mobile data and have a device that supports Bluetooth connectivity.
- Users must use a mobile device that is either on Android or iOS platforms to download the app.
- The app will support English, Mandarin, Malay, Hindi and Tamil languages.
- The app is only designed for mobile phone usage.

3 External Interface Requirements

3.1 User Interfaces

The app features a personal information page that includes fields that accepts user input where the system only accepts a valid value. For example, the 'age' field must only accept values in $[0, 120]$, 'height' field must be between $(0, 272]$, etc.

3.1.1 Personal Information page

This page includes fields where the user is able to key in his/her own information. Each detail can be changed and saved accordingly.

- Users can click on the input text box and type in the respective values and select their gender from 2 options, Male or Female.
- To save the data, they will click on the 'Save' button and the data will always appear there even after they have navigated to another tab. To change the input data, they just have to click on the box, retype in a new data and click on 'Save'.
- The data will always appear there even after they have navigated to another tab.

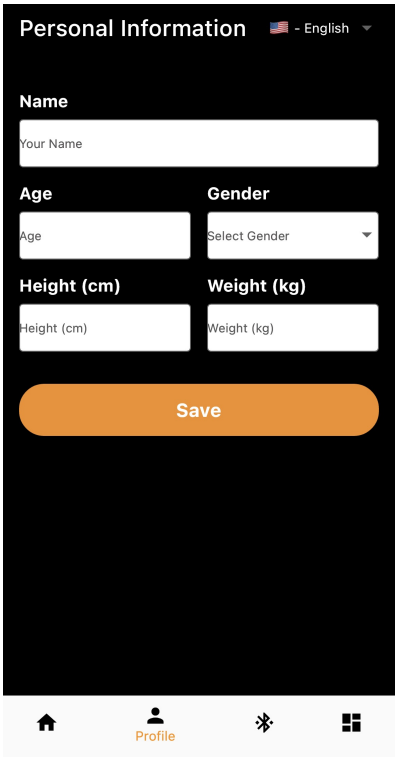
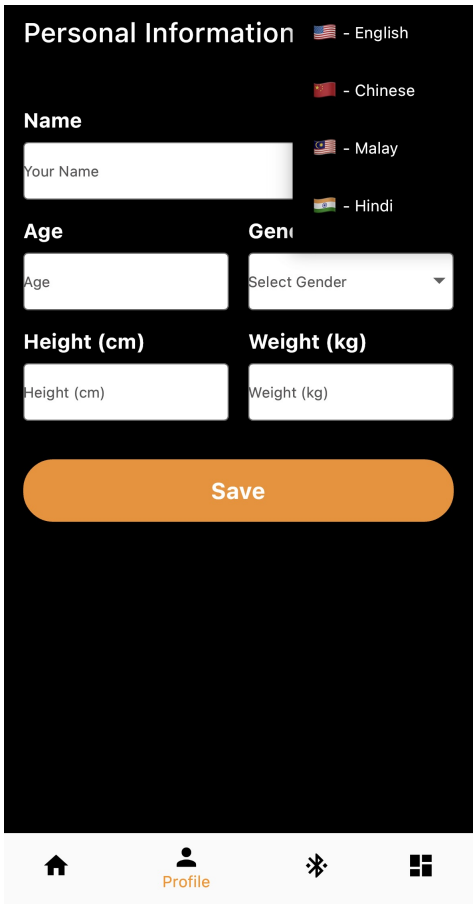


Figure 3.1: Personal Information page

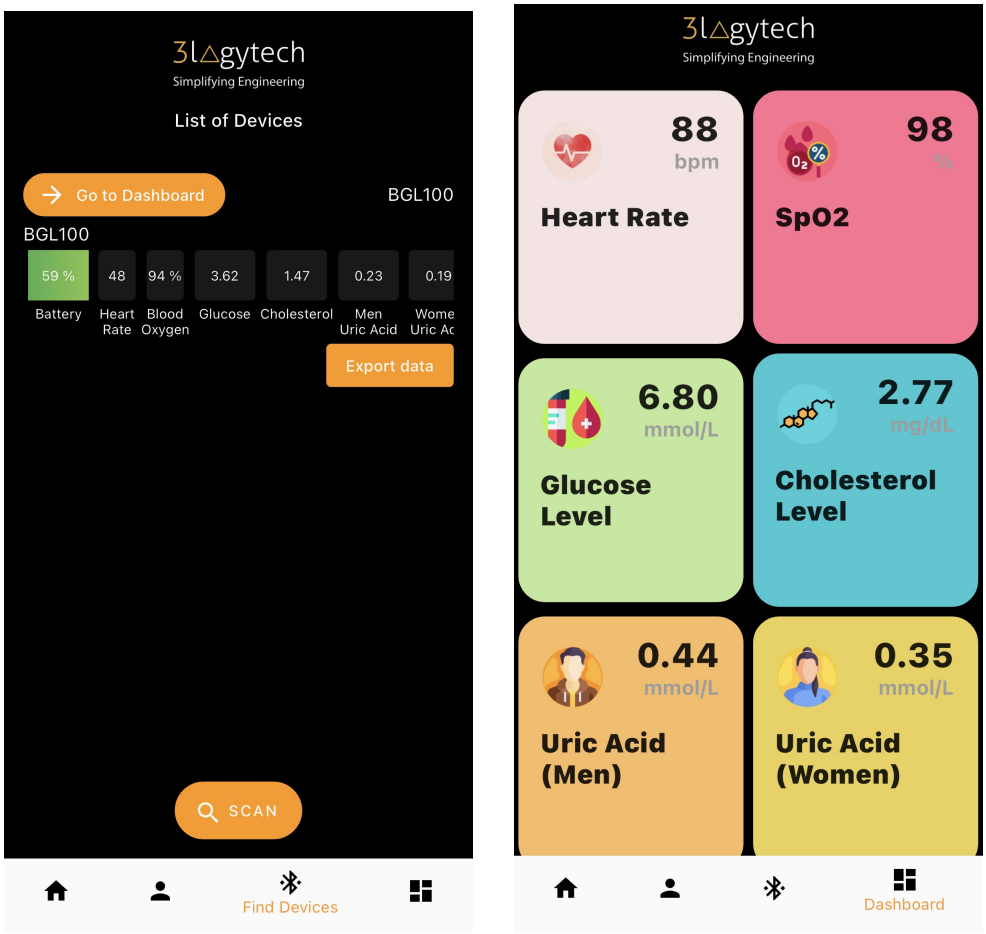


(a) All languages

(b) Example: Chinese

3.1.2 Find Devices page

This page will be where the user search for the oximeter. The live updates of the biodata received will be redirected to the next Dashboard page.
When a device is connected in 3.1.2(a), a row of data containing the variables that are being measured is displayed. They include heart rate, blood oxygen level, glucose level, cholesterol level, men uric acid level and women uric acid level.



(a) Device connected

(b) Dashboard with data

3.1.2.1 Export data

To export data, the user will click on the 'Export data' button. The data will be exported into either .xls or .csv.

The first 2 options:

- Exports the data and saves into the phone's local storage as a CSV file or an Excel file.

The last 2 options:

- Emails the data in either as a CSV file or an Excel file. The file will also be saved into the phone's local storage automatically.

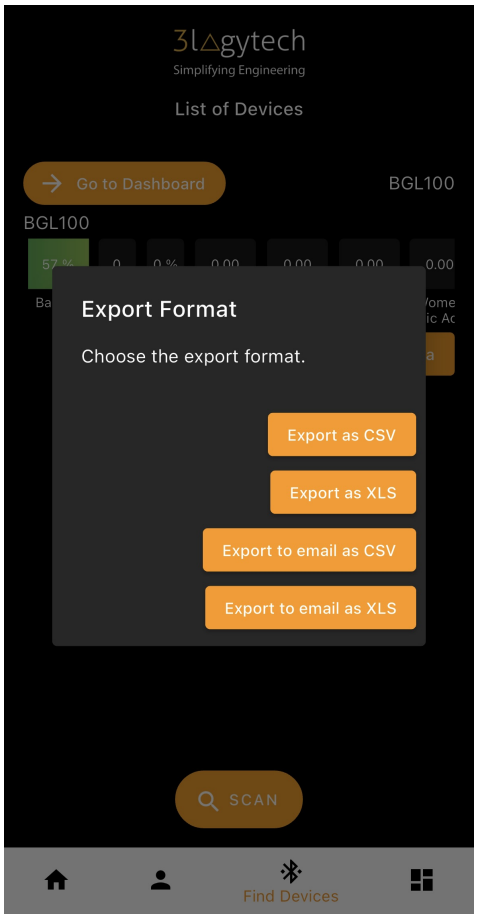
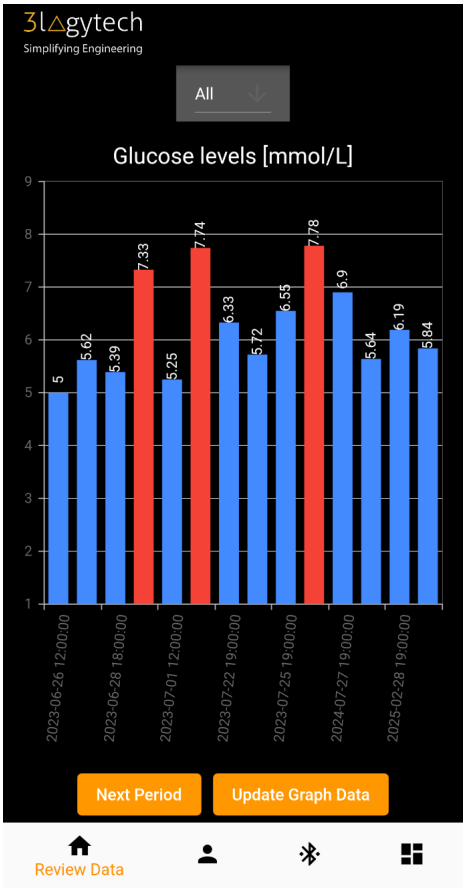


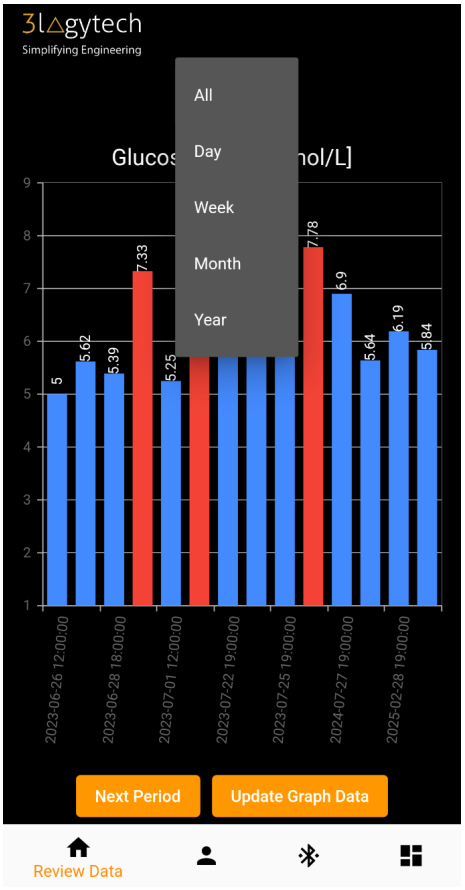
Figure 3.4: Export dialogue

3.1.3 Graph page

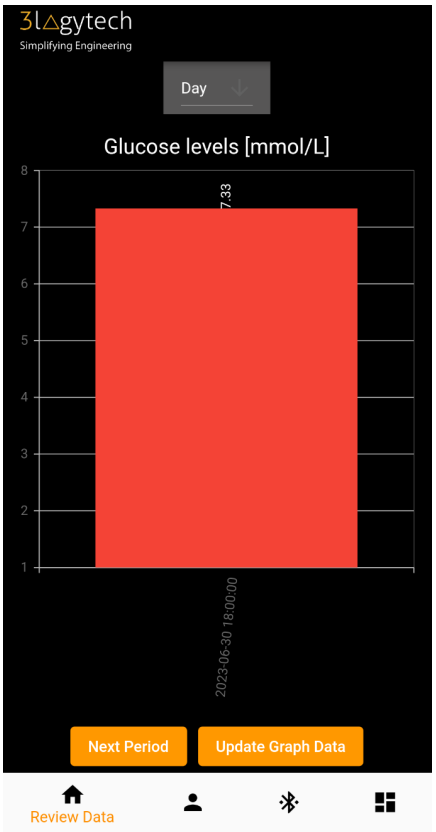
This page will be where the user search for the oximeter. The live updates of the biodata received will be redirected to the next Dashboard page.



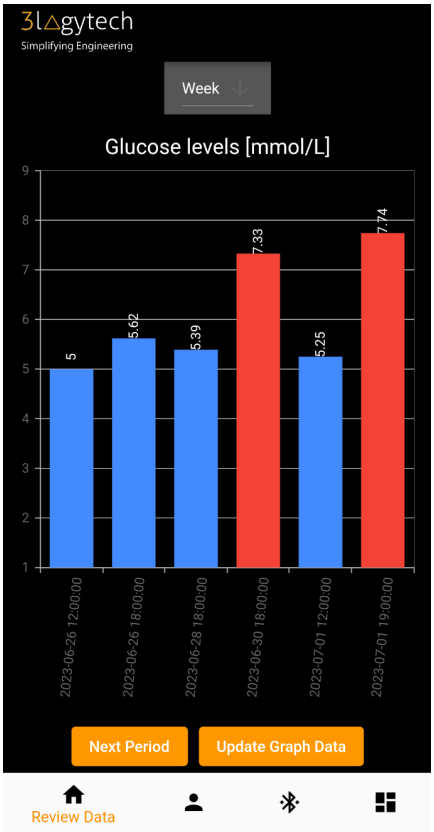
(a) Graph showing 'All' data



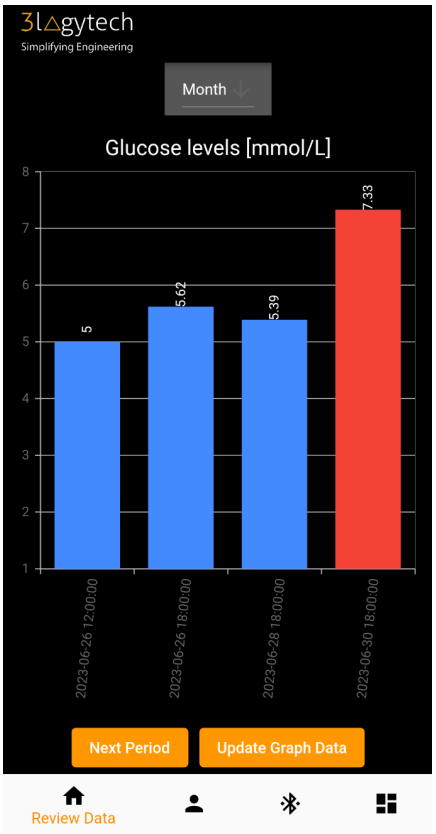
(b) Graph's filtering options



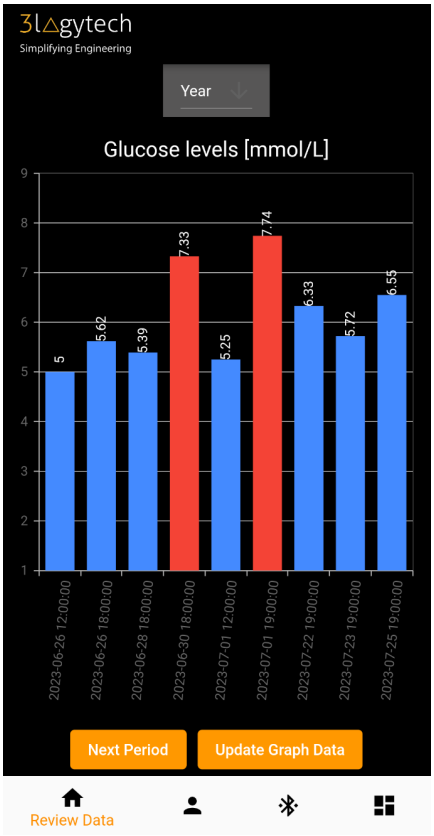
(c) Data filtered by day



(d) Data filtered by week



(e) Data filtered by month



(f) Data filtered by year

3.2 Hardware Interfaces

The app interacts with the following hardware components:

- Glucometer device
  - The app establishes connection with a compatible glucometer device via BLE.
  - A compatible glucometer is one that contains "BGL" string in the name, eg. "BGL100", belonging to Trilogy Technologies.
  - The app communicates with the glucometer to retrieve biodata.

3.3 Software Interfaces

The app interacts with the following software components:

- Operating system
  - The app is designed to run on Android (version 8.0 or higher) and iOS (version 11.0 or higher).
- Third-party libraries The app incorporates several third-party packages.
  - app_settings (version 4.2.0): Opens device settings page.
  - csv (version 5.0.2): Enables reading and writing of CSV files for import and export functions.
  - cupertino_icons (version 1.0.5): Provides a collection of Cupertino-style icons for use in the app's user interface.
  - device_info_plus (version 8.0.0): Retrieves device-specific information such as model, operating system and hardware details.
  - excel (version 2.1.0): Enables reading and writing of Excel files for import and export functions.
  - flutter_background_service (version 3.0.1): Allows the app to run in the background.
  - flutter_blue_plus (version 1.4.0): Provides Bluetooth functionality and enables communication with Bluetooth devices.
  - flutter_email_sender (version 5.2.0): Enables sending of emails from within the app using the device's local email application.
  - flutter_launcher_icons (version 0.13.1): Simplifies the process of generating an app launcher icon for different platforms.
  - flutter_native_splash (version 2.2.9): Simplifies the generation of native splash screens for the app on different platforms.
  - fluttertoast (version 8.2.1): Displays toast messages on the screen after certain actions.
  - intl (version 0.18.0): Offers internationalisation and localisation support for the app's user interface.
  - path_provider (version 2.0.15): Retrieves the file system paths for various platform-specific locations, enabling files to be retrieved and saved.
  - permission_handler (version 10.2.0): Handles runtime permissions, to provide users with the option to grant or deny specific permissions.

- shared_preferences (version 2.1.1): Stores and retrieves the app preferences and settings in a persistent manner.
- sqflite (version 2.2.8+4): Provides SQLite database support for the app, for storing of data locally.
- syncfusion_flutter_charts (version 21.2.4): Enables visualisation of blood glucose trends and generate informative graphs.

4 System Features

4.1 Personal Information page

- App must be able to accept user input.
- App must be able to store user data.
- App must allow users to change language.

4.2 Find Devices page

- App must be able to scan and detect compatible glucometer devices via BLE.
- App must give the user an option to start and stop scanning for devices.
- App must be able to display the data retrieved from the glucometer device.
- App must give the user an option to view the data in a more visually appealing dashboard.

4.3 Dashboard page

- The dashboard must be visually appealing.
- The dashboard must have big fonts for easy viewing.
- The dashboard must be display data labelled with the appropriate units of measurement.

4.4 Graph page

The graph used in the app is a bar graph.

- The graph page must provide a graph for the user to view their glucose level trends.
- The graph must only be displayed if the database is nonempty.
- The graph should provide a function for the user to view their data at specific time periods eg. day, week, month, year, all.
- The bars in the graph should be a different colour (red) to highlight to the user that the glucose level is above the threshold.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- The app must not crash when the user opens it.
- The app must be able to display each page within 3 seconds of the user opening it.

5.2 Usability Requirements

- The app design must be intuitive for ease of navigation.
- All features of the app must be clearly displayed.
- The app must offer insightful information to the user about their health or glucose level management.
- The app must provide necessary feedback to the user if there are invalid inputs.
- The app must display appropriate error messages when certain processes fail or when vital system permissions are not enabled.
- The app must support English languauge.

5.3 Reliability Requirements

- The app must not break due to any invalid input.
- The app must have all features available at all times, provided the user's device is at 100% functionality.

5.4 Security Requirements

- User data should not be disclosed without their consent.
- User data must not be seen or accessible by other users, except the user's own certified medical professional(s).
- All data must be deleted when the user chooses to delete it.

5.5 Maintainability Requirements

- Maintenance shall be conducted regularly to ensure the app and its' dependencies are up to date.
- Security or software updates shall be rolled out periodically and when necessary.

5.6 Software Quality Attributes

The app adheres to good software engineering design principles of

- Good design
  - Reusability of features
    - Methods for user account creation and storing of user data in a database can be reused for another project.
  - Testability of features
    - Each feature can be tested independently.
  - Maintainability of features
    - Ensured readability of code.
    - Included comments to help anyone else reading the source code understand what the function does.
  - Extensibility of features
    - The flexible architecture allows new modules or components to be added without affecting the existing system.
- Design Patterns
  - Single responsibility principle
    - Our application is broken down into smaller, individual functions, each with a single responsibility.
    - Improves reusability as each function can be reused in other modules.
    - Improves maintainability as each function is less likely to affect other functions.
- Design Principles
  - High cohesion
    - The modules in the system have a single, well-defined responsibility.
  - Low coupling
    - Changes in one module will not affect other functions.
    - Modules do not have overlapping responsibilities and does not affect each other at all.
  - Open-Closed
    - Extension of current functions is possible.
  - Separation of concerns
    - The code is more readable since all the components do not have overlapping functionalities.

6 Appendix A: Installation

6.1 Setup

To run this app on your local machine, follow these steps:

1. Ensure you have Flutter installed on your system. If not, you can install it from the official Flutter website: https://flutter.dev/docs/get-started/install.
2. To build: You'll need Android Studio, Xcode, iOS Simulator, and Cocoapods. You will need to sign in to your Apple Developer account to run the app on your device.

From your command line:

```
# Go into the folder
$ cd max_heart_reader

# Install dependencies
$ flutter clean
$ flutter pub get

# Run the app
$ flutter run
```

Note: If you're testing on iOS, see this guide. The default mode is `--debug` where the app cannot run in standalone mode.

Build for iOS: Create Podfile

To begin, create a `Podfile` in the root directory of your iOS project. The `Podfile` specifies the dependencies for your project. Open the terminal and navigate to your project's root directory:

To begin, create a 'Podfile' in the root directory of your iOS project. The 'Podfile' specifies the dependencies for your project. Open the terminal and navigate to your project's root directory:

```
cd /path/to/YourProjectIOSFolder

# If Podfile does not exist
touch Podfile

pod install

# Open .xcworkspace
open Runner.xcworkspace

# Click on the play button in Xcode to build and run the app.
```

6.2 Internationlization

The app containing the code for the different languages is located in the *lib/l10n* folder. *l10n* is the naming convention for localization. The number 10 is the number of letters between the *l* and *n* in the word localization.

To add or edit more languages for the app, follow these steps as taken from the official Flutter documentation.
The steps listed more than suffice for the purposes of this app. The l10n.yaml file is already created, which tells the compiler where to find the *.arb* files and will auto generate the required *.dart* files.

To generate new *app_localizations_*{*YourLocaleHere*}*.dart* files, run in command line:

```
flutter gen-l10n
```

7 Appendix B: Troubleshooting

This page lists the common problems encountered during the development of the app and how they can be resolved. Do note that this list is not exhaustive and there may be other solutions to the problems.

- Change the name of the app: https://stackoverflow.com/questions/49353199/how-can-i-change-the-app-display-name-build-with-flutter
- Change the icon of the app: https://pub.dev/packages/flutter_launcher_icons
- Change applicationId and bundle identifier to prevent overwrite: https://stackoverflow.com/questions/58229150/how-to-run-a-copy-of-flutter-project-without-overwriting-the
- Run background service to keep uploading data when app is turned off: https://pub.dev/packages/flutter_background_service
- Splash screen: https://pub.dev/packages/flutter_native_splash/example https://www.youtube.com/watch?v=dB0dOnc2k10&ab_channel=JohannesMilke
- Loading screen: https://codewithflutter.com/create-splash-screen-in-flutter-app/
- Enable and Check Location Permissions: https://pub.dev/packages/permission_handler
- To compile on iOS:
  - nil:NilClass for Flutter App / CocoaPod Error: https://stackoverflow.com/questions/67443265/error-regarding-undefined-method-map-for-nilnilclass-for
  - Make sure to click "Trust" on iPhone when connecting iPhone to PC
  - bundle ID error: https://stackoverflow.com/questions/51098042/how-to-get-bundle-id
  - "Untrusted Developer" after XCode build is done: https://developer.apple.com/forums/thread/660288 General -¿ VPN & Device Management -¿ Developer App -¿ Verify
  - Make sure that the sign in is successful in XCode: https://docs.flutter.dev/deployment/ios Go to XCode -¿ Runner -¿ Signing & Capabilities -¿ Team -¿ Select apple developer account
  - Dart SDK is not configured: https://stackoverflow.com/questions/48650831/dart-sdk-is-not-configured Click on settings gear (IDE and Project Settings) -¿ Preferences -¿ Language & Framework -¿ Flutter -¿ choose flutter SDK
  - (when compiling Android app on Macbook): Execution failed for task ':app-compileFlutterBuildDebug': https://stackoverflow.com/questions/61930007/how-to-solve-execution-failed-for-task-appcompileflutterbuilddebug