Final Report

Development of a Smartphone App for Prediction of Coronary Heart Disease Risk (Industry-based Project)

Team Member:

Hom Long Hin (3035692450)
Hadipranata Mellisa (3035663019)
Harney Vieri (3035663368)
Chan Hay Yin (3035686047)

Supervisor:
Dr. Loretta Choi

Date of Submission: 18/4/2023
Abstract

Coronary heart disease (CHD) is a major cause of death in Hong Kong and around the world. The risk of developing CHD can be influenced by hereditary and lifestyle variables, while properly monitoring CHD risk could save many lives. This project aims to provide a simple and streamlined mobile application that enables users to self-monitor their risk of getting CHD, hence reducing it. In collaboration with The University of Hong Kong's School of Public Health, this project will be using a series of risk prediction algorithms developed by Dr. Youngwon Kim and his team. The development team will be using React Native to develop a cross-platform mobile app, Fitbit as metrics collecting devices, Flask as a web API, and Firebase as the backend database. The later part of the report outlines the schedule, challenges, and limitations faced by the team, followed by potential improvements on the product such as integrating machine learning for health advice generating service.
Acknowledgment

I would like to thank Dr. Loretta Choi for supervising our project, and Mr. Jerry Tam for providing useful, inspiring advice on development.

I would like to give credit to my groupmates, Hayward, Melissa, and Harney for their contributions.

I would also like to thank Miss Grace Chang for guiding me on technical writing.

Finally, the project would not exist without Dr. Youngwon Kim, from the School of Health. His ideas lead to this industrial FYP project.
# Table of Contents

ABSTRACT .......................................................................................................................... I

ACKNOWLEDGMENT ........................................................................................................... II

TABLE OF CONTENTS ........................................................................................................... III

LIST OF FIGURES ................................................................................................................ VI

LIST OF TABLES .................................................................................................................... VIII

ABBREVIATION ...................................................................................................................... IX

1. INTRODUCTION ................................................................................................................. 1

   1.1. BACKGROUND ........................................................................................................... 1

   1.2. MOTIVATION .............................................................................................................. 1

   1.3. OBJECTIVES ............................................................................................................... 2

   1.4. DELIVERABLES .......................................................................................................... 2

   1.5. REPORT OUTLINE ...................................................................................................... 3

2. RELATED WORK ................................................................................................................. 5

   2.1. FRAMINGHAM RISK SCORE CALCULATOR .............................................................. 5

   2.2. ASCVD CALCULATOR ............................................................................................. 6

   2.3. EVALUATION .............................................................................................................. 6

3. METHODOLOGY .................................................................................................................. 8

   3.1. PLATFORM WORKFLOW ............................................................................................ 8

   3.2. IMPLEMENTATION ....................................................................................................... 9

      3.2.1. Frontend .............................................................................................................. 10

      3.2.1.1. React Native .................................................................................................. 10
3.2.1.2. SwaggerUI ........................................................................................................ 11

3.2.2. Backend ........................................................................................................ 11

3.2.2.1. Python Flask and Flask-RESTx ................................................................. 11

3.2.2.2. Google Firebase .......................................................................................... 12

3.2.2.2.1. Cloud Firestore and Firebase Storage .................................................... 12

3.2.2.2.2. Firebase Authentication .......................................................................... 12

3.2.2.2.3. Firebase Local Emulator Suite ............................................................... 13

3.2.3. Wearables ...................................................................................................... 13

3.2.3.1. Fitbit API and OAuth 2.0 ........................................................................... 14

4. RESULTS AND FINDINGS .............................................................................. 15

4.1. FRONT-END ................................................................................................. 15

4.1.1. Mobile App ................................................................................................. 15

4.1.1.1. Project Structure ...................................................................................... 15

4.1.1.2. Screens and UI Design ............................................................................. 16

4.1.1.2.1. General .................................................................................................. 16

4.1.1.2.2. Homepage ............................................................................................ 18

4.1.1.2.3. Risk ........................................................................................................ 19

4.1.1.2.4. Community .......................................................................................... 20

4.1.2. Web Dashboard ........................................................................................... 22

4.1.2.1. Project Structure ...................................................................................... 22

4.1.2.2. Screens and UI Design ............................................................................. 23

4.1.3. SwaggerUI ................................................................................................... 24

4.2. BACKEND ..................................................................................................... 25

4.2.1. Project Structure ......................................................................................... 26

4.2.1.1. REST API ................................................................................................. 27
List of Figures

Figure 1. High-level Overview of mobile app workflow .............................................. 8
Figure 2. Overview of System Infrastructure ................................................................. 9
Figure 3. Firebase Emulator UI ..................................................................................... 13
Figure 4. Example of Applying OAuth 2.0 between Client and Resources ............ 14
Figure 5. Project Structure of Mobile App ................................................................. 15
Figure 6. Login Screen; .............................................................................................. 17
Figure 7. Signup Screen .............................................................................................. 17
Figure 8. Signup Successful Splash Screen .................................................................. 17
Figure 9. Self-input Form ............................................................................................. 17
Figure 10. Wearables Connecting Screen; ................................................................. 18
Figure 11. Fitbit Login Screen; ..................................................................................... 18
Figure 12. Sample Homepage Dashboard .................................................................. 19
Figure 13. Personal Profile Page .................................................................................. 19
Figure 14. Change Password Page ............................................................................... 19
Figure 15. Sample Risk Analysis ................................................................................ 20
Figure 16. “Show All Data” Page ............................................................................... 20
Figure 17. Front Page of Community ......................................................................... 21
Figure 18. List of Latest Posts ...................................................................................... 21
Figure 19. List of Comments under articles .............................................................. 22
Figure 20. Project Structure of Web App ................................................................. 23
Figure 21. Homepage of Dashboard .......................................................................... 24
Figure 22. User Management in the Dashboard Application .................................. 24
Figure 23. SwaggerUI for backend API ....................................................................... 25
Figure 24. Descriptions of Model for API ............................................................... 25
Figure 25. Fields to input custom parameters and “Try it out” ........................................ 25
Figure 26. Expected Result for 200 Success Response ....................................................... 25
Figure 27. Project Structure of API .................................................................................. 26
Figure 28. Data Models for User ....................................................................................... 29
Figure 29. Sample Data for User ....................................................................................... 29
Figure 30. Sample Data in Firebase Authentication ............................................................ 30
Figure 31. Workflow of a Unit Test .................................................................................... 30
Figure 32. Homepage of Firebase Console ......................................................................... 31
List of Tables

Table 1. Some conversion of Points-to-Risk between Male and Female ............... 5
Table 2. Comparison of three popular development languages for Frontend .......... 10
Table 3. Overview of Endpoints in /user .................................................. 27
Table 4. Overview of Endpoints in /risk ................................................... 28
Table 5. Overview of Endpoints in /community ......................................... 28
Table 6. Overview of Endpoints in /fitbit ................................................... 29
Table 7. Project Schedule ......................................................................... 32
Table 8. Contribution of Members ............................................................. 33
## Abbreviation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASCVD</td>
<td>Atherosclerotic Cardiovascular Disease</td>
</tr>
<tr>
<td>BMI</td>
<td>Body Mass Index</td>
</tr>
<tr>
<td>CHD</td>
<td>Coronary heart disease</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create, Read, Update, Delete</td>
</tr>
<tr>
<td>FRS</td>
<td>Framingham Risk Score</td>
</tr>
<tr>
<td>iOS</td>
<td>iPhone OS</td>
</tr>
<tr>
<td>NoSQL</td>
<td>Non-SQL</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform-as-a-Service</td>
</tr>
<tr>
<td>REST</td>
<td>REpresentational State Transfer</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSO</td>
<td>Single Sign-On</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>YAML</td>
<td>YAML Ain’t Markup Language</td>
</tr>
</tbody>
</table>
1. Introduction

This section provides an overview of the project by presenting its background and motivation, followed by the objectives, and expected deliverables of this project.

1.1. Background

Coronary heart disease (CHD) remains a significant global health issue, contributing to substantial mortality and morbidity worldwide. Over the past six decades, heart-related ailments have consistently ranked as the third leading cause of death in Hong Kong, accounting for 13.0% of total fatalities in 2020 [1]. The diagnosis and treatment of CHD often necessitate sophisticated medical technologies, resulting in considerable expenses and strain on healthcare systems. As a result, early detection and intervention are essential to mitigating the effects of this disease on both patients and the healthcare industry.

Simultaneously, the rapid advancements in mobile technology have created novel avenues for utilizing smartphones and smartwatches as effective tools for individuals to assess their CHD risk and proactively implement preventive measures.

1.2. Motivation

Some lifestyle-related risk factors for developing CHD are controllable and preventable, including obesity and blood pressure [2]. Research has indicated that maintaining a healthy lifestyle is linked to a decreased risk of CHD and stroke, regardless of hereditary risk [3]. In comparison to an unfavorable lifestyle, a favorable lifestyle was linked with 37% and 30% decreased risks of CHD and stroke, respectively, in all genetic risk categories.
Currently, mobile apps are becoming a popular platform for monitoring health status and analyzing health data trends. Some health apps played a useful role in tracking risk factors and metrics. Yet, there seems to be a lack of applications that provide an integrated CHD risk assessment that shows an overall risk. Also, most apps failed to leverage personal data and could not provide personalized recommendations and concrete insights according to a user’s unique health profile and priorities.

1.3. Objectives

In this project, the development team aims to develop a user-friendly smartphone application that allows users to easily and constantly monitor their own risk of getting coronary heart disease, based on their health data. It provides personalized risk assessments and recommendations after analyzing users' genetic and lifestyle factors with an arithmetic algorithm.

This integrated solution would be able to empower individuals with tools to monitor and manage their health by incorporating a simple and convenient way to assess their CHD risk into citizens’ smartphones. Furthermore, the app can serve as an educational tool, increasing awareness of CHD risk factors and promoting healthy lifestyle choices. It can also facilitate communication between patients and healthcare providers, enabling more targeted and effective preventive care. By accomplishing these objectives, this project is hoped to potentially improve the early detection and management of CHD.

1.4. Deliverables

To fulfill the objectives, the following are delivered in the project:

• A properly documented Application Programming Interface (API) that:
• Implements a series of algorithms in collaboration with The University of Hong Kong's School of Public Health, which can calculate the predicted user's CHD risk.
• Handles authorization and gets data from Fitbit Company to get users’ physical data regularly and store them in a database.
  ▪ Supports CRUD operations between end-node applications and databases.
  ▪ Allows future developers to leverage it for more end-node health applications.
• A functional and user-friendly mobile application that:
  ▪ Allow users to manage their personal accounts and health information.
  ▪ Allow users to track their own CHD risk with the aid of graphical visualizations and analysis of individual metrics.
  ▪ Allow users to connect with their own Fitbit trackers to retrieve lifestyle metrics.
  ▪ Provide dynamic insights on lifestyle adjustments or treatments to coach users on behavioral changes for CHD prevention.
  ▪ Allow users to join communities to read healthcare-related articles and share ideas with other individuals.
• An easy-to-use web dashboard application that:
  ▪ Allow admins to manage articles in Community efficiently.
  ▪ Allow admins to manage users efficiently.

1.5. Report Outline

To present the project, the following report is structured into five chapters. Chapter two presents some related work of the project and analyzes existing problems. Chapter three illustrates the methodology of the application, including the frontend and backend
workflows and implementations of the application, followed by the justifications behind them. Chapter four describes the challenges and limitations faced by the project team, followed by proposed solutions. Chapter Five sums up the project schedule. Chapter six concludes the report by summarizing the content of the project.
2. Related Work

This section is delivered to understand the current situation and existing tools to predict coronary heart disease risk and to evaluate the limitations of existing approaches.

2.1. Framingham Risk Score Calculator

The Framingham Risk Score (FRS) is one of the most widely recognized and utilized models to estimate the risk of developing coronary heart disease. It was commonly used by healthcare providers to guide preventive measures such as lifestyle modifications. The model takes various risk factors into account in the calculation, including age, sex, and diabetes status.

<table>
<thead>
<tr>
<th>Point Total</th>
<th>10-year risk</th>
<th>Point Total</th>
<th>10-year risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 0</td>
<td>&lt; 1</td>
<td>&lt; 9</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>0-4</td>
<td>1</td>
<td>9-12</td>
<td>1</td>
</tr>
<tr>
<td>5-6</td>
<td>2</td>
<td>13-14</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>&gt;= 17</td>
<td>&gt;= 30</td>
<td>&gt;= 25</td>
<td>&gt;= 30</td>
</tr>
</tbody>
</table>

Table 1. Some conversion of Points-to-Risk between Male and Female

However, there are numerous limitations to the approach. Framingham Heart Study, the study that developed this scoring system, only included white participants, which is not generalized enough to include all ethnic groups globally. Furthermore, only a very limited set of risk factors are considered, without including common factors such as physical activities and diet habits. It may also not be able to reflect an accurate
association in the modern world, as the study was conducted in the 1970s when people lived different lifestyles.

2.2. ASCVD Calculator

Another recent tool that is leveraged to calculate CHD risk is the Atherosclerotic Cardiovascular Disease (ASCVD) risk calculator, developed by the American College of Cardiology. It also estimates an individual 10-year risk of developing CHD by considering risk factors including smoking status, blood pressure and medications. Some online calculators utilize the Pooled Cohort Equations derived from the study and return the required risk.

Although the Equations has also considered African American besides White individuals, it is still not accurate enough for other racial backgrounds. Moreover, the estimation obeys several assumptions, including that the patients have not developed the cardiovascular disease before, and that they are 40 to 79 years old. A more generalized tool is to be implemented.

2.3. Evaluation

After studying several common risk prediction models, these major limitations are identified:

- The algorithm should be generalized to more people, especially those of different races and ages.
- The algorithm should consider more risk factors that could lead to CHD, e.g., medications, genetics, and physical activities.
- The calculation process is not convenient:
  - Parameters involved in prediction might not be easily accessible, e.g., blood
pressure and cholesterol levels, which are measured in clinics.

- Manual input of parameters can lead to input errors and reduced motivation to learn about one’s own risk in the disease.

The following project aims to overcome these limitations. Deploying integration with mobile phones and smart watches allows users to get and view their data easily by streamlining the whole fetch-and-calculate process. We are also using an algorithm to predict an accurate result of getting CHD risk, in collaboration with and referencing studies from the School of Public Health at the University of Hong Kong. This value is calibrated from several key indicators, including age, sex, BMI, diet, smoking status, alcohol consumption, physical activity, and medication use of cholesterol, blood pressure, and insulin.
3. **Methodology**

In this section of the project report, more elaboration on the workflows and designs in the app implementation will be done, including the justifications behind the technology choices.

### 3.1. Platform Workflow

The workflow of the platform (Fig. 1) shown below is based on three core components: mobile app, cloud API, and database. The first time starting the mobile app, users are prompted to log into their accounts or sign up by entering genetic information and connecting to their own Fitbit accounts. Once the user is logged in, the application will begin its regular workflow cycle.

![Diagram of mobile app workflow](image.png)

Figure 1. High-level Overview of mobile app workflow

Upon starting the application, contents within the application are rendered. An overview of the workflow is as follows:

1. **Data synchronization:** To periodically synchronize with the latest information and ensure users have access to the most recent data, API calls are made periodically.
2. **Fitbit Authorization and Data Fetching:** Given the user has connected to his own Fitbit account, the mobile app can be authorized and fetch relevant lifestyle metrics
from Fitbit API, such as heart rate and activity logs.

3. Risk Calculation: The web application behind cloud API processes the data and calculates the user’s CHD risk based on the algorithm, incorporating pre-inputted factors such as age and gender.

4. Data Storage: All processed and fetched data are stored in the database, which allows users to access their historical data and keep track of their progress.

5. Data Visualization: Calculated results are returned to the mobile device so that the mobile application will eventually display the result in intuitive graphs.

Other provided functions, such as providing health advice, work similarly in this app-API infrastructure.

3.2. Implementation

This section further outlines the methodologies used that are critical in the project. Multiple components form the technology stack in the project, and Figure 2 shows a brief picture of how components interact with each other.

![Figure 2. Overview of System Infrastructure](image-url)
3.2.1. Frontend

The frontend components are responsible for the user interface (UI) and user experience. Justifications on React Native and SwaggerUI will be discussed.

3.2.1.1. React Native

React Native is used during front-end development in this project. React Native is a popular framework developed by Facebook. The open-source JavaScript library has high reliability and provides a hot-reloading feature [4], which accelerated the development. React Native also allows developers to create cross-platform performant mobile applications with its native rendering engine. Given the short turnaround time for this project, React Native is the best option where maintaining a single codebase for a cross-platform application helped ensure that features and designs are consistent, thus saving time and resources to develop separate apps for iOS and Android.

The following table compares React Native with its alternatives.

<table>
<thead>
<tr>
<th></th>
<th>React Native</th>
<th>Flutter</th>
<th>Swift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Platforms</td>
<td>Android + iOS + Web</td>
<td>Android + iOS</td>
<td>iOS only</td>
</tr>
<tr>
<td>Library Support</td>
<td>Most</td>
<td>Fewer</td>
<td>Least</td>
</tr>
<tr>
<td>Documentation</td>
<td>Similar</td>
<td>Similar</td>
<td>Fewest</td>
</tr>
<tr>
<td>App Size</td>
<td>Medium</td>
<td>Largest</td>
<td>Smallest</td>
</tr>
<tr>
<td>Performance</td>
<td>Worst</td>
<td>Better</td>
<td>Best</td>
</tr>
<tr>
<td>Results</td>
<td>Selected</td>
<td>Not selected</td>
<td>Not selected</td>
</tr>
</tbody>
</table>

Table 2. Comparison of three popular development languages for Frontend

Compared to other native frameworks like Flutter, a UI toolkit from Google that uses the language Dart, React Native has a larger community and provides a more authentic native experience, with only slightly slower performance. To summarize, React Native
is a good choice for this project because no intensive animations or complicated calculations are involved within the application while providing an appealing experience. In addition, React is also leveraged to develop the web dashboard application to ensure consistency between UIs.

### 3.2.1.2. SwaggerUI

As the API directly serves as a web application, a static web page is not involved in the project. Instead, the API implemented SwaggerUI that generates interactive documentation and allows developers to understand and interact with the API in a self-explanatory way. Users can directly utilize the web application like a web frontend. SwaggerUI helps users explore the API’s endpoints, input parameters, and data models without referring to separate documentation, by executing API calls directly from the documentation. Future developers can leverage it while building end-node applications and testing the API.

### 3.2.2. Backend

The backend components acted as the backbone of the product which supported the main business logic. Justifications on Python Flask and Firebase are discussed.

#### 3.2.2.1. Python Flask and Flask-RESTx

Flask is a popular lightweight micro-framework built on Python based on Werkzeug and Jinja. It provides powerful functionalities including URL routing and request handling, while its development debugger and server provide a hot-reloading function like React Native which boosts the development process and shows immediate changes. For its rich built-in features and minimal boilerplate to build an API, Flask is used in the project to serve database CRUD operations and Fitbit data fetching to reduce workload within the mobile application and ensures reusability for future development.
In addition, its extensible architecture and useful extensions can be leveraged and plugged in for more complex usage. An extension Flask-RESTx is utilized to build a more RESTful API, which provides support for request data validation when parsing incoming requests and response marshaling to ensure consistency as an API, along with error handling features that define custom error responses. Furthermore, Flask-RESTx can automatically generate Swagger UI for documentation which simplified the development process.

3.2.2.2. Google Firebase

Another core backend solution adopted is Firebase, a cloud platform from Google that provides a suite of backend services and tools. Serving as an integrated backend-as-a-service, several key features simplified the development process and reduced the time and resources needed for infrastructure management.

3.2.2.2.1. Cloud Firestore and Firebase Storage

Firebase offers a serverless database service namely Cloud Firestore, which enables the team to store and synchronize data across multiple platforms. Being a NoSQL database allowed the team to store documents in flexible data models without pre-defined schema and query data of any kind. It offers higher scalability compared to a SQL database, which is beneficial to the project because the scope of the deliverables has been subjected to further changes along the timeline.

Firebase Storage is another file storage solution incorporated into the application that can be integrated with Firestore and is used to manage media such as images.

3.2.2.2.2. Firebase Authentication
Authentication services from Firebase allowed the components to manage user authentication securely. It is crucial in the project to safeguard the private information gathered from users, including their personal health and identification data. Furthermore, Firebase provided rich UI libraries and a wide range of sign-in methods such as social media logins and SSO providers. This makes sure of secure logins while offering high flexibility for developers to streamline the sign-up process that is tailored to specific development requirements.

3.2.2.2.3. Firebase Local Emulator Suite

The local emulator suite is powerful during the testing stage and offered a set of tools that facilitated the team to prototype and run local unit tests without affecting existing contents on live Firebase. Emulators for Cloud Firestore, Authentication, and Storage have been used locally for Quality Assurance.

![Firebase Emulator Suite](image)

**Figure 3. Firebase Emulator UI**

3.2.3. Wearables

Smartwatches are crucial in the project to collect physical data from users and be used
in risk calculations. While there are many choices for wearable trackers, devices from Fitbit are selected as pilots, which is a popular brand and has a large consumer base. They are also able to track various metrics relevant to the algorithm, including heart rate and activity levels.

3.2.3.1. **Fitbit API and OAuth 2.0**

Developers can utilize the Fitbit API to integrate applications with Fitbit devices when data fetching is needed. To secure users’ data, third-party applications are required Fitbit to use the OAuth 2.0 Authorization Framework, which grants minimal permissions to third-party applications without sharing login credentials. To minimize user manual controls and streamline the authorization process, both the mobile application and REST API will handle and automate most procedures, such as generating code verifiers and exchanging authorization codes for tokens, according to the OAuth 2.0 specification.

![Figure 4. Example of Applying OAuth 2.0 between Client and Resources](image-url)
4. Results and Findings

This chapter presents the resulting product from the development team, described in 2 major parts: frontend and backend respectively in sections 3.1 and 3.2.

4.1. Front-end

The frontend portion of the deliverable included the mobile application to be directly delivered to most application users and a SwaggerUI to be delivered to API users.

4.1.1. Mobile App

The implementation of the smartphone app relies on React Native framework and numerous open-source libraries that supported the UI and application workflow. This section will outline the implementations and designs in the three main features: Home, Risk, and Community.

4.1.1.1. Project Structure

![Project Structure](image)

Figure 5. Project Structure of Mobile App
Developing a React Native mobile app, the powerful tool Expo was leveraged to streamline the app development, which offered an environment for easier tests and previews on simulators, and access to APIs and libraries such as Firebase. Other external package dependencies included Axios for calling API endpoints and Victory React Native for generating appealing statistical graphs.

The development team also adopted the strategy of having a modular structure for arranging related files in the repository. Static files such as images are located in ‘assets’, and reusable UI components are stored in ‘components’. The Javascript in the ‘screens’ folder represents different UI screens in the app respectively, while leaving App.js as the code entry point and will serve navigation between screens and tabs.

### 4.1.1.2. Screens and UI Design

For basic users, the mobile application consists of three fundamental screens: application homepage, risk prediction, and health forums. All User Interfaces are designed to be native to users’ devices, interactive, and easy to use. This section will be further divided into multiple sub-sections to showcase the implementation.

#### 4.1.1.2.1. General

Some screens in the mobile app are not categorized into the above tabs, including splash screens and user sign-ups, and thus categorized as General in this report. Upon downloading and launching the app, the user is prompted to log into their accounts (assuming the smartphone has not stored previous log-in credentials) (Fig. 6). The user can also choose to register for a new account, with email and password (Fig. 7). Once the signup is completed, a splash screen appears notifying account creation is successful (Fig. 8).
After successfully creating a new account with validations, the user is required to input his information on personal and genetics, such as date of birth and smoking status (Fig. 9), following a screen prompting the user to connect to a Fitbit wearable (Fig. 10).
The user shall then be prompted to log into a Fitbit account to connect to their wearables after a native pop-up browser showed up (Fig. 11). After logging in successfully, the user is navigated to the homepage of the application.

4.1.1.2.2. Homepage

The homepage is a dashboard that contains three components: the user's profile, the user's collected data, and a simple chart representing the user's performance (Fig. 12). The dashboard allows users to have a quick overview of their recent performance in reducing their CHD risk, including activity data fetched from Fitbit. Using the navigation bar at the bottom of the screen guides the user to other pages.
The user can view his profile by clicking the avatar icon (Fig. 13), change the account password by clicking the button at the bottom of the profile page (Fig. 14), or change the profile picture by clicking Edit Image, which will subsequently pop up a native photo gallery. The ‘<’ button on the top left allows the user to return to the previous page.

4.1.1.2.3. Risk

The second page shows a detailed analysis of the time series on the user's CHD risk. A summary of the user's risk and data appears on the top of the page, visualized by line graphs (Fig. 15). By clicking the “View trend” button in the middle, the graph will be toggled between “View one month” and “View one-year” mode. Furthermore, clicking “refresh” allows the user to view the latest trend by updating the numbers and graphs.
The team implemented literal messages that tell users the change in predicted risk over time, making the analysis more intuitive. A more detailed list can be accessed by “show all data”, which gives users the list of all available risk prediction results across dates (Fig. 16).

4.1.1.2.4. Community

The Community is an extra implemented feature, which is originally out of the project scope yet designed for better user experience. The project team hopes the users using the app could have the chance to share, either health advice or personal experience. The
Community can be accessed at the bottom-right corner (Fig. 17). Users can click into any of these forums to view articles and posts under the specific forum, which are related to reducing heart disease risks in a certain area (Fig. 18). Users can also share opinions by leaving comments under the articles (Fig. 19).

Figure 17. Front Page of Community

Figure 18. List of Latest Posts
4.1.2. Web Dashboard

A simple web application is built on React, which serves as a dashboard for community administrators to use additional features such as creating new articles in the forum. This section will outline the dashboard’s implementations.

4.1.2.1. Project Structure
Similar to mobile apps, the repository is composed of components and screens, whereas most UI components in the application are directly imported from the MUI library to mimic a more web-like, dashboard-like theme.

4.1.2.2. Screens and UI Design

Administrators can view existing posts across various forums, including respective dates of creation and numbers of comments and likes, where clicking into each card component allows admins to do operations such as editing the post (Fig. 21).
Forum moderators can also manage the users who participated in the community. By selecting any of the users, moderators can view details of the user’s profile and choose to remove their rights from viewing or posting content in the forum.

4.1.3. SwaggerUI

The SwaggerUI serves as a documentation and web UI for the Python API and is automatically generated from the API code (Fig. 20), while descriptions for endpoints
and models are manually updated (Fig. 21).

![SwaggerUI for backend API](image1)

![Descriptions of Model for API](image2)

These descriptions allow API users to know how to leverage these endpoints.

![Fields to input custom parameters and “Try it out”](image3)

![Expected Result for 200 Success Response](image4)

By inputting the arguments required and clicking “Try it out” (Fig. 22), responses are reflected immediately so that testing can be done effectively and shall be expected to match the data model in Example Value (Fig. 23)

### 4.2. Backend
As mentioned in Methodologies, the backend involving API and database was implemented mainly using Python Flask and Firebase. This section showed the result of the backend implementation.

4.2.1. Project Structure

The API is developed using the Flask framework in a modular approach as shown below (Fig. 3), which is optimized for scalability and readability.

- The ‘app.py’ file serves as the entry point and is responsible for starting the app server, while its fellow namespaces are modularized in the ‘apis’ folder, such as ‘user.py’ handling CRUD endpoints only related to users.
- The ‘model.py’ stored some frequently used data models for request parsing and response marshaling.
- ‘test.py’ contains several unit tests that help to evaluate the product.
- ‘config.yaml’ is a YAML file listing customized environment variables and configurations as code, such as Fitbit credentials and Firebase parameters.

![Figure 27. Project Structure of API](image)
‘requirements.txt’ maintains only the roots of library dependencies. Upon installing all dependencies, libraries that are involved in the code included:

- Firebase Admin Python SDK
  (https://firebase.google.com/docs/reference/admin/python)
- Flask-RESTX (https://flask-restx.readthedocs.io/en/latest/)
- PyYAML (https://pyyaml.org)
- Requests (https://requests.readthedocs.io/en/latest/)

### 4.2.1.1. REST API

The mobile application relies on the REST API to support its functionalities, including user management, metrics fetching, risk calculations, and forum management. To facilitate API calls, routings are done to set up URLs for different functions and use cases. Functions under the above 4 major features are arranged into endpoints under 4 namespaces respectively. Details are shown below for /user (Table 3), /risk (Table 4), /Fitbit, and /community.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Route</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/user/</td>
<td>Returns the list of ID of users (not public visible)</td>
</tr>
<tr>
<td>POST</td>
<td>/user/</td>
<td>Creates a new user, in both Authentication and Database</td>
</tr>
<tr>
<td>GET</td>
<td>/user/&lt;uid&gt;/</td>
<td>Returns all data of the specified user with UID</td>
</tr>
<tr>
<td>DELETE</td>
<td>/user/&lt;uid&gt;/</td>
<td>Deletes the specified user with UID</td>
</tr>
<tr>
<td>GET</td>
<td>/user/info/&lt;uid&gt;/</td>
<td>Returns personal info of the specified user with UID</td>
</tr>
<tr>
<td>PUT</td>
<td>/user/info/&lt;uid&gt;/</td>
<td>Updates personal info of the specified user with UID</td>
</tr>
<tr>
<td>GET</td>
<td>/user/health/&lt;uid&gt;/</td>
<td>Returns health info of the specified user with UID</td>
</tr>
<tr>
<td>PUT</td>
<td>/user/health/&lt;uid&gt;/</td>
<td>Updates health info of the specified user with UID</td>
</tr>
<tr>
<td>GET</td>
<td>/user/name/&lt;uid&gt;/</td>
<td>Returns the username of the specified user with UID</td>
</tr>
<tr>
<td>GET</td>
<td>/user/verifyData/&lt;uid&gt;/</td>
<td>Returns whether the specified user completed sign-up</td>
</tr>
<tr>
<td>PUT</td>
<td>/user/verifyWearable/&lt;uid&gt;/</td>
<td>Returns whether the specified user connected Fitbit</td>
</tr>
</tbody>
</table>

Table 3. Overview of Endpoints in /user
Besides account settings, /user endpoints are also used in other features, such as getting usernames from UID in the Community instead of storing usernames repeatedly in the database.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Route</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/risk/&lt;uid&gt;/</td>
<td>Returns the overall risk of specified user</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/&lt;uid&gt;/&lt;date&gt;/</td>
<td>Returns the overall risk of user on the specified date</td>
</tr>
<tr>
<td>POST</td>
<td>/risk/&lt;uid&gt;/&lt;date&gt;/</td>
<td>Creates new risk data of user on the specified date</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/weekly/&lt;uid&gt;/</td>
<td>Returns the risk of specified user in the nearest week</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/monthly/&lt;uid&gt;/</td>
<td>Returns the risk of specified user in the nearest month</td>
</tr>
<tr>
<td>GET</td>
<td>/risk/yearly/&lt;uid&gt;/&lt;year&gt;/</td>
<td>Returns the risk of specified user in the specified year</td>
</tr>
</tbody>
</table>

Table 4. Overview of Endpoints in /risk

Several risk endpoints are implemented for risk data queries at different time intervals. Combinations of these endpoints shall be leveraged for specific use to minimize calls to Fitbit API due to Fitbit's restriction on 150 calls per user per hour.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Route</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/community/</td>
<td>Returns the list of existing forums (communities)</td>
</tr>
<tr>
<td>POST</td>
<td>/community/</td>
<td>Creates a new forum</td>
</tr>
<tr>
<td>GET</td>
<td>/community/&lt;forumID&gt;/</td>
<td>Returns data of the specified forum with forum ID</td>
</tr>
<tr>
<td>POST</td>
<td>/community/&lt;forumID&gt;/</td>
<td>Creates a new post under the specified forum</td>
</tr>
<tr>
<td>DELETE</td>
<td>/community/&lt;forumID&gt;/</td>
<td>Deletes the specified forum</td>
</tr>
<tr>
<td>GET</td>
<td>/community/post/&lt;postID&gt;/</td>
<td>Returns data of the specified post with post ID</td>
</tr>
<tr>
<td>POST</td>
<td>/community/post/&lt;postID&gt;/</td>
<td>Creates a new comment under the specified post</td>
</tr>
<tr>
<td>DELETE</td>
<td>/community/post/&lt;postID&gt;/</td>
<td>Deletes the specified post</td>
</tr>
</tbody>
</table>

Table 5. Overview of Endpoints in /community

These endpoints are more straightforward and handle CRUD operations for the Community feature.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Route</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/fitbit/auth/url/&lt;uid&gt;/</td>
<td>Returns a redirect URL for Fitbit user authentication</td>
</tr>
</tbody>
</table>
GET /fitbit/auth/token/  | Returns the tokens required to fetch Fitbit data |
POST /fitbit/user/<uid>/ | Returns user’s profile on Fitbit |
GET /fitbit/activities/<uid>/ | Returns user’s daily activities summary on Fitbit |
GET /fitbit/weeklyAvgCal/<uid>/| Returns user’s weekly average burnt Calories |
GET /fitbit/weeklySteps/<uid>/ | Returns number of user’s weekly steps on Fitbit |
PUT /fitbit/storeToken/<uid>/ | Updates user’s authorization token on database |

Table 6. Overview of Endpoints in /fitbit

These endpoints shall not be directly manipulated by users, and instead should only act as utilities by the mobile app while handling OAuth operations and Fitbit API.

4.2.1.2. Database and Authentication

The integrated solution from Firebase is implemented in the project's backend, serving as both the database and authentication provider. Cloud Firestore, which is NoSQL, does not require a fixed schema, and stores data like dictionaries. However, to ensure consistency between objects and that some mandatory information is received, some data models are designed to filter data when parsing incoming data.

---

![Data Models for User](image1)

![Sample Data for User](image2)

Figure 28. Data Models for User
Figure 29. Sample Data for User
For user authentication, Firebase Authentication allows developers to implement user authentication-related operations without revealing users’ credentials, which promoted the privacies and security of the information. In the project, only the email-password sign-in method is enabled. The following information (Fig. 7) can be viewed in the Authentication console.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Provider</th>
<th>Created</th>
<th>Signed In</th>
<th>User UID</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:spitest@gmail.com">spitest@gmail.com</a></td>
<td></td>
<td>Apr 17, 2023</td>
<td>Apr 17, 2023</td>
<td>dP0OTVC9tEpYDj/6W3WJ4J12</td>
</tr>
<tr>
<td><a href="mailto:hello@mal.co">hello@mal.co</a></td>
<td></td>
<td>Apr 15, 2023</td>
<td>Apr 15, 2023</td>
<td>NY02xKv6yp0JbjFy6yB6q0Z1W8B2</td>
</tr>
<tr>
<td><a href="mailto:le@gmail.com">le@gmail.com</a></td>
<td></td>
<td>Apr 15, 2023</td>
<td>Apr 15, 2023</td>
<td>jtV7HvQgAixxQv190/B7N0B2</td>
</tr>
<tr>
<td><a href="mailto:xuy@gmail.com">xuy@gmail.com</a></td>
<td></td>
<td>Apr 15, 2023</td>
<td>Apr 15, 2023</td>
<td>mtS5xmyLmBBU5N0B16h1q280</td>
</tr>
<tr>
<td><a href="mailto:lgy@gmail.com">lgy@gmail.com</a></td>
<td></td>
<td>Apr 15, 2023</td>
<td>Apr 15, 2023</td>
<td>KgT1xJ/k6O1wE3x88y590h20c3</td>
</tr>
<tr>
<td><a href="mailto:xia@gmail.com">xia@gmail.com</a></td>
<td></td>
<td>Apr 15, 2023</td>
<td>Apr 15, 2023</td>
<td>chJNh3Kymv4kFy59B1h0n1yf2</td>
</tr>
<tr>
<td><a href="mailto:tyl@gmail.com">tyl@gmail.com</a></td>
<td></td>
<td>Apr 14, 2023</td>
<td>Apr 14, 2023</td>
<td>K/WCEk1/p6tL6h0xLJ3B3</td>
</tr>
<tr>
<td><a href="mailto:melisa@gmail.com">melisa@gmail.com</a></td>
<td></td>
<td>Apr 14, 2023</td>
<td>Apr 14, 2023</td>
<td>d/E0y/cKlfw0Ta1/jz1yF85x1</td>
</tr>
<tr>
<td><a href="mailto:android@gmail.com">android@gmail.com</a></td>
<td></td>
<td>Apr 13, 2023</td>
<td>Apr 13, 2023</td>
<td>qreU5P0B7EyeWkH091y08ZL</td>
</tr>
<tr>
<td><a href="mailto:test123@gmail.com">test123@gmail.com</a></td>
<td></td>
<td>Apr 13, 2023</td>
<td>Apr 13, 2023</td>
<td>q/5y6h5x5y/5/3S5rE29h4n2</td>
</tr>
</tbody>
</table>

Figure 30. Sample Data in Firebase Authentication

4.2.1.3. Unit Tests and Monitoring

Unit tests are done to test endpoints granularly using the unit test Python library. The coverage scores can be referenced to improve code quality. The following chart shows how a unit test on the backend is done in the project.

Figure 31. Workflow of a Unit Test
Utilizing Firebase as a PaaS (Platform-as-a-Service), the usage of products can be easily monitored at the homepage of Firebase console UI (Fig. 32). These metrics can be viewed as indicators of the performance and stability of the product after release.

Figure 32. Homepage of Firebase Console
5. Project Development

This section discusses the details of the development progress during the project.

5.1. Project Schedule

The following table outlines the development schedule of the project (Table 2).

<table>
<thead>
<tr>
<th></th>
<th>October 2022</th>
<th>November 2022</th>
<th>December 2022</th>
<th>January 2023</th>
<th>February 2023</th>
<th>March 2023</th>
<th>April 2023</th>
<th>May 2023</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Finalized app architecture.</td>
<td>• Complete basic UI</td>
<td>• Disease risk prediction algorithm implementation</td>
<td>• First presentation</td>
<td>• Additional features implementation (e.g., health article page)</td>
<td>• Finalized platforms.</td>
<td>• Final report</td>
<td>• Project exhibition</td>
</tr>
<tr>
<td></td>
<td>• Finalized UI design.</td>
<td>• Complete basic app features</td>
<td>• Smartwatch API integration</td>
<td>• Preliminary implementation</td>
<td></td>
<td>• Finalized tested implementation</td>
<td>• Final presentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Set up database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Completed</td>
<td>Completed</td>
<td>Completed</td>
<td>Completed</td>
<td>Completed</td>
<td>Completed</td>
<td>Incomplete</td>
<td>Incomplete</td>
</tr>
</tbody>
</table>

Table 7. Project Schedule

5.2. Contributions

Table 7 shows the distribution of tasks within the project team.

<table>
<thead>
<tr>
<th>Frontend</th>
<th>Mobile Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UI Implementation</td>
</tr>
<tr>
<td></td>
<td>(e.g., Data visualization, Community Page)</td>
</tr>
<tr>
<td></td>
<td>Harney &amp; Mellisa</td>
</tr>
<tr>
<td>Task</td>
<td>Responsible</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Fitbit Connection</td>
<td>Harney &amp; Mellisa</td>
</tr>
<tr>
<td>Web Dashboard</td>
<td></td>
</tr>
<tr>
<td>UI Implementation</td>
<td>Harney &amp; Mellisa</td>
</tr>
<tr>
<td>API</td>
<td></td>
</tr>
<tr>
<td>SwaggerUI Documentation</td>
<td>Justin</td>
</tr>
<tr>
<td><strong>Backend</strong></td>
<td></td>
</tr>
<tr>
<td>API</td>
<td></td>
</tr>
<tr>
<td>/user and /community, REST management</td>
<td>Justin</td>
</tr>
<tr>
<td>/risk</td>
<td>Hayward</td>
</tr>
<tr>
<td>(Including algorithm design and literature review)</td>
<td></td>
</tr>
<tr>
<td>/fitbit</td>
<td>Harney &amp; Mellisa</td>
</tr>
<tr>
<td>Unit tests</td>
<td>Hayward</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td></td>
</tr>
<tr>
<td>Firestore Management and Database Design</td>
<td>Justin</td>
</tr>
<tr>
<td>Authentication Management</td>
<td>Justin</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td></td>
</tr>
<tr>
<td>Project Plan</td>
<td>Harney &amp; Mellisa</td>
</tr>
<tr>
<td>Project Webpage</td>
<td>Justin &amp; Hayward</td>
</tr>
<tr>
<td>Agile and Notion Management</td>
<td>Justin</td>
</tr>
</tbody>
</table>

Table 8. Contribution of Members

During the second half of the project timeline, spanning from January to April 2023, the project team adopted the Agile work environment with the aid of Notion. In this approach, the project was divided into multiple granular tasks, where the team aimed to complete each task within the sprint cycle of 1 week, and the results shall be discussed and evaluated in weekly regular team meetings. The project team believed this approach promoted benefits such as increasing team collaboration and reduced risks by allowing early identification and continuous progress tracking. The development has greatly sped up and qualities of deliverables are improved.
6. Discussion

This section describes the challenges faced during the development stage and gives recommendations on the project for future work.

6.1. Limitations

The development team identified the following limitations for the project.

6.1.1. Data Collection and Privacies

User genetic data and physical data collected from smartwatches formed the foundation of the project, which is used to predict users’ coronary heart disease risk. However, collecting these necessary health data is challenging due to privacy concerns. Genetic data was originally designed to be imported from genetic testing services from companies like 23andMe, yet it is difficult to be a partner and collaborate with these companies to streamline data fetching, eventually tuning the team’s approach to inputting genetic data manually. For lifestyle metrics to be fetched from wearables API, developers are required to declare purposes and usages in detail and ensure the team clearly understands security policies during applications to accessing API. As a result, there are limited ways to retrieve necessary data given the small scale of the project.

6.1.2. Technical Requirements in Using Application

This application relies heavily on the Internet and smartwatches, along with smartphones, to handle API calls and fetch necessary data for risk calculations. However, these can be less accessible in developing countries, or less affordable for particular communities, especially since certain platforms for applications and smartwatches are required, which limited the use of the application. The project team
has paid an effort in reducing the need for the Internet, such as reducing API calls and the size of the payload to alleviate the problem partially.

6.1.3. Limitations in Calling API

For security reasons, Fitbit bounded an application’s rate limit to making 150 API requests hourly for each user, regardless of purpose. Therefore, API calls from the mobile application must be optimized, without breaking the user experience. One approach made is capping users’ ability of refreshing data to only once every 10 minutes, resulting in a total of 25 possible requests for one risk prediction cycle which are further divided for requesting various metrics such as activities and heart rates. On the other hand, the API executes the most efficient calls. For example, to get monthly data, time series endpoints are utilized to reduce the number of endpoint calls, instead of calling 30 daily data.

6.2. Challenges

The development team encountered the following challenges during the development.

6.2.1. Collecting Correct Metrics

Being the initiative of the project, the project team collaborated with the School of Public Health on the risk prediction algorithm. After analyzing the materials provided, the team successfully developed the algorithm, while getting the required parameters has been a challenging task. Having genetics data being pre-inputted and most metrics being explicitly retrieved from Fitbit, the parameter ENMO cannot be directly fetched from smartwatches which are deducted from users’ acceleration values. Several extraction methods have been explored, including getting raw accelerometer values from Fitbit which is going to require another application developed for Fitbit itself; or
getting raw accelerometer values from the smartphone, which would be heavily biased and inaccurate. Eventually, the development team can estimate ENMO by converting the calories burned, which is also related to energy expenditure and can be fetched directly from wearables.

6.2.2. Obstacles during Collaboration

The development team is segmented into two smaller teams for frontend and backend separately, to finish sub-tasks effectively. Although shared repositories like GitHub have been used to avoid code conflicts, team members met difficulties to understand and leverage each other’s product, especially when the team is not familiar with all practices and methodologies. To facilitate communication, documentation is made for readability, and instant messengers helped to solve issues instantly.

6.3. Recommendations

The project team would like to suggest some future work available to overcome the limitations mentioned and improve the product.

6.3.1. Provide Personalized Health Advice

The project team was not able to implement another feature due to time constraints, which is giving tailored recommendations to users after analyzing users’ health profiles. Future work could involve generating useful advice by applying big data analysis and machine learning to users’ metrics and showing specific ways to reduce CHD risk such as quitting smoking and targeting exercise goals or pushing app notifications to users to remind and motivate them for behavioral changes in their daily lives.

6.3.2. Expanding Range of Data Sources

Involving more parameters in the calculation might be useful in predicting a more
accurate risk score, including exploring fetchable metrics from wearables, and linking with more healthcare and wearables providers in facilitating all-rounded risk calculations.

**6.3.3. Refining Product Infrastructure**

Some useful libraries and tools can be paired with the product to overcome its limitations and shortcomings, such as leveraging Flask-JWT-Extended and Flask-User for better authentication and user management. The product shall also be deployed and hosted on platforms such as AWS Cloud and Google Cloud Engine.

**6.3.4. Involving more Stakeholders**

The users using the product can be differentiated to be an either ordinary user who wants to tack their own CHD risk or community administrator who wants to manage certain forums and users. The project team would also suggest adding roles for doctors, where therapists can also utilize the app to monitor patients’ CHD risk and suggest corresponding recommendations for lowering risks.
7. Conclusion

Since technology is developing quickly, numerous inventions today can effectively spot illness symptoms and treat them, saving countless lives. But in addition to being expensive, these diseases and medical operations often cause some physical harm to patients before they can fully recover. A successful, convenient, accessible risk prediction tool could be essential to improve people’s quality of life.

The applications in the solution aim to be a tool that everyone can know how likely they are to get heart disease and prevent getting CHD by simply maintaining healthy lifestyles according to the health advice provided. Existing models with limitations including being insufficiently generalized and inaccessible are expected to be replaced.

The algorithm in the application can compute users' risk of developing coronary heart disease based on genetic and lifestyle data analysis, which gives clarity on potential personal risk factors for the condition. The implementation of applications also involved React Native being responsible for user-friendly interfaces, Firebase being selected for multiple provided services, and Python Flask housing business logic that reinforces the whole product.

The project team faced several challenges during the development stage, including being unable to fetch responsible parameters from data sources, which is eventually solved by finding another way to derive the desired argument. Future work and improvements on the solution include applying machine learning to users’ lifestyle metrics and providing tailored suggestions on reducing CHD risk. Additionally, adding more roles such as doctors would provide a one-stop solution for a better healthcare system.
8. Reference


