Development of a Mobile Application for Coronary Heart Disease Risk Prediction

COMP4801 – INDIVIDUAL FINAL REPORT

Written By: Harney Vieri (3035663368)

Group Member:
Hadipranata Mellisa  (3035663019)
Chan Hay Yin  (3035686047)
Hom Long Hin  (3035692450)

Supervisor: Dr. Loretta Choi
Abstract

Over the past century, heart disease has always been one of the top three causes of death. This project will specifically focus on coronary heart disease (CHD), one of the leading causes of mortality in comparison with other heart conditions. A government-sponsored initiative, HealthyHK, shows that on average 10.5 died from CHD every day in 2020. Mobile applications are now becoming more and more integrated into our daily lives with the advancement of technology. In particular, the market for health apps has been growing rapidly, especially those that allow users to track their own health data, otherwise known as mHealth technology.

As such, the motivation behind this project becomes clear. It is to raise awareness of the danger of CHD, one of the most fatal causes of death, and at the same time emphasize the importance of physical activity to reduce the risk of CHD. The aim of the project is therefore to utilize the trend mHealth technology. By the end of this project, a well-integrated smartphone application with a wearable health device that can predict an individual risk of developing CHD using estimation from their genetic information as well as physical activity data collected from the wearable device is expected to be produced. With this project, self-assessment of CHD is made possible, bearing in mind that there are currently no publicly available tools that can be used to assess CHD risk without the help of medical professionals.

This project is made possible with the collaboration with the School of Public Health at The University of Hong Kong. They will be overseeing the genomics portion of the project, including the development of biological equation that will be used to predict an individual risk of developing CHD. Our team, on the other hand, will oversee the technical side of the project, translating the risk prediction algorithm and integrating it into the mobile application in general. This includes, but not limited to, collection of required input and calculation of the risk inside the application.

This report will go over the complete process of the project in detail, all the way from the start to its current finished state. To be more specific, the methodology that has been implemented will be discussed, starting from the design phase, implementation, and finally, result. The engineering choices and justification will also be explained in detail, this includes the benefits and limitations
found in the implemented methodology and alternatives that has been explored. Justification behind the logic of the implementation will also be given throughout the report. This is done to ensure that the methodology used throughout the project is robust and able to ultimately achieve the aim of this project, which is to provide effective yet accurate estimation of one’s prediction of developing CHD.

Although we have achieved our goal in creating this application given the time span of this project, we believe there is always room for future work. There are still certain areas for improvement that we can suggest if the project is to be picked up in the future. Some other methodology can be tried and benchmarked to ensure that the most accurate result of CHD risk prediction is obtained. Some of the limitations that will be discussed in the project could also be overcome easily given additional time to work on this project which might be done by using a more complex and robust development tools, even though they are more time consuming. This can ensure maximum efficiency and accuracy of the mobile application. Furthermore, some additional features can also be implemented in the future aside from the main feature of CHD risk prediction to allow better user interactivity, ultimately making the application more engaging.
Acknowledgement

We would also like to firstly express our deepest gratitude to Dr. Loretta Choi for her continuous support and guidance throughout this project. Without her advice, expertise, and help, this project would not have been able to proceed as smoothly.

Finally, we would also like to give our most sincere appreciation to Dr. Youngwon Kim and his team from the School of Public Health at The University of Hong Kong. Dr. Youngwon Kim has entrusted us to collaborate on this interesting project and helped us in many biological aspects of this project. With his consultation and suggestion, we were able to make many calculated and reasonable decisions quickly on the methodology used for the project.
# Table of Contents

List of Figures ................................................................................................................................. 6  
List of Abbreviations .......................................................................................................................... 7  
1. Introduction ................................................................................................................................... 8  
2. Methodology .................................................................................................................................. 11  
   2.1 Platform and Equipment Set-up ................................................................................................. 11  
      2.1.1 Fitbit and Fitbit Web API .................................................................................................... 11  
      2.1.2 Self-input Data and Physical Activity Data .......................................................................... 12  
      2.1.3 Prediction Algorithm .......................................................................................................... 13  
   2.2 System Architecture Design ...................................................................................................... 14  
      2.2.1 Fitbit Web API ..................................................................................................................... 14  
      2.2.1.1 Fitbit Web API ............................................................................................................... 15  
      2.2.1.2 Fitbit Web API ............................................................................................................... 15  
      2.2.2 Application Frontend .......................................................................................................... 17  
      2.2.3 Admin Frontend .................................................................................................................. 18  
      2.2.4 Backend and Database ....................................................................................................... 19  
3. Results .......................................................................................................................................... 20  
   3.1 Design and Implementation ....................................................................................................... 20  
      3.1.1 Application Frontend .......................................................................................................... 20  
      3.1.2 Admin Frontend .................................................................................................................. 24  
      3.1.3 Backend and Database ....................................................................................................... 27  
      3.1.3.1 Firebase and Firestore .................................................................................................. 27  
      3.1.3.2 Flask REST API Server ................................................................................................... 28  
      3.1.4 Risk Prediction .................................................................................................................... 31  
      3.1.4.1 Data Collection .............................................................................................................. 31  
      3.1.4.2 Risk Collection ............................................................................................................... 31  
      3.1.4.3 Storing Risk Prediction Results ...................................................................................... 34  
   3.2 Limitation and Difficulties Encountered .................................................................................... 34  
      3.2.1 Fitbit Request Rate Limit .................................................................................................... 34  
      3.2.2 Fitbit Manual Sync ............................................................................................................. 35  
      3.2.3 React Native Performance ................................................................................................... 35  
      3.2.4 Unavailable DTC API ......................................................................................................... 36  
4. Contribution .................................................................................................................................... 37  
   4.1 Application Frontend .................................................................................................................. 37  
      4.1.1 Community Page ................................................................................................................ 37  
      4.1.2 Authentication Process ....................................................................................................... 39  
   4.2 Admin Frontend .......................................................................................................................... 39  
   4.3 Backend .................................................................................................................................... 40  
      4.3.1 Risk API ............................................................................................................................... 40  
      4.3.2 Fitbit API .............................................................................................................................. 41  
      4.3.3 Community Page API ......................................................................................................... 41
5. Future Works ..........................................................42
  5.1 Integrating Other Brands of Wearables ..............................................42
  5.2 DTC Report ..................................................................................42
  5.3 Accelerometer to Collect Physical Activity Data .................................43
  5.4 Additional Features for Admin Page ................................................44

6. Conclusion ..........................................................................................45

7. References ............................................................................................47
List of Figures

Figure 1 – CHD illustration ........................................................................................................8
Figure 2 – mHealth Technology ..............................................................................................9
Figure 3 – Fitbit web API parameters ..................................................................................12
Figure 4 – System Architecture Design .............................................................................14
Figure 5 – Fitbit OAuth 2.0 authorization URL format .......................................................15
Figure 6 – POST request to Fitbit server to get access tokens, ........................................15
Figure 7 – Expo app running on two different devices. ....................................................18
Figure 8 – Initial UI design and Final UI of the application ..............................................20
Figure 9 – Account creation flow of the application ........................................................................21
Figure 10 – User information edit flow of the application ..................................................22
Figure 11 – Risk screen flow of the application .....................................................................23
Figure 12 – Community screen flow of the application .......................................................24
Figure 13 – Admin page UI. ....................................................................................................25
Figure 14 – Admin page forum creation pop-up. .................................................................25
Figure 15 – Admin page article creation pop-up. .................................................................26
Figure 16 – Admin page article preview pop-up. .................................................................26
Figure 17 – Google Firebase console and its parameters ....................................................27
Figure 18 – User schema ........................................................................................................27
Figure 19 – Community feature schema .............................................................................28
Figure 20 – Swagger UI. ........................................................................................................28
Figure 21 – Example of API endpoint call in Swagger UI ....................................................29
Figure 22 – Example response in Swagger UI .....................................................................29
Figure 23 – Fitbit data not syncing. ......................................................................................35
Figure 24 – Community card components .........................................................................38
Figure 25 – Comment component. ......................................................................................38
Figure 26 – Post summary card component for admin page .............................................40
**List of 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

There are many different types of heart diseases, many of which has been one of the leading causes of death in Hong Kong for the past century. The major contributor to death caused by heart disease is from a type known as coronary heart disease (CHD). Coronary heart disease is a type of cardiovascular disease that occurs due to the narrowing of blood vessels caused by build-up of fat, cholesterol, and other materials inside the arteries, illustration shown in Figure 1 below. This narrowing in turn causes supply of oxygen and nutrients into the heart to be blocked, which can be fatal in the long-run or in extreme cases [1]. HealthyHK, a government organization, conducted a study in 2020 and showed that a daily average of 10.5 persons died from CHD [2]. Thus, it is undeniable to say that prevention of CHD has not been successful or meaningful enough in recent years. Hence, more ways are needed to help prevent it and bring awareness to people of its danger.

![Figure 1 - An illustration of CHD. Atheroma (fats, cholesterol, and other materials) get build up over time inside the heart arteries, blocking supply of oxygen and nutrients into the heart. Cases gets more severe on the illustration towards the right. [1]](image)

It is also indisputable to say that the advancement of technology has become increasingly rapid over the past years, which includes the mobile application and health wearables industry [3]. The health app market particularly has been growing gradually, especially mHealth apps, which are mobile applications that allow users to track their personal health data. To justify this claim, a recent study conducted by Harvard Business Review has suggested that the adoption of mHealth apps can bring various benefits to its users, improving both their short-term and long-term health lifestyle [4]. Some of the most notable examples include immediate risk reduction when tracking specific chronic disease and reduction in medical expense in the long run.
Furthermore, as of right now, there are no readily available tools to help assess one’s risk of developing CHD without the help of medical professionals. The most accessible method to predict CHD currently is the use of a risk calculator website that mainly uses traditional risk markers such as blood pressure, cholesterol level, etc. One example of such a website can be found through this link [https://cvdcalculator.com/](https://cvdcalculator.com/). These websites are mainly administered purely based on traditional risk markers as the basis of its prediction and can be relatively less accurate as it does not consider other factors such as an individual’s unique genetic make-up and physical activity level. In addition, most of these websites still require some data collected by medical professionals as its input field. Hence, it is justifiable to say that the existing and most accessible methodology of predicting CHD is still not effective enough.

As such, bearing in mind the statistics mentioned earlier and the research gap, the motivation and importance of this project becomes much clearer. That is, to try and utilize the advancing trends and well-proven method of mHealth technology to try and improve awareness and ultimately reduce the risks of CHD. The aim of this project is therefore to create a well-integrated mobile application with a wearable health device to help prediction and visualization of one’s risk of developing CHD become more accessible. Aside from that, this project also aims to stress the importance of physical activity for user’s overall health, and in-turn shows in real-time how it can help mitigate one’s risk of developing CHD. With the dramatic advancement in wearables and mobile application technology, self-assessment of CHD can be made possible. Hence, this application hopes to use the advancement of mHealth technology to make self-assessment possible and provide a more accessible yet accurate method for laymen to easily obtain and understand their own risk of developing CHD.
In addition, this project is also a collaboration with the School of Public Health of The University of Hong Kong, who will be mainly in charge of the genomics part of the project, which mainly involves the development of biological equations to predict one’s risk of CHD. Our team, on the other hand, will be taking care of the technological aspects of the project. Hence, this report will discuss in detail the methodology that has been used and chosen throughout the project in order to achieve the goal. To be more specific, this report will first discuss the developmental tools and methodology implemented in Section 2, which includes the platform and equipment setup in Section 2.1; and system architecture design in Section 2.2. Section 3 of the report will then discuss the results of the project, the implementation in Section 3.1 starting from the design phase to the implementation phase; the limitations and weaknesses that have been identified on the current methodology will then be discussed in Section 3.2. The contribution that has been made by me to the project along with the general division of task will then be discussed in Section 4. All in all, the project has been completed successfully and we are satisfied with our end-product.
2. Methodology

This section of the report will discuss in detail the methodology that have been incorporated into the project including alternatives that have been explored if any. This includes the initial set-up that has been implemented; system design; and initial UI design. The developmental tools that have been chosen to implement each part of the application will also be discussed with the justification behind each engineering choices.

2.1 Platform and Equipment Set-up

First, there are several equipment and development platforms that need to be set-up before starting the implementation phase. The initial set-up requirements that have been done will be explored in more detail in the sub-sections below.

2.1.1 Fitbit and Fitbit Web API

One of the most important data that needs to be collected in this project is the physical activity data of the users. To achieve this, a wearable device is required to track the daily movements, physical activity, and other health related data of the users. We have decided to use a smart watch from Fitbit to collect these data. There are several well-known brands of smart watches that can be achieved to achieve the project’s goal. In the end, we decided to use Fitbit because of several benefits of using Fitbit as compared to other brands. One of the major benefits of Fitbit is its affordability as compared to other smartwatches while still being compatible with most if not all popular smartphones today. Furthermore, with this benefit, Fitbit is therefore one of the most used and trusted health wearable devices in the world. Research conducted by Statista shows that the number of active Fitbit users is constantly growing yearly, with almost 31 million active users in the year 2020 alone [5]. Integrating Fitbit into the application means that all users who already own a Fitbit watch can immediately start using our application, and considering its large database, this will indirectly add credibility into our application.
Fitbit also provides a web application programming interface (API) which is extremely useful for our project. Its API is robust and has all the features needed to extract the data needed for the risk prediction. Once users finished creating their Fitbit account, the data collected from their watch will be able to be extracted easily from the web API after they have allowed our application to extract data using access token given by the users. To set-up the integration with our mobile application, a Fitbit developer account first needs to be created and the mobile application will then need to be configured according to the client id and client secret as shown in Figure 3 below.

![Figure 3 – Fitbit web API information that needs to be integrated with the application, in this project’s case, the backend server. Users must first create a developer account before being able to access this information.](image)

2.1.2 Self-input Data and Physical Activity Data

As mentioned earlier, user physical activity data is one of the most important data that needs to be collected for this project to be successful. Another important data that also needs to be collected for the risk prediction equation is personal data, which includes data such as, age; sex; gender; etc. The risk prediction equation will be explored in more detail in the next sub-section and will explain how these data will be used for the risk prediction calculation. For the genetics data, a form inside the application has been set-up. This form needs to be filled in at the very start once a user created an account for our application before they can fully utilize the application, i.e., accessing other screens. As for the physical activity data, they will be collected through Fitbit by first prompting the user to connect their Fitbit account into our application after they finish finishing the genetics data form.
2.1.3 Prediction Algorithm

The School of Public Health of The University of Hong Kong, led by Dr. Youngwon Kim and his team are the ones in charge of the genomics aspect of the project as mentioned earlier in the introduction. This project did not focus on the development of the algorithm, but rather on its integration and the translation from its biological and mathematical form into codes that run on our backend server, the implementation will be explained in more detail on Section 3. The algorithm that Dr. Youngwon and his team developed is a lifestyle-based model (LBM) equation which calculate one’s risk of developing CHD by assigning coefficient to each lifestyle indicator such as age; gender; BMI; physical activity level; smoking status; alcohol consumption; and medication use of blood pressure, cholesterol, and insulin. The total coefficients will then be used to estimate the 10-year risk of CHD of the user by estimating it relative to the MeanX’B value, which is an estimation derived from the data source of other individuals’ data and their CHD risk status. The coefficient of the algorithm is estimated using data obtained from UK Biobank as its primary data source, as its robustness allows for accurate prediction of each individual’s unique genetic risk for CHD, estimated based on the recently published list of 240 Single Nucleotide Polymorphisms (SNPs). The algorithm is still yet to be published publicly and hence still has some level of confidentiality to it and cannot be fully disclosed yet in this report.
2.2 System Architecture Design

This section of the report will discuss the system design that have been designed and used as a guidance throughout the project, which can be seen from Figure 4 below. The system design is useful to show how each major component interacts with the other components and how they can be integrated into one fully functioning application. Each component and their roles will also be explained in further detail along with the development tools chosen to develop them and the reason behind their choices.

![Figure 4 - The system architecture design that illustrates the flow of communication between different parties with the application. a) The mobile application comprises of two services, frontend and backend which will handle communications with other parties. b) The client or user provide their genetic data to the application through self-input, according to their knowledge, or according to their DTC genetic testing results. c) The Fitbit API will be called by the application for user authentication and by the backend to get user’s physical activity data. d) The Firebase is installed in the backend service and connected to the database of the application, where all user’s data will be stored. e) An admin page client is also developed to manage community feature of application.](image)

2.2.1 Fitbit Web API

As mentioned earlier, users’ physical activity data will be collected through the Fitbit wearable watch which will then be extracted through the web API provided by Fitbit once the oAuth2 authentication process is completed. The oAuth2 process follow the Authorization Code Grant Flow with Proof Key for Code Exchange (PKCE) to ensure the security of user data [6]. As shown in the system design, the flow of oAuth2 that is used for this project is on the server side. The server side will generate a random code verifier and code challenge encrypted in SHA-256 method; the client will then request an authorization URL embedded with the code challenge created earlier as shown in Figure 5.
The client side will then access this URL and communicate with Fitbit server directly and receive an authorization code as a callback if the sign-in is successful. The authorization code will then be sent back to the server side which will then be exchanged with an access token and refresh token through a POST method to the URL below.

Once the access token and refresh token are obtained, they will be stored by the server to the Firebase database according to the user. These tokens will then be able to be used in the future to be exchanged with requested user physical activity data according to Fitbit documentation.

### 2.2.1.1 Calories Burned and Active Time

As mentioned earlier in Section 2.1.3, the equation requires physical activity data in the unit of milli-g. One way to obtain this is by estimating it through converting the calories burned in a day and comparing it to activity times and users’ basal metabolic rate (BMR). Further explanation and the actual implementation of this in the project will be discussed further in Section 3.1.4.2. Hence, Fitbit web API plays a key role in this project to provide these data.

### 2.2.1.2 Alternative Tried - Accelerometer

Another method to estimate users’ physical activity is by estimating users’ acceleration in movement, which can be done using accelerometer sensors available from either the wearable watch worn by users, including Fitbit devices. This method has initially been tried and explored as the main method to estimate users’ physical
activity level in milli-g. The way physical activity can be estimated through accelerometer sensor is to calculate the Euclidian Norm Minus One (ENMO) value for the user throughout the day. ENMO is a summary score of raw acceleration levels, which is a vector magnitude of acceleration in the X-Y-Z axes minus one gravitational unit (1g). The value was subtracted by 1 to take into account local gravitational unit of 1g when the device is still [7]. Therefore, the project initially tried to incorporate a separate in-watch application that constantly runs in background into this project to keep track of raw acceleration data in the three axes, which then can be converted into ENMO simply using the formula:

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

The sum of these values throughout the day can then be used directly into the risk prediction equation as the user physical activity level.

Asides from web API, Fitbit also provides some devices API that can be accessed by making an application that can run inside Fitbit watch [8]. However, upon further exploration, it was found that this might not be the most suitable method to be pursued for this project as this require us to build another separate application which can be time consuming considering the short time span of this project. Another problem that was encountered through this method is that many devices do not actually allow accelerometer sensor to be used in background, including Fitbit devices. Through research, it was found that the reason accelerometer is commonly not allowed to be run in background in most devices, or Fitbit in this project’s case, is to protect the device’s health as running this task continuously can damage the battery and slow down devices since accelerometer sensor typically consumes a lot of battery and processing power [9]. Hence, this method was no longer pursued as there would be no guarantee on its results and was later changed to the current methodology as mentioned in Section 2.2.1.1.
2.2.2 Frontend Application

The frontend or the client-side of the application has certain roles in this system. Firstly, as mentioned earlier in Section 2.1.2, the equation requires some genetics data from the user as part of the equation parameters. As can be seen from the system design, this will be done in the frontend and users will need to input their genetic information as soon as they register for an account for our application before being able to use the application fully. The frontend will then communicate with the backend server which in turn will communicate with the database to store this information. Furthermore, the client side will also communicate with the backend which will call the Fitbit API explained in section 2.2.1 to get the relevant data regarding users’ physical activity which will then be displayed in the summary screen of the application. More explanation regarding the backend of the application will be explained in subsection 2.2.3. Aside from that, the frontend has its general role of providing UI for the user to look at and shows relevant information including risk prediction, trend, and community pages. The UI will also provide relevant components for users to edit their information and calculate CHD risk at any point in time, which will then allow the frontend to communicate with the backend at users’ will behind the scenes.

The developmental tool that is used for the client-side is Expo Application, which is built on top of React Native. This tool is ideal for this project because of several reasons. First, it is a cross-platform development tool, which means that the benefits of cross-platform development can be taken advantage of. The application can be run and packaged into two of the most popular smartphone platforms (Android and iOS) using a single codebase, saving development time, giving a more efficient debugging process, and promoting consistency in design [10], examples can be seen in Figure 7. This also means that the application can gain as much exposure as possible having been able to cover two of the most popular smartphone operating systems in today’s day and age. Although cross-platform applications typically perform slower and is less efficient that native development tools such as Swift, Kotlin, etc., its performance is still relatively comparable for the purpose of this project. The frontend is also mainly used to visualize relevant data and most of the heavy calculations will be performed on the backend of the application. Hence, the
benefit of cross-platform development outweighs the disadvantage of it being slower in this project’s case.

2.2.3 Frontend Admin

Another component in the system is the frontend or the client-side of the admin page, which is a web-application. The purpose of admin page is still quite limited for this project, it is only used to manage the community feature in the application which is a side feature of the application. Upon completion of the project, the admin would be the Dr. Youngwon Kim and his team who will take over the project and implement any other future works. In the community page, there will be articles which users can read and comment on to interact with other users and the admin of the application. Hence, the admin page is only used for admin to delete and create application for this project. Therefore, the admin client is still deployed locally for this project and the application can only be run by the admin locally. The admin frontend will then communicate with the backend server and store articles posted or delete articles in database according to the request on the frontend admin side.

The developmental tool that is used for the admin frontend is React.js, the reason React.js is used for developing this component is to have some kind of consistency for the code in the frontend of the application which is developed in React Native. Other benefits of using React.js to develop web-application also includes faster rendering time, ease of use, and a more stable code structure [11].
2.2.4 Backend and Database

The last major and important component of the system is the backend and database, which together act as an API server that responds to client requests as mentioned earlier in the previous sub-section. The application deals with a huge amount of data while also needing to perform some type of computation, i.e., the risk prediction of developing CHD. As such, having a separate server to help offload the client-side task makes sense for this project. Hence, the backend plays an important role in this system, which is to perform calculations of the risk to help avoid slowing down the client side, which is what user will be facing most of the time.

Another key role of the backend is to provide authentication method to Fitbit before being able to fetch relevant users’ physical data accordingly as explained in Section 2.2.1. The backend will also process the response from Fitbit that the client requested, processing it into format that contains only relevant information to reduce payload size to the client-side. The backend will in summary act as a REST API server that responds to on-demand request from the client.

Regarding the development tools used to develop the backend and database, **Flask and Firebase** will be used respectively. There are several advantages of using Firebase, which is a backend-as-a-service platform offered by Google. One of the major reasons is its extensive and detailed documentation to help developer easily use their application and help debugging be more efficient as a result. Furthermore, Firebase also offers many helpful features for mobile application development such as authentication feature, storage, etc. A NoSQL cloud database known as **Firestore** can also be used from Firebase. This means that the backend and database of the application is tightly integrated with each other which can help the performance of the server.
3 Results

This section will discuss the result of the project, i.e., what has been implemented and how each of the component in Section 2 is implemented. Other than that, limitation and problems that have been encountered in the current implementation will also be discussed.

3.1 Design and Implementation

The sub-sections below will dive deeper into the results that have been produced by the end of this project for each of the component discussed in Section 2 earlier. This includes every step starting from the design, set-up, and finally, the implementation phase.

3.1.1 Application Frontend

The initial UI design of the frontend of the application is shown in Figure 8a below, while the final implementation is shown in Figure 8b. As can be seen, the final implementation of the frontend does not differ much from the initial design asides from the Genetic page which has not been developed due to the limited availability of the DTC tests API as will be further explained later on in Section 3.2.4 of the report.

Figure 8a: The initial UI design of the application. The screens in the picture from left to right are as follow: Home, Home, Genetics, Risk, Community
Figure 8b: The final UI design of the application. The screens in the picture from left-to-right: Home, Home, Risk, Risk, Genetics
Furthermore, the overall design of the UI is kept simple while still providing all the necessary components to ensure its effectiveness in usability too. Many of the UI components are visualization tools such as line charts and bar graphs to help users understand their data more easily and be able to identify the trends and correlation between physical activity and the risk of developing CHD. As a result, this will help achieve the projects goal of raising awareness to user the importance of physical activity to reduce the risk of developing CHD.

As explained earlier in Section 2.1.2 and 2.1.3, users’ self-input data and physical activity data is required as part of the input to the risk prediction algorithm. Hence, users must first fill in this information upon signing in for the first time after they created their account. A form has been developed and UI component to authenticate and sync their Fitbit account into the application is provided as seen in Figure 9 once user logs in into the application if this has not been done. The frontend will then communicate with the backend to complete this whole process, which will be further explained in Section 3.1.3.
Once the authentication and data filling process is completed, users will be redirected to the Home screen, illustration shown in Figure 10 on the leftmost frame. The Home page has the necessary UI components such as card components to provide the activity summary of the day such as active calories; distance; activity time; steps; and calories burned, a bar graph component is also included to provide a summary of users’ weekly steps which can act as a good indicator of how active the user has been over the span of that specific week. All of these data is fetched upon request to the backend which will then communicate with Fitbit API to get the data for the corresponding user. However, Fitbit has a limit on the request rate, which can cause a bug inside the application if the quota is hit, elaborated more in Section 3.2.1. Hence, the request data is stored in the local storage of the device implemented using AsyncStorage library from Expo and this data would be use temporarily for re-rendering before being able to make the next request in the next 10 minutes.

From the Home page, users can also access their profile by tapping their avatar on the top-right corner and update any data that they want accordingly as this data plays an important part in the risk prediction as well, the complete illustration can be seen from Figure 10 above.

Figure 10 – The workflow of editing user information that they has initially inputted when creating an account. Starting from the Home screen on the leftmost screen, tapping their avatar to be redirected to the detail screen. Clicking the edit button to edit the information.
Moving on to the Risk screen, shown in Figure 11, which shows all the CHD risk-related data of the user. In the summary component inside the Risk screen, users can see their CHD risk value for today, yesterday and the previous month. The risk prediction calculation will be explored in more detail in Section 3.1.4. To summarize, when users enter the Risk screen, the frontend will send a request to the backend to get the risk value of today and fetches the required data from Fitbit API. To avoid the problem of having too many request to Fitbit web API again, a limitation on the risk prediction calculation amount is that it can only be calculated once every 10 minutes. Once the risk is calculated, the refresh button will get disabled and can only be pressed again in the next 10 minutes to request the risk data from the backend again.

As can be seen in Figure 11 above, line chart component is also provided in the Risk page plotted with the all-risk data of the users at a certain date to help user visualize the risk trend over a period of time, a monthly view and a yearly view of the trend is available. Users will also be able to see the whole history of the risk value by pressing the show all data button which will lead them to the list of risk result value and the date of the calculation. By clicking one of the dates, they would also be able to see the parameters of the risk on that specified date so they can understand what causes the risk value to be that specific amount on that day, the flow is again illustrated in Figure 11.
The last screen in the application is the Community screen, which is a side-feature of the application. As can be seen in Figure 12 below, users can see the list of forums created by the admin which will be explained further in the next sub-section. With the community feature, users will be able to read articles posted by the admin and also interact with other users and the admin by posting comments and reading other users’ comments at the same time.

![Image](Figure 12.png)

**Figure 12 – The flow of the Community screen, starting from a list of forums which users can click to go into the second frame showing the list of articles and clicking the articles will direct user to the article screen shown in frame three and four.**

### 3.1.2 Admin Frontend

As aforementioned in Section 2.2.3, the sole purpose of the admin page for this project is to manage the community feature of the application. Therefore, features that were implemented includes basic features such as creating and deleting forums and articles. As this is a side-feature of the application, not much can be completed given the limited time of this project. Hence, the project is still deployed locally and admin can run the project as per the instruction if they want to manage the community page.

The overall design of the admin page is right now is very simple and straightforward. As shown in Figure 13, it is only a one-page web application which contains a list of forums and the corresponding list of articles posted in that group.
Although the design is simple, it has all the required UI component to help the admin manage the community feature. Delete buttons are provided to delete the specific articles or forums, the frontend will then communicate with the backend to delete the related articles data in the database accordingly. To add a new forum, the admin can click the “New Group” button and a pop-up with fields input would need to be filled accordingly as can be seen from Figure 14. Once all the data is filled, the admin frontend will communicate with the backend and send the corresponding payload forum data to be stored in the database accordingly. Similar process is done for the “New Post” button to create new article in the available forums which can also be seen in Figure 15.
The admin can also preview the articles that they have posted by clicking on the article card UI component. The UI will send a request to the backend server to send the specific articles data which will then be previewed as seen in Figure 16.

The admin features can further be expanded in the future as it has potential to help manage the application and its users more efficiently. Some ideas will be suggested in the later part of the report in Section 5.4 of the report.
3.1.3 Backend and Database

For the backend of the application, it can generally be divided into two main implementations, Firebase and API development using Flask. The detail of each implementation will be discussed in the sub-sections below.

3.1.3.1 Firebase and Firestore

Most of the implementation of Firebase were mainly about the set-up and its integration into both the Flask application and the frontend application. As can be seen from Figure 17 below, we need to first initiate a Firebase application and then integrate the required parameters such as project id, client secret, etc. into our Flask application in order to use its features.

![Google firebase console of the project which shows the project id, number and web API key that needs to be configured into the Flask backend in order to use the database and other features.](image)

Schemas have also been set up to make sure that the data have some kind of structure that can be referred to as a guidance when developing the application. The schemas used are listed in Figure 18 and 19.

![The schema that have been designed for the user model which contains the users' info and health data along with the type of data that will be stored in the field.](image)
3.1.3.2 Flask REST API Server

For the backend of the application, as mentioned earlier in Section 2.2.4, a Flask application has been developed which will act as REST API server that handle client on-demand request. Therefore, endpoints will be developed inside the Flask application that the client can request too, and the backend will handle the task accordingly depending on the requested URL. To visualize its usage, a Swagger UI documentation is also integrated to allow direct interactions with the API endpoints and provide usage examples on these API endpoints, which can be seen from Figure 20 below.

![Figure 19 – The schema that has been designed for the community feature, it includes schema for the forum, post, and comment along with the required type to be stored for the corresponding field.](image)

![Figure 20 – The Swagger UI that can be accessed through a specified link URL (localhost:3000 if deployed locally) once the backend is deployed. The Swagger UI will provide a list of endpoints that can be accessed along with the required parameters and response.](image)
Using Swagger, the endpoints can be tested and while also giving clear specifications to the input parameter and gives example of the response of the requested at the same time. An example of a request is shown in Figure 21 below.

![Swagger UI](image)

**Figure 21** – Clicking one of the API endpoints, detail of the endpoints will be shown, including the required parameters and a call example to the URL to use the endpoint. The figure shows an example of calling the `/user/` endpoint which does not require any argument and input field.

Then, the response to the requested URL will be documented too by Swagger, illustrated in Figure 22 below.

![Swagger Response](image)

**Figure 22** – Continuing from Figure 21, a response example will also be shown according to the input for the API endpoint. In the case of `/user/` endpoint a list of users ID that is stored in the database is returned.

Most of the endpoints in the Flask server serves as an intermediary between Firebase and Fitbit web API, implementing additional processing and logic if required so that the data that is sent to the frontend can be made more simplified and effective.
The list of endpoints that are available from the Flask server grouped by their namespaces is as following:

1. **General**

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</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>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/users</td>
<td>Get list of all user ID</td>
</tr>
<tr>
<td>POST</td>
<td>/users</td>
<td>Create a new user</td>
</tr>
<tr>
<td>PUT</td>
<td>/users/health/({uid})</td>
<td>Modify health by User ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/health/({uid})</td>
<td>Get Health by User ID</td>
</tr>
<tr>
<td>PUT</td>
<td>/users/info/({uid})</td>
<td>Modify Info by User ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/info/({uid})</td>
<td>Get Info by User ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/name/({uid})</td>
<td>Get usernames by User ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/verification/({uid})</td>
<td>Get verification by User ID</td>
</tr>
<tr>
<td>GET</td>
<td>/users/({uid})</td>
<td>Get a user from Firestore by User ID</td>
</tr>
<tr>
<td>DELETE</td>
<td>/users/({uid})</td>
<td>Delete user in Firestore by User ID</td>
</tr>
<tr>
<td>POST</td>
<td>/users/({uid})</td>
<td>Modify user in Firestore by User ID</td>
</tr>
</tbody>
</table>

3. **Risk**

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/risk/monthly/({uid})</td>
<td>Get user's monthly risk by User ID</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/weekly/({uid})</td>
<td>Get user's weekly average risk</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/({year})/({uid})</td>
<td>Get user's yearly risk by given year</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/({uid})</td>
<td>Get user's risk by User ID</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/({uid})/({date})</td>
<td>Get user's risk by date</td>
</tr>
<tr>
<td>POST</td>
<td>/risk/({uid})/({date})</td>
<td>Post user's risk by date</td>
</tr>
</tbody>
</table>

4. **Community**

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/community</td>
<td>Get all forums</td>
</tr>
<tr>
<td>POST</td>
<td>/community</td>
<td>Post new forum</td>
</tr>
<tr>
<td>GET</td>
<td>/community/post/({postID})</td>
<td>Get specific post by post ID</td>
</tr>
<tr>
<td>DELETE</td>
<td>/community/post/({postID})</td>
<td>Delete specified post</td>
</tr>
<tr>
<td>POST</td>
<td>/community/post/({postID})</td>
<td>Create new comment under specified post</td>
</tr>
<tr>
<td>GET</td>
<td>/community/({forumName})</td>
<td>Get specific forum by forum name</td>
</tr>
<tr>
<td>DELETE</td>
<td>/community/({forumName})</td>
<td>Delete specified forum</td>
</tr>
<tr>
<td>POST</td>
<td>/community/({forumName})</td>
<td>Create new post in the specified forum</td>
</tr>
</tbody>
</table>
5. Fitbit

3.1.4 Risk Prediction

Another major implementation and the most important feature of the application is the CHD risk prediction feature for each individual based on the risk prediction algorithm developed by Dr. Youngwon Kim and his team. This section will explain the whole workflow of risk prediction starting from user data collection to the actual prediction and how it is implemented and integrated into the application.

3.1.4.1 Data Collection

As mentioned in Section 2.1.3, the prediction equation requires several input data from the user which includes users’ self-input data and users’ physical activity data. To collect these data, two separate ways are implemented in the application which was already explained in detail earlier in Section 2.2.2 for the self-input data. For the physical activity data, the workflow is a little bit different, once users can login into the application, they would then be immediately prompted to connect their Fitbit watch into the application too. Following the authentication process in Section 2.2.1 for integrating their Fitbit account to the application. An access key and refresh token will be sent by the frontend to the backend who will then store the tokens in the user collection in the database. These access key will then be used by the backend to obtain the physical activity data needed for the risk prediction.

3.1.4.2 Risk Calculation

Once everything is set-up correctly by the users, they can start to use the app to predict and visualize their own risk of developing CHD. Users must first enter the risk prediction tab as shown in Section 3.1.1 earlier and a request will then be sent
to the backend server to start calculating the risk prediction for that user. The backend API endpoint for calculating the risk will then be called and the backend will start the process of calculating the risk. The backend first fetches all the required input data from Firebase and Fitbit web API. Then the data will be assigned to the coefficient values as described in the biological equation as explained in Section 2.1.3 earlier.

The next step of the calculation process is the estimation of physical activity level or ENMO in the unit of milli-g. To estimate the ENMO value, as already mentioned in Section 2.2.1.1, the data of calories burned and total activity time throughout the day collected by Fitbit will be converted into ENMO value. Furthermore, to ensure that the risk prediction values is more representing of users’ lifestyle, the calories burned value used for the prediction on that day will be the average 7-day value of the calories burned starting from the day itself. This is done to ensure that users’ risk value does not fluctuate to the daily lifestyle as much because even if a user is not active on one day the risk prediction would still give a better representation because of the weekly average calculation instead of having a risk value for that certain day.

To convert calories burned and active minutes in a day into ENMO, there are several steps that need to be taken:

1. Calculate users’ BMR.
   BMR is defined to be the amount of energy expenditures in a day produced by the body when it is at rest [12]. The BMR value is important for the next step to estimate the intensity of activity that has been done by user on the day itself. Fitbit uses a well-known equation known as Mifflin – St Jeor to estimate users’ BMR value [13].

2. Calculate MET value.
   MET measures the ratio in which a person expends energy relative to their mass [14]. The MET value would then be helpful to estimate the intensity of activity in the unit J•min⁻¹•kg⁻¹. One formula that can be used to estimate MET from
calories burned, BMR, and duration of activity, which was referenced by Dr. Kim is as follow:

\[
METs = \frac{\text{Calories burned}}{((BMR \times (\text{Duration of activity in hours})) / 24)}
\]

3. Convert MET into ENMO in J•min⁻¹•kg⁻¹.

ENMO is a raw measure of an individual acceleration throughout the day which would then be able to be used to estimate an individual movement throughout the day. To calculate ENMO, a formula in a research paper written by White, et. Al. is used which was derived through experimentation of user acceleration in comparison to the MET values of the activity [15]. Note that there are many formulas that can be used in this case since there are many ways to estimate this value. The reason the simpler formula is chosen is to avoid overworking our backend by doing heavy computation and ensure smoothness to the application. Even though the equation is simple, it can still provide a relatively accurate estimation and hence is suitable for the project. The formula that was used for the project is as follow:

\[
(5.01 + 1 \times \text{ENMO})/71 = \text{MET (J•min}⁻¹\text{•kg}⁻¹)
\]

Rearranging the equation,

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

4. Convert ENMO into milli-g.

The last step is relatively simpler, since the physical activity values in the equation developed by Dr. Youngwon Kim was in the unit of milli-g, the ENMO value needs to be converted into this unit as well. To convert the value, the standard gravitational unit needs to be accounted for, which leads us to the conversion formula of:

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

Once the ENMO value is found, similarly, it would then be assigned to the coefficient weight pre-defined in the biological equation. To calculate the risk, the coefficient would then be multiplied according to the users’ parameter and summed up to be estimated according to the MeanX’B value of CHD risk probability data as aforementioned in Section 2.1.3.
3.1.4.3 Storing Risk Prediction Results

Once calculation is completed, the server then sends a response to the client for the request and display it in the UI. Other than that, the risk would also be stored in the risk collection according to the user, including the parameters and date. This was done to help user understand their risk prediction data more clearly and understand the trends of their CHD risk as explained in Section 3.1.1.

3.2 Limitation and Difficulties Encountered

Throughout the implementation phase, some limitations and difficulties were found when implementing the project using the current methodology and development tools. Each problem that has been encountered along with solution to minimize them would be discussed further in the sub-sections below.

3.2.1 Fitbit Request Rate Limit

The project relies heavily on Fitbit web API to collect users’ physical activity data, to be more specific, active calories burned and activity time. These data are required to be shown in the homepage of the screen to provide users with summary of their physical activities and used as input for the calculation of the risk prediction as well. The project initially envisioned to have a real-time visualization of this data and risk calculation. However, upon implementation, it was discovered that there is a limit for the rate of request towards the Fitbit web API to avoid users from spamming request into the server which can cause overload to the server. When this happens, an error code would then be returned instead of the correct response when requesting for certain data from the Fitbit web server which then can cause a bug to occur in the application as the application does not get the expected response from the Fitbit web server. To deal with this problem, a solution that has been implemented in the project is by limiting the query towards the Fitbit web API, currently, the request toward Fitbit API would be made every 10 minutes. So, the data can only be updated once every 10 minutes, including the risk prediction. We believe this solution is still good enough for the purpose of this application. Although it is not a real-time visualization, an interval of 10 minutes is good enough for users to understand their data summary throughout the day.
3.2.2 Fitbit Manual Sync

Another limitation found in this project with the current methodology is the Fitbit synchronization procedure. Fitbit requires its users to open their own Fitbit application in order for the data collected through the wearables to be synced into the Fitbit server and ultimately their web API.

Therefore, this might cause some discrepancies in the actual user data and the data provided by the server, as can be seen in Figure 23 below, as user must constantly remember to manually sync their data to the Fitbit server by opening the Fitbit application if they want to have the most updated data. There is nothing much that can be done in our end to solve this limitation as of right now other than waiting for Fitbit to improve this feature in the future.

![Figure 23 – Even though the Fitbit watch was used all day, the data shown in a) still shows that the steps and distances is 0, the Fitbit application shows the data that is currently in the server, hence, our application shows similar results. After syncing the device to the web API in b), the data then gets updated and so does our application.](image)

3.2.3 React Native Performance

As mentioned in Section 2.2.2, the client side of application is developed using React Native to take advantage of cross-platform development benefits. However, as already briefly mentioned, a major drawback of using React Native is its performance which is relatively slower compared to Native development tools such as Swift or Kotlin. A benchmark performed by inVerita [16], shows that application developed in React Native often use higher CPU power and consume more battery, almost as much as twice on
average. Furthermore, the computation power and efficiency are also weaker, typically performing slower when doing calculations and animations. To solve this problem, the project pans to keep the application simple and avoid including heavy animation and computation on the frontend side of the application which still fits the nature of the project. Therefore, this limitation should not cause a major hindrance to the project.

3.2.4 Unavailable Public DTC API

As mentioned in Section 3.1.1, the application is initially designed to have a DTC report page which shows user data collected from a genetic test done by a provider known as 23andme. Users’ genetic data can then be used as an additional input to the algorithm for a more accurate prediction. However, during the set-up phase of this feature, it was then discovered that the API was not publicly accessible by developers. Developers must first sign-up and explain their application before given access to the API they provided. Furthermore, 23andme has a certain application window for the API application and during our project timeline, the application was already closed. As a result, this feature was unable to be integrated into the project and was not implemented into the application. Moreover, as this feature was optional, we consulted with Dr. Kim from the School of Public Health of The University of Hong Kong and decided that it is best to not pursue this feature right now as risk prediction can still be done without the genetics data. This feature would be best implemented in the future as will be explained in more detail later in Section 5.2.
4 Contributions

The application in this project is developed by a group of four individuals, each with their own contributions to the project. This section will explain in detail the contributions that have been made by myself, i.e., which part of the application and what major contributions have been made by the author for the development of the application. Initially, the team was divided into two sub-teams, one in charge of Frontend (Mellisa Hadipranata and I) and the other in charge of the backend (Chan Hay Yin and Hom Long Hin). However, some parts of the backend was also done by the frontend team due to time constraints of the project. The sub-sections below will then mention and explain in detail which specific part of the components is developed by the author.

4.1 Application Frontend

There are several screens and UI components of the mobile application that needs to be developed. Therefore, the frontend team has divided the responsibility by assigning screens to be developed per team member. Note that even though the task was divided in this manner, both Mellisa and I also helped each other and developed minor components that was needed for the other screens that was not initially assigned to us. The UI that I was mainly responsible for was as follow:

4.1.1 Community Page

The community page explained in Section 3.1.1 was mainly developed by me. I oversaw the whole process starting from the initial screen which first request data of list of articles from the backend stored in the database. I developed the UI component as shown in Figure 24 a) below, the logic behind the UI development in this part is to map over the data response from the backend into a forum card component which shows the title, description, and image of the forum as inputted by the admin. This means that my task was implementing the forum card component and also integrating with the backend to get the required data to be put inside the UI component.
Clicking this card component will redirect user to the list of articles posted in that specific forum. I have implemented the routing logic accordingly and again integrating with the backend to request for the article summary containing image or articles, author, title, and date of the articles. I developed the UI component, as shown in Figure 24 b), for this part to have a similar design to the previous page to have a consistency in design and make the application less confusing with unnecessary design. The only part that I need to modify in this UI component is the parameters of the component.

After clicking one of the articles in the list shown, user will be redirected to the article content. Therefore, I had to develop another request to the backend server and shows the article data correspondingly. Other than that, this screen also contains a comment section where user can post their opinion once they finish reading the articles. I had to implement the logic to communicate with the backend to store the comments and user information into the database for the corresponding article, while also rendering the UI comment component into the screen at the same time.

Figure 24 – The card components that I have developed for the community features including its functionality such as routing process and request to the backend. a) the card component to show forum in the forums list b) the card component to show article in the forum

Figure 25 – The comment components that I have developed, implemented to fetch both the user detail and comment detail posted in specific article.
4.1.2 Authentication Process

Another major contribution and task that I was in charge of was the authentication workflow of users, i.e., login and signup process. As mentioned earlier in Section 2.2, the authentication process was done directly to Firebase without communicating with the backend as we feel that this would be more efficient in terms of API calls and response time, managing sessions would also be more efficient with the availability of built-in library provided by Firebase that can be used in the frontend. Hence, I had to first understand how the library works and implement the authentication process accordingly, sending request to the firebase server with the credentials and routing them to the correct screen if the authentication is successful. I also had to handle the error process when users’ authentication fails and inform them of the error and how to fix them through the UI. For the signup process, it has a slightly different approach, I had to request data from the backend instead so that the backend is able to create a user document which would later be used to store users’ personal and health information. Other than the basic authentication features such as login and signup, I also had to develop additional authentication feature such as changing password.

4.2 Admin Frontend

I was also in charge of the admin page UI, which was developed as a separate web-application. The admin page has limited functions as of right now due to the limited time frame of the project and only has one screen which shows a summary of posted articles grouped by the forums and also functions to create new forum and posts. I developed most of the UI components in the admin screen starting from the overall design to the components in the page. The logic in the admin frontend has similar flow to the one in community screen of the mobile application. The only difference is that, in the admin page, the layout is different since this is a web-application, it can show much more information as compared to the smaller screen of smartphones. I have developed the card components which shows the data that I have requested from the backend which was able to be clicked as can be seen in Figure 26 to show the relevant post information as a popup as mentioned earlier in Section 3.1.2. In addition, a delete button which then shows a confirmation pop-up has also been developed which I then used to handle
the delete logic accordingly, communicating with the backend endpoint to delete the relevant data from the database.

As mentioned in Section 3.1.2, there will be pop-up to handle group creation and article creation too. I had to develop the pop-up components for each function respectively and implement the request logic with the corresponding payload data to communicate with the backend and ultimately store these data into the database.

4.3 Backend

As mentioned, due to time constraints, some tasks on the backend team were also off-loaded to the frontend team so that the project could be completed in time. Some of the functions inside the backend Flask server that I had to help develop will be explained in the sub-sections below.

4.3.1 Risk API

For the risk prediction, I helped developed the risk calculation API, which can be accessed from the /community/<user_id>/<date> endpoint. In this endpoint, I had to request the 7-day active calories from the specified data backwards and then find the average. Similarly for the active minutes as well, a 7-day average is also calculated. Then, I calculated the ENMO value as explained in Section 3.1.3 by estimating it from the estimated value of MET based on these data requested from the Fitbit. Chan Hay Yin has initially set-up the risk prediction coefficients into the endpoint based on the algorithm that Dr. Kim gave our team. However, I had to finalize the algorithm to match with the business logic of our application such as using 7-day average and using data collected in Fitbit.
Aside from the actual risk prediction, I also developed other utility endpoints for the risk endpoint API in the backend. These utility endpoints are used to give data as requested by user, two other endpoints used in the risk API includes the daily summary of users’ risk prediction value in a specified month, and the monthly summary of users’ risk prediction value in a specified year. These values will then be plotted to the visualization graph as mentioned earlier.

4.3.2 Fitbit API

Another API endpoint that I helped develop is the Fitbit endpoints in the backend. I had to first understand the workflow and all the necessary URL endpoints that I must use to get the necessary data from the Fitbit web API. Some of the data that need to be collected for this application purpose include the 7-day calories burned from a specified date, 7-day activity time from a specified date, and the daily summary of users’ physical activity data. Other than the general request and processing of data, I had to also understand and implement the server type grant flow with PKCE as part of the oAuth 2.0 authentication method required by Fitbit to give our application the access token to access users’ Fitbit data once they finish the authentication process in our application by signing into Fitbit with the authorization URL. As part of the oAuth 2.0 authentication method, I had to set-up a code challenger and code verifier initially while also returning user with the correct authorization URL and receive the authorization code back in the server to be exchanged with the access and refresh token which will then be stored in the database.

4.3.3 Community Page API

One other major API endpoint that I had contributed to in the backend was the community endpoint. As mentioned earlier in Section 4.2, the admin frontend would need to communicate with the backend regularly to show summary of the articles posted and also to post new articles into the application. I had contributed into this part of the backend as well. In specific, the general CRUD functions to help manage articles data in the application was developed. Hom Long Hin, a member of the backend team, then continued to pick up my work and finalized it with the whole Flask application server.
5 Future Works

Although the main feature of the application has been developed fully and the project is successful in achieving its aim and goal. Given more time, there are still some features that can be beneficial to be implemented in the future if the project is to be picked up by other developers. The sub-sections below will explain some of the ideas that our team can recommend implementing into the application in the future.

5.1 Integrating Other Brands of Smartwatches

Due to the limited time frame of the project, only one wearable brand is integrated into the application, Fitbit. Some users might already owned other brands of smartwatches and this cannot be overlooked as it would be a major inconvenience for users to have to purchase Fitbit just to use the application. The priority of the project was to get the risk prediction running and it would be too time consuming to integrate other brands of smartwatches since the developers would have to first understand the documentation and the API documentation for other brands might be different as well. Incorporating other brands will indirectly increase the compatibility of the application and also aligns well with our aim to increase awareness of CHD by expanding the user reach of the application. Users experience of the application would also be improved as they can select the wearables they prefer instead of limited into one wearable brand.

5.2 DTC Report

The initial design of the application includes a feature to integrate users’ DTC test result into the application which could be used as an alternative to some of the parameters in the risk prediction algorithm as mentioned in Section 3.2.4. However, again, due to the unavailability of the API from the provider 23andMe, this feature was not able to be developed into the application as of right now. By adding DTC report feature into the application, user experience would be improved as they now have more options to automatically sync genetics data instead of manually inputting the other health related data. Other providers of genetics data than 23andMe can also be explored in the future to add this useful feature into the application, which ultimately will provide user with insightful data regarding to their physical and health information.
5.3 Accelerometer to Collect Physical Activity Data

As mentioned in Section 2.2.1.2 earlier, another alternative was initially tried to estimate the physical activity level of the users which would then be used as the input parameters to the risk prediction equation. However, it was discovered that the only way to access the raw accelerometer data through Fitbit is through the device API and not the conventional web API that is currently being used in this project along with other problems mentioned such as higher consumption of computing and battery power in the device as the application will need to run constantly in the background to track any kinds of movements of the users. The reason why we believe it is worthwhile to pursue the accelerometer alternative in the future is because predicting ENMO from raw accelerometer data might give a more accurate representation of the users movement and hence, physical activity level data for the equation might be more accurate. Although the current methodology that the project use to estimate ENMO is good enough for the purpose of this application, there is always room for improvement for the accuracy of the risk prediction. The formula for conversion of calories burned into ENMO are all based on estimation formulas that we have researched and hence might have some inaccuracy since they are also just an estimation as mentioned in Section 3.1.4. Given more time, building an in-app watch application could be pursued to improve the accuracy of tracking users’ physical activity and users could also be given options whether to keep using the formula which can be less accurate but consumes less device power or be more accurate but having a trade-off in having a constantly running application in the background of their devices. To implement this in Fitbit, users can be prompted to download the additional in-watch application if they decided to use the accelerometer method to estimate the physical activity. In the future, in case Fitbit allows certain application to constantly run in background, this application can then start to be developed. Other smartwatches brands might have a different logic to the implementation of this feature and might need to be researched in the future too.
5.4 Additional Features for Admin Page

As mentioned in Section 3.1.2, the features of admin page is currently still limited to managing the community feature of the application. However, additional features can be implemented in the future to help the admin manage the application more efficiently and easily. The admin page can be a powerful and useful tool to control the state of the application if implemented correctly with additional features in the future. One example of such additional features that would be extremely useful would be a feature to manage users of the application, from which they could edit, delete or add users accordingly. This would be helpful in case of any bug that occurs in the user account and users were not able to solve the bug by themselves and require assistance from the admin. As of right now, managing users can only be done through the Google Firebase console which is not user-friendly to users that do not understand application development. Hence, this feature would be a good and beneficial if it is to be implemented in the future. Another reason why additional features would be beneficial for the admin page is the admin can control the state of the application and delete unwanted data in the community pages for example. In case of any guidelines violation and users misuse the application, they could ban the users or give warnings to the user through the admin page and delete the harmful content to the application directly from the admin page. As such, this would ultimately improve users’ experience when using the application.
6 Conclusion

CHD remains to be a serious and dangerous disease and will continue to take more lives not only in Hong Kong, but also the rest of the world if no further steps are taken. Hence, actions need to be taken quickly to raise people’s awareness of its danger and educate people of what needs to be done to mitigate its risk. One way to solve this problem is to utilize the trend in digital health technology, mHealth, which has allowed tracking health related data to be easier and more accessible than ever. Insights can then be gained on certain disease and individuals’ current health lifestyle and how it might impact their chances on developing specific diseases. Therefore, the project aims to take advantage of the ever-advancing digital health technology to help reduce the risk of CHD by creating a well-integrated mobile application that allows its users to visualize their tracked data from the wearables and ultimately their risk of developing CHD. With this application, users will be more aware of the danger of CHD and how their health lifestyle has impacted it, promoting a healthier lifestyle at the same time.

The project mainly focuses on building a cross-platform mobile application that helps users to track their risk of developing CHD in the next 10 years. The application will also provide visualization methods for users’ risk of developing CHD and their physical activity level, in hopes to raise awareness of the importance of physical activity towards reducing risk of CHD. As a collaboration project with the School of Public Health at The University of Hong Kong, our team was able to focus more on the development of the application while the genomics part of the project, i.e., the risk prediction algorithm is developed by Dr. Youngwon and his team. The algorithm that was used for risk prediction takes in two main input, the self-input data such as age; BMI; gender; etc., and physical activity data collected from the wearable device. Our responsibility is then to implement this algorithm inside the application, starting from the data collection, all the way to the risk prediction, i.e., how to use the input data and calculate the risk prediction value. This collaboration allows the project to achieve its maximum potential, having a well-made application integrated with a risk prediction algorithm that is fairly accurate and then able to achieve its maximum potential to reduce the risk of CHD.
All in all, the project has been completed successfully as the application has been made fully running with all the necessary features included, i.e., risk prediction feature. An additional feature of community has also been implemented in the project to add user-interactivity for the application. The frontend of the application is developed using React Native to enjoy the benefits of cross-platform design such as saving development time while being able to get a bigger market size of popular smartphones today and also promotes consistency in design at the same time. The backend of the application is developed using Flask, Firebase and Firestore, in which the risk prediction algorithm is implemented in, taking and storing required input given from the user frontend into the database, while also showing the resulting calculations into the frontend, where users can visualize and understand their data there.

Although the project is successful, the current implementation still has its limitations and also encountered some difficulties throughout the development. However, these limitations were able to be mitigated and also minimized by alternative methodologies which is why the end-goal of the project was still able to be met, producing a well-developed mobile application. There are also still room for improvements and future works if the project is to be picked up in the future. More features can be implemented and integrated into the existing application and alternative development tools can be used to migrate the existing code to ensure maximum performance for the application.
7 References


