Department of Computer Science
University of Hong Kong
Final Year Project

Development of a Mobile Application for
Coronary Heart Disease Risk Prediction
Final Report

Written by:
Hadipranata Mellisa (3035663019)

Group members:
Harney Vieri (3035663368)
Chan Hay Yin (3035686047)
Hom Long Hin (3035692450)

Supervisor:
Dr. Loretta Choi

18 April 2023
Abstracts

Heart diseases have consistently been one of the top three causes of death over the past century, with coronary heart disease (CHD) as the leading cause. According to a government organization, HealthyHK, an average of 10.5 people die from CHD every day in 2020. The Fortunately, with the advancement of technology in digital health nowadays, such as wearable health trackers, managing one’s health has been made simpler. Therefore, the utilization of digital health technology is the basis of this project. The main objective is to raise awareness of one’s risk of developing CHD, which will be achieved by developing a mobile application that can be well-integrated with a wearable device. This will allow users to monitor their physical activity data and keep track of their CHD risks, and thus showing users the importance of physical activity in reducing the risk of CHD. By the end of this project, users will be able to perform self-assessments of their CHD risk through their mobile phone, without the need of seeking medical assistance. Through the use of digital health technology, we hope to make CHD risk assessment more accessible to the general population, and ultimately reduce the number of CHD-related deaths. This project is in collaboration with a team from the School of Public Health at The University of Hong Kong, who will provide this project with the algorithm for predicting CHD risk. On the other hand, our team is responsible for developing the mobile application and implementing the risk prediction feature by translating the given algorithm into code and integrating it into the mobile application.

This report will provide a detailed account of the entire process of this project from start to finish. Specifically, the report will discuss the methodology used, including the design phase, implementation, and results. The report will also explain the engineering choices made and the reasoning behind them, including the benefits and limitations of the chosen methodology and any alternatives that were considered and explored. Although we were successful in developing the application within the project's timeline, there are potentially rooms for future improvements. Recommendations for future work are proposed if the project is to be continued in the future. By comparing different approaches, the most precise result for predicting CHD risk can be achieved.
Acknowledgment

We would like to express our deepest gratitude to our final year project supervisor Dr. Loretta Choi and her team for their continuous guidance and feedback throughout the project. Without their expertise and advice, this project would not have been able to proceed as smoothly. We would also like to sincerely thank Dr. Youngwon Kim and his team from the School of Public Health at The University of Hong Kong for believing in our team and allowing us to collaborate and take responsibility for developing the mobile application of this interesting project.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>4</td>
</tr>
<tr>
<td>ABBREVIATIONS</td>
<td>5</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>6</td>
</tr>
<tr>
<td>2. METHODOLOGY</td>
<td>7</td>
</tr>
<tr>
<td>2.1. PLATFORM AND EQUIPMENT SETUP</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1. User Data</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2. Fitbit</td>
<td>8</td>
</tr>
<tr>
<td>2.1.3. Risk Prediction Algorithm</td>
<td>9</td>
</tr>
<tr>
<td>2.2. SYSTEM ARCHITECTURE DESIGN</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1. Fitbit Web API</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2. Frontend Application</td>
<td>13</td>
</tr>
<tr>
<td>2.2.3. Frontend Admin</td>
<td>14</td>
</tr>
<tr>
<td>2.2.4. Backend and Database</td>
<td>15</td>
</tr>
<tr>
<td>3. RESULTS</td>
<td>17</td>
</tr>
<tr>
<td>3.1. FRONTEND AND USER INTERFACE (UI): MOBILE APP</td>
<td>17</td>
</tr>
<tr>
<td>3.2. FRONTEND AND USER INTERFACE (UI): ADMIN PAGE</td>
<td>21</td>
</tr>
<tr>
<td>3.3. BACKEND AND DATABASE DESIGN</td>
<td>24</td>
</tr>
<tr>
<td>3.3.1. Firebase and Firestore</td>
<td>24</td>
</tr>
<tr>
<td>3.3.2. Flask REST API Server</td>
<td>26</td>
</tr>
<tr>
<td>3.4. RISK PREDICTION</td>
<td>29</td>
</tr>
<tr>
<td>3.4.1. User Data Collection</td>
<td>29</td>
</tr>
<tr>
<td>3.4.2. Risk Calculation</td>
<td>29</td>
</tr>
<tr>
<td>3.4.3. Storing Risk Prediction Results</td>
<td>31</td>
</tr>
<tr>
<td>4. DIFFICULTIES AND LIMITATIONS</td>
<td>32</td>
</tr>
<tr>
<td>4.1. FITBIT REQUEST RATE LIMIT</td>
<td>32</td>
</tr>
<tr>
<td>4.2. FITBIT MANUAL SYNC</td>
<td>33</td>
</tr>
<tr>
<td>4.3. REACT NATIVE PERFORMANCE</td>
<td>33</td>
</tr>
<tr>
<td>4.4. UNAVAILABLE PUBLIC DTC API</td>
<td>34</td>
</tr>
<tr>
<td>5. CONTRIBUTIONS</td>
<td>35</td>
</tr>
<tr>
<td>5.1. FRONTEND</td>
<td>35</td>
</tr>
<tr>
<td>5.1.1. Self-input Form</td>
<td>35</td>
</tr>
<tr>
<td>5.1.2. Home Page</td>
<td>36</td>
</tr>
<tr>
<td>5.1.3. Profile Page</td>
<td>38</td>
</tr>
<tr>
<td>5.1.4. Risk Page</td>
<td>39</td>
</tr>
<tr>
<td>6. FUTURE WORK</td>
<td>41</td>
</tr>
<tr>
<td>6.1. EXPAND COMPATIBILITY WITH WEARABLES</td>
<td>41</td>
</tr>
<tr>
<td>6.2. IMPLEMENT FITBIT ACCELEROMETER</td>
<td>41</td>
</tr>
<tr>
<td>6.3. DTC REPORT.</td>
<td>41</td>
</tr>
<tr>
<td>6.4. ADDITIONAL FEATURES FOR ADMIN PAGE</td>
<td>42</td>
</tr>
<tr>
<td>7. CONCLUSION</td>
<td>43</td>
</tr>
<tr>
<td>8. REFERENCES</td>
<td>45</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 – Fitbit Web API parameters ................................................................................. 8
Figure 2 – The system architecture design diagram .............................................................. 10
Figure 3 – Fitbit oAuth 2.0 authorization URL format .......................................................... 11
Figure 4 – POST request to Fitbit server ............................................................................ 11
Figure 5 – Expo application running on two platforms ......................................................... 14
Figure 6a – The initial UI design of the application............................................................... 17
Figure 6b – The current UI design of the application ............................................................. 17
Figure 7 – Account creation workflow .............................................................................. 17
Figure 8 – Edit user workflow ............................................................................................. 19
Figure 9 – Risk page ........................................................................................................... 20
Figure 10 – Community page ............................................................................................... 21
Figure 11 – Admin page UI ................................................................................................. 22
Figure 12 – Create forum pop-up ......................................................................................... 22
Figure 13 – Create article pop-up ......................................................................................... 23
Figure 14 – Article pop-up ..................................................................................................... 23
Figure 15 – Firebase console ............................................................................................... 23
Figure 16 – User data schema ............................................................................................. 25
Figure 17 – Community-related schemas .......................................................................... 25
Figure 18 – Swagger UI ........................................................................................................ 26
Figure 19 – Swagger example ............................................................................................. 27
Figure 20 – Fitbit manual sync ............................................................................................ 33
Figure 21 – Validation .......................................................................................................... 33
Figure 22 – Automatically calculated fields .......................................................................... 35
Figure 23 – Icon helper components .................................................................................... 36
Figure 24 – DataCard component ......................................................................................... 37
Figure 25 – Bar graph component ......................................................................................... 37
Figure 26 – DetailsCard component ..................................................................................... 38
Figure 27 – Line chart components ....................................................................................... 39
Figure 28 – Risk Summary Component .............................................................................. 40
Figure 29 – Risk Detail page ................................................................................................. 40
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CHD</td>
<td>Coronary Heart Disease</td>
</tr>
<tr>
<td>LBM</td>
<td>Lifestyle Based Model</td>
</tr>
<tr>
<td>PKCE</td>
<td>Proof Key for Code Exchange</td>
</tr>
<tr>
<td>BMR</td>
<td>Basal Metabolic Rate</td>
</tr>
<tr>
<td>ENMO</td>
<td>Euclidian Norm Minus One</td>
</tr>
<tr>
<td>MET</td>
<td>Metabolic Equivalent of Task</td>
</tr>
<tr>
<td>DTC</td>
<td>Direct-to-consumer</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create-read-update-delete</td>
</tr>
</tbody>
</table>
1. Introduction

Cardiovascular diseases (CVD) are responsible for a significant number of deaths globally. According to the World Health Organization (WHO), CVD caused the deaths of 17.9 million people in 2019, which accounts for 32% of all worldwide fatalities [1]. In Hong Kong, CVD has been the third most common cause of death in the past 60 years, with coronary heart disease (CHD) being the primary cause [2]. In 2020, a daily average of 10.5 individuals died in Hong Kong due to CHD, as reported by HealthyHK, a government organization that focuses on public health [3]. The risk of developing CHD is influenced by various lifestyle factors such as Body Mass Index (BMI), blood pressure, as well as genetic factors [4].

Currently, the assessments for predicting one’s risk of developing CHD are mainly performed in clinics by medical professionals. Another method to predict one’s CHD risk is by using a risk calculator website, such as https://cvdcalculator.com/, and inputting traditional risk markers such as blood pressure, cholesterol level, etc. However, this method does not reflect the dynamic changes of one’s CHD risk, and professional help is still needed. Hence, the currently available methods are not accessible for everyone.

Luckily, the advancement of technology, particularly in the mobile application and health wearables industry, has become increasingly rapid over the years, allowing one to easily get health insights and take appropriate actions to mitigate any risk of disease. Therefore, this project aims to develop a mobile application that can integrate with a wearable health device, which will help provide the data needed for the application to calculate users’ risk. Additionally, with the physical activity data obtained from the wearable device, this project aims to be able to show users how their physical activities can mitigate their CHD risk in real-time. By utilizing the advancing digital health technology, this project hopes to be able to provide a more accessible yet accurate tool to predict one’s CHD risk and help more people to understand their own CHD risk and how to reduce it. Additionally, our team is also in collaboration with a team from the School of Public Health of The University of Hong Kong led by Dr. Youngwon Kim for this project. While Dr. Youngwon Kim and his team provide this project with the risk prediction algorithm, our team is responsible of the technological aspects of the project, including mobile application development and implementation of the risk prediction algorithm into the application. Therefore, this report will elaborate in detail
the methodology that has been implemented, the current results that have been achieved, any difficulties and limitations that have been encountered, individual contributions made, and recommendations for future work.

2. Methodology
This section will elaborate on the methodology that have been implemented in this project, including the platform and equipment set up, system architecture, and the development tools that have been used in this project along with the justifications. This section will also include any alternatives that have been explored and the reason why they are not incorporated into this project.

2.1. Platform and Equipment Setup
The following subsections are several development platforms and equipment setups that needed to be done in the initial stages of this project. The setups are mainly necessary for collecting relevant users’ data used as an input for the CHD risk prediction.

2.1.1. User Data
Two kinds of user data will be collected in this project: user personal self-input data and physical activity data. Both sets of data are crucial in this project as those are the required data to calculate the CHD risk prediction. Physical activity data, in particular, can be used to show users their changing risk value over a period of time since it has been proved that physical activity inversely correlates with undesirable CVD outcomes, such as CHD [5]. The collection of physical activity data will be elaborated further in the next subsection. The other set of data that is critical for the risk calculation is users’ health-related data, including BMI, smoking status, alcohol consumption, medication use of cholesterol, blood pressure, and insulin, age, and gender. All of these data are collected upon account creation, where users are required fill in a self-input form regarding their personal and health information.
2.1.2. Fitbit

As mentioned previously, physical activity data is needed to produce a more dynamic risk prediction, and since one of the goals this project is aiming to achieve is to able to show users how their CHD risks may dynamically change in correspond to their daily physical activities, physical activity data a crucial data that needs to be focused on in this project. In order to capture users’ physical activities in real time, this application needs to be connected to a wearable hardware device that can provide relevant health-data with high accuracies for the risk calculation. Fortunately, with new technological innovations in digital health that can be found today, tracking users’ physical activity and other health-related data can be done easily by using small wearable devices, such as smartwatches [5]. There are numerous smartwatches in the market, such as Fitbit, Apple Watch, Garmin, etc. However, given the limited time for this project, this project mainly focused on integrating data with one smartwatch brand. Our team has considered several popular brands, such as the ones mentioned previously, and decided to choose Fitbit given its affordability, compatibility with popular mobile phones and tablets, and ability to provide the essential data for this project [6]. Fitbit is also one of the most used fitness trackers in the world. It has been recorded in 2021 that a total of 111 million users were registered in the Fitbit database [6].

Figure 1 – Once developers has successfully signed-up for a Fitbit developer account, they can start integrating Fitbit API into the application by configuring their application according to the parameters of the Fitbit developer application as shown in the figure. Inputting the configured Client ID, Client Secret, and Redirection URL.
Furthermore, Fitbit provides a well-documented web application programming interface (API), which is very helpful for the development of this project since it can deliver all the essential data needed for this project. Upon connecting with users’ Fitbit account and gain access to collect users’ Fitbit data, the application can easily fetch users’ data from Fitbit by providing the given access token when calling the web API. Some initial setup needed to be done before being able to call the web API. A Fitbit developer account had to be created first, then we received the client id and client secret data to be used in the application for configurations (see Figure 1).

2.1.3. Risk Prediction Algorithm

As mentioned earlier, this project is a collaboration project with a team from School of Public Health at The University of Hong Kong led by Dr. Youngwon Kim. The algorithm that has been developed by Dr. Youngwon Kim and his team is an equation of the Lifestyle-based Model (LBM) for CHD risk prediction that takes into account of users’ lifestyle data as risk predictors, which are age, sex, BMI, diet, smoking status, alcohol consumption, physical activity (in milli-g), and medication use of cholesterol, blood pressure, and insulin. This algorithm has been translated into code and implemented in the backend service of our application. The equation can estimate someone’s personalized CHD risk within the next 10 years by making use of coefficients assigned to each lifestyle indicator set by Dr. Youngwon Kim and his team. The coefficients used in the algorithm are primarily estimated using data obtained from UK Biobank, which allows for each individual’s unique genetic CHD risk accurate prediction. Unfortunately, since the algorithm has not been publicly published, the exact equation cannot be disclosed in this report.
2.2. System architecture design

This part of the report will explain the current system architecture design of the application. The diagram shown in Figure 2 gives a clear idea on how each major components of the application interacts with each other to construct a fully functioning application. In the subsequent sub-sections, deeper explanation and details on each component will be presented.

![System Architecture Diagram](image)

*Figure 2 – The system architecture design diagram that has been designed illustrates the flow of communication between different parties and components of the application. a) The frontend and backend is the main components which will handle communications with other parties. b) The users will have to input their lifestyle health-related data into the client. c) The Firebase is installed in the backend service and connected to the database of the application, where all user’s data will be stored. d) An admin page client is also developed to manage community feature of the application. e) The Fitbit API will be called by the frontend for user authentication and by the backend to get users’ physical activity data for the risk prediction calculation.*

2.2.1. Fitbit Web API

As previously mentioned, our application is collecting users’ physical activity data through Fitbit. The data collected by Fitbit from users’ Fitbit watches can be fetched by our application through the Fitbit public web API. However, before gaining access to fetch users’ Fitbit data, an OAuth2 authentication process has to be completed. Fitbit protects user data by using Authorization Code Grant Flow with Proof Key for Code Exchange (PKCE) for the OAuth2 process [7]. As can be seen from the system design diagram above, the Fitbit authorisation (OAuth2) is done on the server side. The server-side is responsible for generating a unique code verifier and code challenge through SHA-256 encryption. Once generated, the client will request an authorization URL that contains the code challenge created earlier (see Figure 3)
After accessing the URL, the client will communicate with the Fitbit server and receive an authorization code if the sign-in is successful.

Once the authorization code is received, it will be sent to the server side and exchanged for an access token and refresh token using a POST method to the specified URL (see Figure 4).

Once the server obtains the access token and refresh token, it will save them to the Firebase database based on the user's details. These tokens can be utilized in the future to exchange the user's requested physical activity data as per the documentations provided by Fitbit.

2.2.1.1. Calories Burned and Active Time

Since the algorithm requires physical activity data in milli-g unit, our current implementation is estimating the value by converting the calories burned and the activity time in a day into physical activity data using users’ basal metabolic rate (BMR). All of these input data can be obtained from users’ Fitbit account through the web API. This implementation will be further elaborated in section 3.4.2.

2.2.1.2. Explored Alternative – Accelerometer

Another possible alternative to obtain the physical activity data in milli-g is by the estimation of users’ raw accelerometer data.
Accelerometer data can be collected by using accelerometer sensors present in wearables, including Fitbit. Fitbit provides an Accelerometer API which can give users’ raw accelerometer data collected from the Fitbit watch. Through the collected raw accelerometer data, the physical activity data can be calculated as Euclidian Norm Minus One (ENMO) value throughout the day. ENMO itself is a summary score of raw acceleration levels, which can be calculated by counting the vector magnitude of acceleration values in the x, y, and z axes minus one gravitational unit (1g), to consider the gravitational unit of when the device is still [8]. The formula is as follows:

\[ 1 - \sqrt{x^2 + y^2 + z^2} \]

The risk prediction can use the total ENMO values throughout the day as the physical activity data input. This method of getting users’ physical activity data was the main method that was initially explored for the project. However, upon further research, our team discovered that the Accelerometer API that Fitbit provides for developers is not available in the web API set, but instead in the device API set. According to Fitbit documentation, the only way to call the device API is through an in-watch application [9]. Hence, to utilize this method for the project, a separate in-watch application would need to be developed in addition to the mobile application. Due to the time constraints of the project, this method was no longer pursued as it would be time consuming to develop another application. Furthermore, after thorough investigation, there would be no guarantee that this method will work as expected for the purpose of the project which is to track users’ movement at all time when they are using the wearable device. To achieve this function, the in-watch application would have to run constantly in background to track the acceleration of every users’ movement changes. Through further research, the
reason that many devices, including Fitbit, do not allow application be run constantly in background when using the accelerometer sensor is due to the fact that accelerometer sensors generally consume a lot of processing and battery power of the device [10]. Hence, if used excessively, such as running continuously in background, devices health could be damaged. As a result, this method was abandoned and replaced with the estimation alternative as mentioned in section 2.2.1.1 earlier.

2.2.2. Frontend Application

As can be seen from the system architecture design diagram in Figure 2 above, the frontend service has several key roles in the application. The main function of frontend is to serve as an interface for users to interact with, including viewing, changing, and inputting data into the application. From the diagram, it can also be seen that the lifestyle-based health data is collected in the frontend and then sent to backend for further processing. As mentioned previously, users are required to provide their personal health details upon account creation to be used as crucial inputs for the risk calculation. Generally, frontend is responsible for collecting data from users and sending the data to the backend to be stored in DB as necessary. Frontend communicates with backend through API calls, where frontend can request and send data to the backend and backend can receive and send data to frontend. Frontend also communicates with the Fitbit server through backend to fetch relevant data to be shown in the UI, such as users’ physical activity data.
The frontend service is developed using Expo, which is an open-source framework for building React Native applications. There are a couple of factors that were taken into consideration when deciding on which development tools should be used for frontend. First, React Native is a cross-platform development tool that can run on both Android and iOS in a single codebase. This means that by only developing one application, two application can be produced without any additional setups, saving overall development time while also reaching more target users [11]. Second, development in a single codebase may also promote feature and design consistency across platform, as can be seen in Figure 5. Although it is undeniable that the performance of native applications (i.e., iOS using Swift Android using Kotlin) are arguably better, our application does not include any heavy animation and/or computation to be done in the frontend. Therefore, the benefits of using React Native counterbalance the drawbacks.

![Figure 5 – Our Expo application run in two different simulators, Android (left) and iOS (right). The application is using the same codebase but is able to run with the same functionality while also having similar overall design.](image)

2.2.3. Frontend Admin

Another frontend component that can be found in the system design is a web-application that serves as the Admin page. Currently, the admin page is only used for managing the Community in the mobile application, including creating and deleting forums and articles. The admin of this
application is the team that will take over this project upon the completion of this Final Year Project, which is our project collaboration partner, the team from School of Public Health at HKU. Since the admin page has very minimum functionality, as of now, we do not see the need to deploy this web-application to the cloud, so it can only be run locally by the admin. The frontend admin also communicates with the same backend server to create and delete forums and articles from the database.

This web-application is developed using React.js for its faster render time, ease of use, and more stable code structure [12]. Additionally, since the frontend application is developed using React Native, using React for this web-application promotes some code consistency in the frontend part.

2.2.4. Backend and Database

Lastly, the backend and database, which together serve as an API server which responds to client queries, are two interconnected components that holds important roles in the system. Since this application deals with large users’ data and needs to do computations for the CHD risk calculation, our team separated the client-side tasks from the heavy data processing tasks, to avoid overloading the application with too many task which may lead to slow performance. Therefore, the data processing tasks are offloaded to a separate backend server, including the CHD risk calculation.

The backend also holds the key role of authenticating users to connect their Fitbit account with the application. This authentication needs to be done before the application can fetch relevant users’ physical activity data from Fitbit. Since the response obtained from Fitbit may hold too many information or data that are not needed in our application, backend helps process the response data retrieved from Fitbit before sending as a response to the client request. This is also done to help reduce the payload size of each response to requests made by the client and prevent overloading client with too many data to process.
As for the development tools, Flask and Firebase will be utilized for the backend and database, respectively. Firebase is a backend-as-a-service platform provided by Google, which offers several advantages for this project. One of its major benefits is its extensive and detailed documentation that facilitates application development and debugging. Additionally, Firebase provides many helpful features for mobile application development such as authentication and storage. Firestore, a NoSQL cloud database available on Firebase, is also being used in this project. The tight integration between the backend and database, or Firebase and Firestore, can enhance server performance.
3. Results

In this section, the current progress of this project along with several limitations encountered will be explained. The following subsections will elaborate on the current developments of frontend and backend services.

3.1. Frontend and User Interface (UI): Mobile App

The final UI design of the application does not differ much from the initial UI design, as can be seen from Figure 2a and Figure 2b below. The Genetics page from the initial design has been removed from the current design because of a difficulty our team faced in obtaining users’ DTC test results, which will be further explained in section 4.4.

Figure 67a - The initial UI design of the application. The leftmost frame and the second frame from the left are the Home page. The middle frame is the Genetics page. The second frame from the right is the Risk page. The leftmost frame is the Community page.

Figure 6b - The final UI design of the application. The leftmost frame and the second frame from the left are the Home page. The middle frame and the second frame from the right is the Risk page. The leftmost frame is the Community page.
The overall design of the application is simple and straightforward by also utilising bright colours to increase the attractiveness of the application. The application also makes use of data visualization tools such as bar graphs and line charts to show users’ weekly steps and CHD risk trends (i.e. one-month trend and one-year trend), respectively, to better show users how their physical activities may affect their 10-year CHD risk.

Since the risk calculation needs users’ personal health-related data and physical activity data from users’ Fitbit, the application will collect the relevant data upon account creation and require users to connect their Fitbit account so that the application can also fetch users’ physical activity data from Fitbit (see Figure 7).

Users can always update their personal data to make sure that their risk is calculated with accurate data (see Figure 8 on the next page). Some fields in the personal data form, such as age and BMI, are automatically calculated by the app to avoid any data mismatch.
In the Home page, as shown in the first frame of Figure 8, users can view their physical activity data that is fetched from their Fitbit account, including active calories, distance, activity, steps, total calories (Basal Metabolic Rate in calories + active calories), and weekly steps, which is visualised by a bar graph. Since all of these data will have to be fetched from Fitbit every time the FE renders the page, the number of API calls to the Fitbit server may be very large. These frequent calls may lead to an error raised from the Fitbit server, which will be elaborated further in section 4.1. With that problem in mind, we limit the number of API calls made to the Fitbit server by only updating the data shown in the Home page once every 10 minutes. For every data fetch from Fitbit, the fetch time will be recorded in the application by using the AsyncStorage library by Expo, which acts as an asynchronous local storage for the application. Before making any fetch, the application will first check the last Fitbit fetch time and only do another fetch when 10 minutes have passed since the last fetch.

The Risk page shows all risk-related data about the user. As mentioned earlier, the 10-year CHD risk trends are visualised by using a line chart, and users can switch between their risk trends (i.e. one-month trend or one-year trend). For a more detailed view of the risk data, the application allows users to view all of their risks from previous days, as can be seen in Figure 9. Each of the risk rows can also be pressed to show the details of each risk, including the input data that
have been used to calculate each risk, such as age, gender, BMI, medication use for blood pressure, cholesterol, and insulin, smoking status, alcohol consumption, and average calories. With this feature, users can better track their lifestyle factors and learn how each may affect their CHD risk.

Users can also keep track of their CHD risk in more detail in the summary box, where they can view their today’s risk, yesterday’s risk, and last month’s risk including how much higher or lower their today’s risk is compared to yesterday’s and last month’s. Since the today’s risk is constantly calculated during the day by using data from users’ Fitbit, the application also have to call the Fitbit server numerous of times. These frequent calls may lead to an error from the Fitbit server, which will be elaborated further in section 4.1. Subsequently, we avoid this problem by adding the ‘Refresh’ button beside the today’s risk which can only be pressed when 10 minutes have passed since the last time the today’s risk is updated. With this button, the API call to BE and Fitbit can be minimized, thus lower the chance of receiving error from the Fitbit server.

Figure 9: The Risk page showing user’s 10-year CHD Risk. The line chart on the left shows the one month trend of user’s CHD risk by plotting the risk of every day before today. Upon clicking the button under the chart, user can view the one year trend, and the line chart now plots the risk of end of each previous months, as can be seen in the middle frame. The rightmost frame shows the page that shows all of the risks that have been recorded in the system.
As can be seen in Figure 10, there is also a Community page where users can read informative health-related articles that may piqued their interests. Users can also share their thoughts about the articles they read by posting comments and read other users’ comments. To ensure the credibility of the article contents, the articles can only be posted by admins of the application through the admin page that will later be explained in more detail in section 3.3.

Figure 10 - The Community page. The leftmost frame shows the list of forums, with each containing related articles as shown in the second leftmost page. The two frames on the right show the article page. Upon scrolling down the article page, users can find the comment sections where they can view and also post their comments.

3.2. **Frontend and User Interface (UI): Admin page**

As mentioned in section 3.1., the forums and articles added to the application are created by the admin of the application. Currently, this admin page only serves as an interface to help the admin to maintain the Community in the application. Admin can creating new forums and articles, and deleting forums and articles as necessary. Since the Community page is a side feature of the application, not much of the development time was allocated for the admin page, and. As can be seen in Figure 11, the overall design of the admin page is very simple and straightforward. The admin page is just a one-page web application, containing a list of forums with their respective list of articles.
Although the design of the web-application is simple, all of the essential components for managing the Community feature are present. First, beside every forum and in every article card, a delete icon button is provided for easy deletion. When the delete icon is pressed, the admin frontend will send a delete request to the backend and the selected object will be deleted from the database accordingly. A button for creating a new forum is also provided on the top right of the page, with the label “New Group”. Once the button is clicked, a modal will show up on the screen and there are necessary inputs that would need to be filled (see Figure 12). Upon submission of the complete data, the admin frontend will send the forum data to the backend to be stored in the database. The button “New Post” behaves similar to “New Group” but for creating new articles, and the creating workflow is also similar (see Figure 13 on the next page).
Once the articles have been created and stored in the database, the admin page will show the posted articles, and the admin can also preview them by clicking on the article card component. When the card is clicked, the frontend admin will fetch the article data from the database through backend, and article content will be shown like in Figure 14 below.

Figure 13 – A similar pop-up form will appear when the “New Post” button is clicked. The admin will need to input the article author, image, forum that it belongs to, title and content of the articles.

Figure 14 – An article pop-up will appear by clicking one of the article cards shown in the article summary screen when admin first entered the admin application.
3.3. **Backend and Database Design**

The implementations for backend development can be separated into two: Firebase, together with Firestore, and API development using Flask. The detail of each implementation will be discussed in the sub-sections below.

3.3.1. **Firebase and Firestore**

Firebase implementation was mainly for the setup for integrating it into both the Flask application and the frontend application. Figure 15 below shows the initialised Firebase application and the required parameters such as project id, client secret, etc. that we need to integrate into our Flask application to use Firebase features.

![Google Firebase console](image)

*Figure 15 – The Google Firebase console showing the relevant parameters that would need to be configured by developers into the application in order to connect their application into Firebase and Firestore.*

Schemas or data models have also been set to ensure data consistency of each collection. We want to make sure that each entry has all the essential attributes that are needed in the application before storing it in the database. The schemas that are used in the application can be found in Figure 16 and Figure 17 shown on the next page.
Figure 16 – The schema that we have designed for users of the application, we need to store both their personal and health data which is one of the required input for the risk prediction algorithm.

```javascript
infoUser = Model(
    "info",
    {
        "firstName": fields.String(description="User's surname"),
        "lastName": fields.String(description="User's given name"),
        "dob": fields.String(description="Date of Birth (YYYY/MM/DD)")
    },
    
    infoHealth = Model(
        "health",
        {
            "insulin": fields.Boolean,
            "cholesterol": fields.Boolean,
            "diet": fields.String,
            "smokingStatus": fields.Integer,
            "alcoholConsumption": fields.Integer,
            "bloodPressure": fields.Boolean,
            "medications": fields.List(fields.String),
            "sex": fields.String,
            "height": fields.Float,
            "weight": fields.Float,
            "bloodType": fields.String,
        }
    )
)
```

Figure 17 – The schema that we have designed for the community feature. This includes forum, post and comment. The schema is kind of recursive since forum contains posts and post contains comments.
```javascript
forum = Model(
    "forum",
    {
        "description": fields.String(description="Forum description"),
        "img": fields.String(description="Blob string of Image"),
        "name": fields.String(description="Forum Formatted Name"),
    },
    
    post = Model(
        "post",
        {
            "img": fields.String,
            "title": fields.String,
            "author": fields.String,
            "content": fields.String,
        }
    ),
    
    comment = Model(
        "comment",
        {
            "user_id": fields.String,
            "content": fields.String,
        }
    )
)
3.3.2. Flask REST API Server

As previously mentioned previously, the backend of this project uses Flask as the REST API server that handles incoming requests from the client side. In the Flask application several endpoints that can be called by client have been developed, and upon every call or URL request received, the backend will handle the task assigned to that particular endpoint accordingly. To better visualise and test the endpoints, a Swagger UI documentation has been integrated, allowing direct interactions with the endpoints (see Figure 18).

![Swagger UI](image)

**Figure 18** – Swagger UI application that can be accessed through a URL once the backend is deployed. The Swagger UI application will provide a list of API endpoints that can be accessed by the Flask application and also shows example request usage and its response.
The example of usage for each endpoints can also be found in the Swagger, including the requested input parameter(s) or payload and the example response (see Figure 19). We can also try out different input parameter(s) and check what the response is, making it easy for testing.

The endpoints that have been developed are mostly used to transfer data between frontend with Firebase and Fitbit. Some further processing and logic may have been implemented in some endpoints as necessary, so that the only the essential data are sent to frontend and database.
The complete list of endpoints available can be found below, with endpoints grouped by their namespaces:

1. **General**

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/images</td>
<td>Upload a new image to Firebase</td>
</tr>
</tbody>
</table>

2. **User**

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/users</td>
<td>Get list of all user IDs</td>
</tr>
<tr>
<td>POST</td>
<td>/users</td>
<td>Create a new user</td>
</tr>
<tr>
<td>POST</td>
<td>/users/&lt;uid&gt;</td>
<td>Update user data</td>
</tr>
<tr>
<td>GET</td>
<td>/users/&lt;uid&gt;</td>
<td>Get profile for user ID</td>
</tr>
<tr>
<td>POST</td>
<td>/users/&lt;uid&gt;</td>
<td>Update heart rate for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/&lt;uid&gt;</td>
<td>Get current steps for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/&lt;uid&gt;</td>
<td>Get current heart rate for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/&lt;uid&gt;</td>
<td>Get current minute burn for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/&lt;uid&gt;</td>
<td>Get current sleep data for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/&lt;uid&gt;</td>
<td>Get current activity data for user ID</td>
</tr>
<tr>
<td>DELETE</td>
<td>/users/&lt;uid&gt;</td>
<td>Delete user in Firebase by user ID</td>
</tr>
<tr>
<td>POST</td>
<td>/users/&lt;uid&gt;</td>
<td>Modify user in Firebase by user ID</td>
</tr>
</tbody>
</table>

3. **Risk**

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/risk/riskyearbyyear/&lt;uid&gt;</td>
<td>Get risk for user by year</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/riskyearly/&lt;uid&gt;</td>
<td>Get overall risk for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/riskbyyear/&lt;uid&gt;</td>
<td>Get risk for user by year</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/riskbydate/&lt;uid&gt;</td>
<td>Get risk for user by date</td>
</tr>
<tr>
<td>POST</td>
<td>/risk/riskbydate/&lt;uid&gt;</td>
<td>Post risk for user by date</td>
</tr>
</tbody>
</table>

4. **Community**

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/community</td>
<td>Get all forums</td>
</tr>
<tr>
<td>GET</td>
<td>/community/new</td>
<td>Post new forum</td>
</tr>
<tr>
<td>GET</td>
<td>/community/post/&lt;postID&gt;</td>
<td>Get specific post by post ID</td>
</tr>
<tr>
<td>DEL</td>
<td>/community/post/&lt;postID&gt;</td>
<td>Delete specific post</td>
</tr>
<tr>
<td>GET</td>
<td>/community/follow/&lt;username&gt;</td>
<td>Get specific forum by username</td>
</tr>
<tr>
<td>POST</td>
<td>/community/follow/&lt;username&gt;</td>
<td>Follow specific forum</td>
</tr>
<tr>
<td>POST</td>
<td>/community/unfollow/&lt;username&gt;</td>
<td>Unfollow specific forum</td>
</tr>
<tr>
<td>POST</td>
<td>/community/create/&lt;username&gt;</td>
<td>Create new post in the specified forum</td>
</tr>
</tbody>
</table>

5. **Fitbit**

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/fitbit/activities/&lt;uid&gt;</td>
<td>Get activity data for user ID</td>
</tr>
<tr>
<td>POST</td>
<td>/fitbit/sets/sets</td>
<td>Set activity data for user ID</td>
</tr>
<tr>
<td>POST</td>
<td>/fitbit/sets/sets</td>
<td>Set activity data for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/fitbit/weeklygoals/&lt;uid&gt;</td>
<td>Get weekly goals for user ID</td>
</tr>
<tr>
<td>POST</td>
<td>/fitbit/weeklygoals/&lt;uid&gt;</td>
<td>Set weekly goals for user ID</td>
</tr>
<tr>
<td>GET</td>
<td>/fitbit/weeklygoals/&lt;uid&gt;</td>
<td>Get weekly goals for user ID</td>
</tr>
</tbody>
</table>

---

28
3.4. Risk Prediction

The most important feature of this application is without a doubt the personal CHD risk prediction feature that is based on the algorithm developed by Dr. Youngwon Kim and his team. In this section, the whole process from collecting user data to the calculation of the risk prediction will be explained in more detail, including the implementation in the application.

3.4.1. User Data Collection

As previously stated in section 2.1.3, the risk prediction algorithm takes several input data from the user which can be grouped into two depending on the collection method: users’ health-related data and users’ physical activity data. The health-related data, as explained in section 2.1, is obtained from users through the self-input form that users have to manually fill in when creating an account. These data can also be edited any time in the application to reflect any changes in users’ personal health. The collection of users’ physical activity data, on the other hand, does not require users’ manual input. After users have signed up for a new account and submitted the self-input form, the application will prompt users’ to connect their Fitbit watch with the application. There will be an authentication process that needs to be done to integrate Fitbit account with the application, as explained in previously. Fitbit server will generate an access key and refresh token and send those to our backend server which will then store the data in the User collection in the database. For every fetch to the Fitbit server, our backend will use the access key, and then the physical activity data can be obtained for the risk prediction.

3.4.2. Risk Calculation

Once the setups for data collection have been completed and the application can get all the data required for the risk calculation, the application can then calculate users’ risk and visualise the risk trend when more risks have been calculated. In order to calculate the risk and display it on the application, the frontend of the application will have to call the backend API for risk calculation to trigger the backend to start calculating
the risk value. The backend will first fetch all the necessary input from Firestore and Fitbit web API, to get users’ personal health-related data and physical activity data, respectively. Each data value will then be assigned to the coefficients used for the algorithm, as earlier explained in section 2.1.3. The physical activity data fetched from Fitbit are not directly used in the calculation since they are in the form of total activity calories and total activity time. To obtain the physical activity level or ENMO, in the unit of milli-g, for the risk calculation, the data fetched from Fitbit will have to be converted into an ENMO value. To ensure that the physical activity level can represent the daily activity of the users, instead of just the data of one day, the backend server will also fetch the total activity calories and total activity time of the previous 7 days from the request date. The risk prediction will then be calculated using the 7-day average of the physical activity data obtained from Fitbit. By using the average of the data, we want to avoid any sudden fluctuation in the daily risk values that may be caused by users’ changing physical activities from day to day, so that we can provide a risk value that can better reflect their daily lifestyle.

The conversion from total activity calories and total active minutes in a day into ENMO is done through several steps:

1. Calculate users’ BMR

   BMR is the rate of energy expenditure in a day while being at complete rest [13]. This value is crucial for estimating the intensity of an activity that has been done on a particular day. The equation used by Fitbit to estimate users’ BMI value is a well-known equation called Mifflin – St Jeor [13].

2. Calculate MET value

   MET value is the ratio of the rate of a person’s energy expenditure, relative to that person’s mass, while performing some specific physical activity compared to a reference of an individual sitting quietly [14]. The formula to calculate MET value that is used in this project and was provided by Dr. Youngwon Kim is as follows:

   \[
   \text{METs} = \frac{\text{Calories burned}}{((\text{BMR} \times \text{Duration of activity in hours})) / 24)
   \]
3. Convert MET into ENMO (in J\(\text{min}^{-1}\text{kg}^{-1}\))

As mentioned in section 2.1.3, ENMO is a summary value of raw individual acceleration values throughout the day which can be used to estimate one’s movement on that day. According to a research paper written by White, et. al, ENMO can be estimated using a formula which was derived through the experiment of comparing subjects’ acceleration value to the MET value of the activity [16]. There are several estimation formulas mentioned and used for the experiments for the research paper, but only one were chosen to be used in this project. All of the formulas shown in the paper can provide relatively accurate estimations, but the simpler formula was chosen to avoid overloading our backend with heavy computation which may disrupt the smoothness of the application. The formula adopted in this project is as follows:

\[
\frac{5.01 + 1 \times \text{ENMO}}{71} = \text{MET (J\(\text{min}^{-1}\text{kg}^{-1}\))}
\]

Then after rearranging the equation, we get:

\[
\text{ENMO (J\(\text{min}^{-1}\text{kg}^{-1}\))} = (71 \times \text{MET}) - 5.01
\]

4. Convert ENMO into milli-g

The last step that needs to be done is to convert the unit of the estimated ENMO value, which is in J\(\text{min}^{-1}\text{kg}^{-1}\), to the requested unit for the algorithm, which is in milli-g. The conversion considers the standard gravitational unit, which is reflected in this formula:

\[
\text{ENMO (milli-g)} = \text{ENMO (J\(\text{min}^{-1}\text{kg}^{-1}\))} \times 0.1019716213
\]

Once the ENMO value has been attained, it would be assigned to the predefined coefficient in the algorithm. The risk is then calculated by summing up the multiplied input data by each of their respective coefficient and comparing them to the MeanX’B value of CHD risk probability data.

3.4.3. Storing Risk Prediction Results

Once the result of the risk prediction has been obtained, the backend server will send the result to the frontend as a response to the request
earlier, and the frontend will display it on the UI. At the same time, the backend server will also store the result in the Risk collection, together with the id of the user, parameters used, and the date of the calculation. The additional details stored together in the database are important to help users better understand the underlying factors that were considered for the risk calculation.

4. Difficulties and Limitations

This section will discuss the difficulties and limitations that have been faced throughout the development stage, including implementation of the current methodology and the development tools. Each difficulty or limitation will be elaborated in more details in subsections below along with the solutions used to mitigate them.

4.1. Fitbit Request Rate Limit

As mentioned earlier, this project needs to communicate with Fitbit to fetch crucial users’ data, be it for display or for risk calculation, making it heavily relies on Fitbit public web API. The initial plan of this project was to fetch Fitbit data constantly so that the application can display users’ physical activity data and risk prediction in real-time. However, during the implementation, our team discovered that Fitbit has set an upper limit to the number of possible API calls to be made under one hour. This limit has been set to avoid spam requests made to the Fitbit server which may overload the server cause harm. When the upper limit has been reached, Fitbit server will only send an error code of 429 as a response without returning any of the requested data. When the application receives the error code instead of the desired data, a bug occurs in the application since the application would not be able to display any of the Fitbit data on the UI nor would it be able to calculate the risk prediction. Therefore, one solution that is currently implemented in the application is to limit the API calls made to the Fitbit server to once every 10 minutes. This solution might not be the best since the application would not be able to always show the most up-to-date data, but it is sufficient for the purpose of this project, which is to allow users to better
understand how their CHD risk may dynamically change throughout the day and track their physical activities.

4.2. **Fitbit Manual Sync**

Another limitation that has been found when integrating Fitbit data into the application is the Fitbit data synchronization procedure. As shown in Figure 20, the Fitbit application apparently does not do automatic data synchronisation with the watch on the background but requires users to open the Fitbit application to allow syncing.

Upon syncing the data with the Fitbit application, the data will then be sent to Fitbit database to update the current data, which will then allow the web API to return up-to-date data from the Fitbit watch. This manual data syncing procedure may cause data inconsistencies between our application with the data from the Fitbit watch since users may not remember to always open the Fitbit application, and this may also cause inaccurate risk predictions. Since this limitation is mainly from the Fitbit server side, there is not much we can do on our side to fix or mitigate.

4.3. **React Native Performance**

As previously explained, the frontend of the application is developed using React Native because of the cross-platform benefits. However, as mentioned in that earlier, one disadvantage of using React Native is its lower performance...
when compared to other Native development tools such as Swift for iOS and Kotlin for Android. According to a benchmark test performed by inVerita [16], React Native applications tend to use higher CPU power as compared to its native counterparts, which leads to bigger battery consumption, almost two times on average. Moreover, React Native also has weaker computational power and efficiency, meaning the performance may be slower when doing large calculations and animations. To prevent any performance problem from happening in our application, we have avoided implementing any heavy animations and computations to be done in the frontend side, and no performance problem has been encountered so far.

4.4. Unavailable Public DTC API
The initial design of the application includes a page where users can sync their DTC test result and view them on the Genetics page. The DTC test provider that we intended to sync data from is a well-known provider called 23andMe. The initial plan was to give users the option to sync their genetic test results into the application as either an alternative or additional input to the risk prediction algorithm. However, upon further research to set up the connection between the provider’s database with our application, we learned that the API is not publicly accessible for everyone, and there are certain steps that need to be done before obtaining the access to their API. Developers would have to submit an application for the access and also explain about the application they are developing. Moreover, there was a certain deadline for applying for the access, and the deadline had passed before our team learned about this. Consequently, the application cannot sync with users’ DTC test result and the Genetics page feature is removed from the application. Upon consulting with Dr. Youngwon Kim and his team, we have decided that it would be best to not pursue this future for now since the risk prediction can still be calculated without the genetics data and that this feature is optional.
5. Contributions

As this project is a group project done by four individuals, this sections serves as a deeper elaboration on the author’s individual contributions to the project. The contributions that have been done(?) will be grouped into two: Frontend and Backend.

5.1. Frontend

5.1.1. Self-input Form

As explained in previously, users are required to complete a self-input form about their personal and health-related data, which will be the key data for the risk calculation. The whole design of the page, its components, and the processing of inputted data before sending them to the backend were developed by me. To ensure data type consistency with the model that has been set for the database, I have set up some data checking or validation before the form can be submitted. In the examples provided in Figure 21, it can be seen that the validation error messages may differ between fields depending on the validation rules that has been set. If any of the field is still violating the validation rule, the submit button will stay disabled to prevent users from submitting wrong data.

![Figure 21](image1.png)

*Figure 21 – Some validation examples that I have worked on for the input fields of the forms, for the Diet component, since it is a selection field, one element must be chosen. For the height field, a number must be inputted. For the name field it must be in the format of string and cannot be left empty.*

Some data are also calculated by the system automatically so that data accuracy can be achieved. For example, in Figure 22 below, the age is calculated from users’ inputted date of birth, and the BMI value is calculated from users’ inputted height and weight.

![Figure 22](image2.png)

*Figure 22 – Automatic calculations of some of the input data in the field. Age is calculated automatically once users input their date of birth, while BMI is calculated automatically from height and weight of users input.*
Additionally, to make sure that users know how to fill in some of the fields that may be slightly confusing, I have created a helper component, which is the blue ‘?’ pressable icon beside some of the fields. On the event of an icon press, a pop up message will be shown on the screen to give more information about the field to be filled (see example in Figure 23).

Figure 23 – helper components that I have developed, which will show when the user clicks the blue question mark button.

5.1.2. Home Page

My next major individual contribution is the development of the Home page. As mentioned previously, the Home page shows all the physical activity-related data from Fitbit. Each of the data, namely active calories, distance, activity, steps, and calories (total) is rendered in a simple boxed container with the data value as the centre point. As can be seen from Figure 24, I have created a reusable component called DataCard that accepts the title of the card or data name, value(s), the unit used, any additional notes, and the desired width of the component as parameters. This component is powerful since it is customizable and versatile, saving development time. I also implemented the solution mentioned previously, which is to prevent Fitbit data fetch error caused by exceeding the Fitbit API call limit. With this solution, the application is less likely to encounter a failed fetch which may cause missing data in the interface. Although the application cannot make another fetch until 10 minutes have passed since the last one, in the case of empty physical activity data in the Home page,
the application will make an exception and fetch the data from Fitbit without regarding the last fetched data.

An other major reusable component that can be found in the Home page is the bar graph that helps visualise users’ weekly steps. I designed the component to be able to accept any data, provided the x and y labels (e.g. for weekly steps, the x is the days in the week, and the y is the number of steps in that day). To also help users better understand how their weekly steps are plotted on the bar graph, each of the bars can be pressed and the number of steps represented by the pressed bar will be shown on the top of the bar and the bar colour will change to bright orange to highlight the press event (see Figure 25). Similar as the other physical activity data, the weekly steps data is also only fetched from Fitbit if the Home page does not store any weekly steps data or when 10 minutes have passed since the last fetch.

Figure 249: DataCard component. The leftmost frame shows that the component can accept more than one value and unit. The middle frame shows that the width of the component can be adjusted as desired. The leftmost frame shows the additional note that can be included in the component.

Another major reusable component that can be found in the Home page is the bar graph that helps visualise users’ weekly steps. I designed the component to be able to accept any data, provided the x and y labels (e.g. for weekly steps, the x is the days in the week, and the y is the number of steps in that day). To also help users better understand how their weekly steps are plotted on the bar graph, each of the bars can be pressed and the number of steps represented by the pressed bar will be shown on the top of the bar and the bar colour will change to bright orange to highlight the press event (see Figure 25). Similar as the other physical activity data, the weekly steps data is also only fetched from Fitbit if the Home page does not store any weekly steps data or when 10 minutes have passed since the last fetch.

Figure 25 – Bar graph component that I have developed, used to help user visualize their weekly steps activity. The left frame shows the initial state of the bar graph which is filled with user steps data from Fitbit. The right frame shows the event that happens when the one of the bars is clicked, it will turn orange and shows the respective number of steps on that day.
5.1.3. Profile Page

Another contribution that is worth mentioning is the Profile page development. This page holds all user details that have been previously inputted upon account creation. The details are divided into two, personal details and health details, and rendered in two separate components, as shown in Figure 26 below. The component that holds the details is a reusable component I developed, called DetailsCard. The component accepts the title of the card, the data to be shown on the card, and the optional button that helps navigate to another page. In the Profile page, the button being passed down to the DetailsCard component is another reusable component I created called NavigateButton, which will trigger navigation while also passing some data to another page. The NavigateButton in this page is used to navigate to the Edit page of the respective data (i.e. edit personal details or edit health details), and the data being passed to the Edit page is the current data fetched from DB upon rendering the Profile page. By passing data from one page to another page, the button helps remove the need to do another data fetch from DB in the target page, which in this case is the Edit page.

![Figure 26 – DetailsCard component showed in user profile screen. This component will fetch data from the backend and be filled with user information that they have previously inputted. Edit button is also provided to route users to the editing screen. 11](image-url)
The input fields in the Edit page behave similarly to the ones in the Self-input Form, which have been explained in section 5.1.1. Every field has a validation rule in accordance with the schema model in DB, and the form cannot be submitted unless all of the validation rules have been followed. The only difference between the form in Edit page and the Self-input form is that in Edit page, all of the fields are prefilled with the data fetched from DB while Self-input form is an empty form.

5.1.4. Risk Page

The last major contribution I have made in this project is the Risk page design, overall flow, and data fetch from DB. One of main goal of this project is to be able to show users how their CHD risk can dynamically change as they do physical activities. In order to help users track the trend of their CHD risk over some period of time, I have developed a line chart component. In the Risk page, the line chart is used to show the one year trend and one month trend of users’ CHD risk (see Figure 27). Since the line chart component is reusable, the two trends can share the same component but show data according to users’ request. By doing this, the style of the chart can be consistent and the development time was also short. The component only needs two parameters, which are the data to be shown and the x domain (e.g. for the one year trend the x domain is a list of all the months in one year, while for the one month trend the x domain is a list of all the days in one month)

![Figure 27 – Line chart components that I have developed, this is useful for visualizing the users risk trend throughout the month (left) or year (right). The data will be fetched from the backend according to the user.](image-url)
Another component I developed is the risk summary component (see Figure 28). I also implemented the Refresh button that has been explained in section 3.1, which similar to the physical activity data fetch in the Home page, prevent the application from calling the Fitbit API more than the set limit.

Lastly, I also implemented the All Risk page, a page that can be accessed through the Risk page. As explained in earlier, each of the risk in the All Risk page can be pressed, and it will navigate to the Risk Detail page that holds the params used to calculate the risk on that particular date (see Figure 29). I implemented this feature to help users to better understand the factors that need to be considered to maintain their CHD risk, allowing them to be more aware of their lifestyle and daily activities.

Figure 28 – Risk Summary Component that I have developed. It is useful to tell user the risk prediction value of today, yesterday, and previous month. A refresh button is also developed which will be disabled once a risk prediction is done and re-enabled after 10 minutes to avoid user spamming.

Figure 29- Risk Detail page. Upon pressing the “Show all data” button shown in the leftmost frame, the page will navigate to All Risk page where users can view all of the risk. If one of the risk is pressed, the page will navigate to Risk Detail page.
6. Future work

6.1. Expand Compatibility with Wearables

Given the limited time frame for this project, this application can only connect with Fitbit as of now. Although Fitbit is one of the most owned wearables in the world, the fact that some users might have wearables from other brands cannot be overlooked. Since this project aims to raise awareness of CHD risk and promote how physical activities can lower it, we want this application to be more inclusive and be able to reach more users. If this application is to be further developed, this application should be made more compatible with other wearable brands, such as Garmin and Apple Watch.

6.2. Implement Fitbit Accelerometer

As mentioned previously, in order to calculate the risk prediction, physical activity data is needed in the form of ENMO in milli-g. However, as explained in previously, ENMO value can only be retrieved from raw accelerometer data, and one way of fetching users’ raw accelerometer data is through Fitbit, which is an alternative we could not explore in this project since it requires developing a Fitbit in-watch application. Although our current formula of converting calories to ENMO value can still be used for the risk calculation, fetching raw accelerometer data directly from Fitbit may provide a more accurate result since our current formula also only serves as an estimation. Therefore, if given more time, an in-watch application can be developed and the accelerometer Device API can be explored. Upon the connecting their Fitbit account to this application, users can be suggested to also download the in-watch application in their Fitbit watch for a more accurate risk prediction. This additional download can be made mandatory or optional, since not all Fitbit version can be downloaded in-watch applications, and the current risk calculation using the conversion formula from calories to ENMO can still help with the risk prediction.

6.3. DTC Report

In the current design of the application, the Genetics page that should contain the result of users’ DTC report, according to the initial design, is not included
because of the limitation on the availability of the public API. As explained in earlier, our team were unaware of the fact that there is a deadline to apply for access to the public API of 23andMe and could not apply until next year. Hence, this application cannot include the page for showing users’ DTC test report as of now. In the future, the Genetics page can be implemented so that users can keep track of their personal health data in one application, and any important genetics data may also be considered in the risk calculation, leading to a more accurate result. Other options of credible DTC test provider beside 23andMe may also be explored and used.

6.4. Additional Features for Admin Page

Since our current project does not require any other functionality for the admin page, the current admin page only serves one function which is to help the admin manage the community page, including forums and articles. Aside from this single function, the admin page can actually be a more powerful tool for the admin to control the application. One example is allowing the admin to manage the users, which means the admin have the power to delete or disable a user if the user violates the general online community policies. Since currently, the Community page in this application does not have any guidelines regarding what comments users can and cannot post, users can freely post any kinds of comment without any limitation, which may lead to harmful problems, such as spreading false or misleading information and malicious comments that may affect other users. Therefore, it would be important for the admin to be able to disable or delete a user if any of these unwanted events happen. This also means that the admin can delete inappropriate comments from the articles as necessary.
7. Conclusion

CHD is a fatal disease that will continue to take move lives not only in Hong Kong, but also worldwide if people are not starting to become more aware of their risks and the danger that comes with it. Educating the public regarding this disease and how to mitigate it is therefore essential. One solution is by utilizing the rapidly advancing trend in digital health technology. Nowadays, many digital health devices, such as fitness trackers or wearables, allow users to easily track and gain insights of their own health, which then also promote the importance of staying healthy. Therefore, by benefiting from the advancing digital health technology we can find today, this project aims to raise awareness of CHD risk and help more people to be able to reduce their CHD risks by developing a mobile application that allows users to track their CHD risk and daily physical activities from their connected wearable. Through this application, users can better understand the underlying factors of their CHD risk and how the risk may dynamically change depending on their physical activities, thus encouraging users to live a healthier life.

The main focus of this project is to develop a cross-platform mobile application that allow users to see the trend of their dynamic CHD risk in the next 10 years. This is achieved by providing data visualisations that may also help users learn about how physical activities may reduce the tendency of developing CHD. Since this project is a collaboration project with the School of Public Health at the University of Hong Kong, the focus of our team’s work can be directed to the development of the mobile application, while the risk prediction algorithm is provided by Dr. Youngwon Kim and his team. The risk prediction algorithm takes two inputs: personal health-related data (e.g., age, BMI, and gender) and physical activity data obtained from users’ wearable device. The algorithm given is in a form of equation which our team has translated into code and implemented in the application.

In conclusion, this project has been successfully executed, and the main goal has also been achieved, which is to develop a mobile application that can provide personalised CHD risk. We were also able to include an additional Community feature in the application to boost user engagement with the application. React Native is used to develop the frontend of the application because with React Native more target users can...
be reached (both iOS and Android phone users), less development time was needed, and the consistencies in design across platform can be achieved. The backend of the application is developed using Flask, Firebase, and Firestore. Backend is also responsible for performing the risk calculation by taking the required input from users and then storing the result in the database. The result will also be sent to the frontend to be visualised in the UI.

Although this project has encountered several difficulties and limitations during development, alternative approaches have been implemented to prevent and minimize limitations. There are also still some areas of improvements and additional features that can be implemented in the future. Alternative development tools can also be explored to increase the performance of the application to its maximum.
8. References


