Exploring and Expanding the Functionalities of the MobileCoach Platform

Bachelor Thesis in Computer Science and Engineering, 15 credits

Halmstad 2020-02-06
Jonas Fockstedt, Ema Krcic
Acknowledgements

We would like to show our sincerest gratitude towards the MobileCoach team who showed great enthusiasm for our thesis whom gladly, when we got in touch, wanted to collaborate with us. We would like to specifically thank Prabhakaran Santhanam, a MobileCoach software engineer, who despite his busy schedule, set aside time to have meetings with us every week. He gave us incredible insight into the MobileCoach platform, describing possible solutions to our problems and answering our many bewildered questions with excitement and for always making us burst into laughter through our long meetings.

A warm thank you to our supervisor, Wagner Ourique de Morais, for guiding us through this thesis. With dedication that always helped us improve our work, his brilliant way of explaining and questioning problems from different perspectives made us more critical and observant of our work. He would always provide us with helpful feedback and answering our confused questions with an eager will to help.

We want to specifically thank Anita Sant’Anna for her participation in the early stages of this thesis where her ideas and research were the beginning of this candidate thesis. She was the one that suggested this candidate thesis to be done on the MobileCoach platform. Providing us with great input on how to solve difficult problems with enthusiasm that motivated us through difficult times. Despite our baffling emotions, always giving wise advice with such kindness and warmth.

Big thanks to our families, for the constant love and support. As the great Michael J. Fox once said "Family is not an important thing, it’s everything," we couldn’t agree more. We love you and we are blessed to have you in our lives!

Yours truly,
Ema & Jonas.
Abstract

Non-communicable diseases are the greatest burden on global healthcare, and because of today’s limited health personnel, scalable behavioral health interventions are in need. MobileCoach is a behavioral intervention platform designed for aiding behavioral change amongst individuals. The work from this candidate thesis extends the MobileCoach platform by integrating the mobile application with Google Fit. Letting the platform collect the amount of steps an intervention participant has taken during a given day with the help of Google Fit. This required modification to the code of the mobile application, allowing MobileCoach to query for user Google Fit step data. As a contribution to MobileCoach, future work in the platform can consist of measuring the intensity of steps. That will give a better measurement of the health condition within the participant rather than only measuring the amount of steps.

Sammanfattning

Contents

1 Introduction 1
  1.1 Purpose & Goals ................................................. 1
  1.2 Structure of this Thesis ........................................ 2

2 Background 3
  2.1 MobileCoach ..................................................... 3
    2.1.1 Description ................................................. 3
    2.1.2 Server Client and Its Interaction With the Mobile Application ........... 5
    2.1.3 Interventions ............................................... 8
    2.1.4 Mobile Application ....................................... 11
    2.1.5 Chat Bot ..................................................... 12
    2.1.6 Chat Messages ............................................. 12
    2.1.7 Security Aspects .......................................... 15
  2.2 Related Work .................................................... 15

3 Theory 17
  3.1 React Native ................................................... 17
  3.2 Google Fit ..................................................... 18
  3.3 User Authentication with OAuth 2.0 ............................. 18

4 Method 23
  4.1 Method Specification .......................................... 23
  4.2 Selection of Tools ............................................. 23
  4.3 Method Description ........................................... 24
    4.3.1 Acquiring OAuth Client ID ................................ 24
    4.3.2 Authorizing MobileCoach to Access Google Fit Steps Data ............... 25
    4.3.3 Querying Step Data of A Participant .......................... 25
    4.3.4 Integrating Authorization and Step Retrieval For the Chat Bot ........... 26
  4.4 Validation of Results .......................................... 26

5 Results 27

6 Discussion 33
  6.1 Implemented Chat Bot Commands ................................ 34
  6.2 User experience ............................................... 34
  6.3 What Could Have Been Done Differently? ......................... 35
  6.4 Future Work .................................................... 35

7 Conclusions 37
Chapter 1

Introduction

The greatest burden on global health are non-communicable diseases (NCDs)[1][2], such as heart diseases, asthma, obesity, diabetes or chronic kidney diseases. Adverse use of alcohol and tobacco or physical inactivity are unfavorable health behaviors[3][4] that is a consequence to NCDs according to WHO\(^1\) (World Health Organization)[5]. Because of today’s limited health personnel, scalable behavioral health interventions[6] are in need to be able to improve efficacy and reduce costs of preventive or therapeutic behavioral interventions. Therefore there is an enormous potential in innovative digital health interventions (DHIs)[7].

With the help of MobileCoach, a behavioral intervention platform, health professionals are enabled to design scalable, low-cost and evidence-based DHIs. Today, the MobileCoach platform provides the ability to design interventions aimed towards improving everyday behavior amongst its participant’s. However, the interventions currently lacks the ability to take participant step data into consideration when guiding them towards behavioral change.

This thesis will focus on expanding the functionality of MobileCoach by integrating it with Google Fit, a health-tracking platform. The integration is essential because the MobileCoach platform needs more data points to better determine progress amongst intervention participants. Google Fit helps in this aspect by providing its measured amount of steps an individual has taken in a given time interval.

1.1 Purpose & Goals

The purpose of this thesis is to integrate the MobileCoach platform with Google Fit. This will be used to fetch the amount of steps a given user has taken during a certain time interval. The MobileCoach platform will act accordingly to the amount of steps a user has taken during this time interval. In order to achieve the purpose of this thesis, there are some concrete goals that needs to be met:

- Understand the MobileCoach platform and its architecture in order to design the integration with Google Fit.
- Extend the MobileCoach mobile application by enabling users to log into their Google account and authenticate MobileCoach to access their step data in Google

\(^{1}\)https://www.who.int/
Fit. With this authentication, MobileCoach should be able to query the amount of steps a user has taken during a given time frame.

- Make the step data retrieved from Google Fit to be stored in the MobileCoach platform.

1.2 Structure of this Thesis

In chapter 2, the relevant background for this thesis is described. This includes the MobileCoach platform as a whole, the described components and their roles in the system. After that, related work will be described, where other applications utilizing Google Fit will be mentioned as well as other applications for MobileCoach.

The theory that stands for the groundwork is described in chapter 3. In this chapter, the role of React Native for this thesis will be described as well as Google Fit and how it operates. The chapter will lastly walk through user authentication with OAuth 2.0.

Next, chapter 4 focuses on the methodology of this thesis, where the method specifications will be specified. Later, the tools selected for the thesis will be described and motivated. Following that, the methods used to achieve the results will be detailed as well as how to validate that the results corresponds to the goals of this thesis.

The results achieved will be described in chapter 5, followed by a discussion regarding the achieved results in chapter 6, where pros and cons of the final work will be considered. Lastly in chapter 7, conclusions about this thesis will be drawn and possible future work will be brought up.
Chapter 2

Background

This chapter will be dedicated to describing the MobileCoach platform. First, an overview of the platform as a whole is presented. After this, the server client and its interaction with the mobile application will be pointed out and visualized. Also, a walk-through of the interventions is made as well as presenting the mobile application. With it, the chat bot and the chat messages is presented in detail. This will be followed up by presenting the security of the platform. Lastly for this chapter, related work will be brought up. All of these components are important in order to understand the entire flow of the MobileCoach platform. This knowledge helps identify how to integrate the MobileCoach platform with Google Fit.

2.1 MobileCoach

2.1.1 Description

Developed at ETH Zürich, Switzerland, the MobileCoach platform[8][9] is an open source behavioral intervention platform, which aims to provide fully-automated digital interventions. The platform is built on the principles of automata theory[10], which assumes a system with different states, where a transition can be made between the different states. A transition can be made by a certain input, and depending on which state the system is currently in, a transition can be made into another state.

The objective of the MobileCoach platform is to be able to provide a platform that offers a cost-efficient way of creating and maintaining health interventions. These interventions start by letting a participant take part of a base assessment survey. The answers on this survey will determine which state a particular participant will be placed in the system. For example, if one would have an intervention aimed to reduce alcohol consumption, which also has 4 different states, a participant may already start in the second state, depending on their answers to the survey. A participant is also able to move between different states in the system. This is made by letting a participant answer, at a given interval, questions regarding their health and current progress. If they fulfill the criteria for the state of the current intervention, they move on to the next state. One can set up different rules for each intervention, and these rules define how to move from one state to the other, amongst other things (i.e. time interval between surveys).

One use case for the MobileCoach platform could be where an individual seeks help
in reducing their consumption of alcohol. This individual will become a participant of an intervention on the platform, where this intervention focuses specifically on alcohol consumption. Depending on the rules of the intervention (more on that in subsection 2.1.3), the participant receives messages from the platform, where the contents may vary. The platform could inform the participant about the health drawbacks of alcohol consumption, or it could ask the participant how many glasses they have drunk during the day, increasing the awareness of their consumption. It has been shown that adolescent smokers who consume larger amounts of alcohol on a single occasion have had a positive effect[11] whilst using the platform.

Initially, a participant could only interact with the MobileCoach platform via SMS. However, in September 2019, the MobileCoach team released version 3 of the platform. With it, a standalone mobile app[12] for the platform was released, where participant’s could interact with a chat bot that was directly connected to MobileCoach.

A general overview of the platform is shown in Figure 2.1. A domain expert designs an intervention, and loads it into the MobileCoach server. This intervention is observable by a health professional from the Cockpit, a view designed for making the health professional as efficient as possible by giving them access to participant messages and communicate with them directly. The intervention is mainly sent to participant’s, who are using the mobile application of the platform.

Figure 2.1: An illustration of the MobileCoach platform, where the interface between the domain expert, health professional and participants is showed.
2.1.2 Server Client and Its Interaction With the Mobile Application

The MobileCoach platform lets developers deploy their own instance of a MobileCoach server with ease thanks to Docker\(^1\). Docker is an open source platform that is based on the increasingly popular container\(^1\) technology. The idea of containers is that every process runs inside its own environment. These containers can later be run together to create a full-scale application that contains a server component, a database component and so on.

Thanks to this technology, Docker offers a lightweight methodology to deploy applications. Figure 2.2 helps explain why Docker is such a good alternative to developing and distributing applications. A developer have their physical hardware as well as their OS. On top of this, an instance of the Docker Engine is run. This engine shares resources with the kernel of the host OS and enables fully-functional applications to be run.

![Figure 2.2: Visual representation of the Docker schema.](image)

Provided on MobileCoach’s BitBucket wiki\(^2\), the MobileCoach team have provided instructions on how to deploy a server for MobileCoach with Docker. This Docker setup contains three components:

- **MongoDB**\(^3\) - Database used for storing information about participant’s to a particular intervention. MongoDB stores its data in a JSON-like format.

- **Apache Tomcat**\(^4\) - Open source Java server that is used to prompt third party servers to send push notifications to the mobile application. It is also responsible for handling the logic, the rules and the interventions on the platform.

- **Deepstream**\(^5\) - Real-time server used as a communication bridge between the mobile application.

---

\(^1\)https://www.docker.com/
\(^2\)https://bitbucket.org/mobilecoach/mobilecoach-documentation/wiki/Setting-up-the-server-with-docker
\(^3\)https://www.mongodb.com/
\(^4\)https://tomcat.apache.org/
\(^5\)https://deepstream.io/
bile application, MongoDB and Tomcat. It is responsible for formatting and syncing messages between these services.

These three components can be translated to the programming design pattern and methodology Model-View-Controller(MVC)[14]. As the name suggests, this design pattern consists of three components, the Model, the View, and the Controller component.

Referring to Figure 2.3⁶, the Model is the part of the application that holds the data. Here, the data is formatted to the desired format and the logic for that data is defined. The View holds responsible to represent this data in such a way that a user can understand it. This requires the View to communicate with the Model and specify which data is to be sent to the View. The Controller is the component to manipulate the data that resides in the Model component. How the Controller chooses which data to manipulate, and how, is dependent on user input.

![Model-View-Controller design pattern.](image)

The MobileCoach server has been divided into three major layers, where each layer represents a corresponding component in the MVC design pattern. These layers consists of a persistence layer, service layer, and application layer. The persistence layer translates to the model component. This layer consists of the database storing the data on the platform, which in this case is MongoDB. The service layer corresponds to the controller component and, as earlier mentioned, where the request of data manipulation in the persistence layer takes place. In this layer, resources such as project Lombok⁷, Javaluator⁸ and JavaMail⁹ reside. These resources aid for the data manipulation. The application layer is therefore the view component. This layer takes use of the Vaadin framework¹⁰ and the template engine Mustache¹¹ to help make a visual representation of the data residing in the persistence layer.

---

⁶Has not been reviewed by third party.
⁷https://thesislombok.org/
⁸http://javaluator.sourceforge.net/en/home/
⁹https://www.oracle.com/technetwork/java/javamail/index.html
¹⁰https://vaadin.com/
¹¹https://mustache.github.io/
When designing interventions (more on that in subsection 2.1.3), an intervention administrator can set up different rules. Logical operations can be performed with these rules, and whether these rules are true or not is up for the Tomcat server to check. Depending on if the criteria is fulfilled or not, it triggers a push notification that notifies the participants through SMS, email or push notifications in the mobile application (will be presented in more detail in subsection 2.1.4) via Firebase (push notification server for Android) or an Apple push notification server. This is done by sending a notification to the external servers. The external servers consists of an SMS gateway (a service for distributing SMS messages), an email server, a Firebase server and an Apple notification server. The flow and connection of these different components is shown in Figure 2.4. The SMS and email services are only used when there is a need of reminding the participant to participate in the study. For example, if the platform senses that a participant has not used the MobileCoach application for some time, it will trigger the Tomcat server to send a SMS or email to the participant to remind them to open the mobile application. Communication with the email server is unidirectional, meaning the participant can not answer the email messages sent to them. However, a participant is able to respond to the SMS messages sent from the SMS gateway.

The chat messages from the mobile application are sent in JSON-format (more on this in subsection 2.1.6) and they are handled by communicating with the Deepstream server, which uses secure web connections. This connection between the mobile application and the Deepstream server exists only when the app is active. This means that the server can not collect any data from the mobile application when a participant is not using it, since Deepstream is the server that formats the data from the mobile application. This is solved by, at fixed intervals, sending push notifications to the application, which makes it wake up for a few minutes, enough time to get the data necessary. The mobile application also has a direct link to the Tomcat server, where REST

https://www.codecademy.com/articles/what-is-rest

Chapter 2 Jonas Fockstedt & Ema Krcic 7
services, is used as a communication medium. This link is used when specific variables of a participant/intervention needs to be written to or read from.

2.1.3 Interventions

The core functionality of the MobileCoach platform are the interventions. The interventions are the key for behavioral change within a participant. Two different interventions may vary greatly from one another, since each intervention is defined by its own rules and may focus on different areas.

These rules define how the system interacts with a participant. For example, a given intervention may be focused on reducing smoking and will therefore give different responses, in the form of messages to a participant, depending on how many cigarettes a participant has been smoking on a particular day. If a participant may have had too many cigarettes than what is the maximum for that given day, the platform may send a message to the participant addressing this. These rules can be easily configured by the intervention administrator via the web application for a given intervention. The administrator for the interventions do not have to posses any programming skills in order to design an intervention.

There are a wide variety of different configurations to be made for interventions. Some of these include, but are not limited to:

- Send messages to the participant at specific times during the day.
- Pre-defined answers on the messages. Allowing for more predictable answers.
- Disable the ability for a participant to respond to certain messages.
- Creation of variables that hold specific participant values.
- Definitions on how to make calculations on the created variables.
- Enable calculations on the defined variables and send messages depending on the calculation.

The different use cases and combinations of all the rules makes up a vast list of possible configurations for one single intervention. This helps ensure that whoever wants to use MobileCoach for further research projects, can configure an intervention to their own liking to suit their needs.

An intervention administrator is able to configure interventions via the web application that is available at https://IP_OF_DOCKER_SERVER/MC/admin, the main screen is shown in Figure 2.5. This screen shows a list of created interventions, as well as their intervention status (whilst active, one cannot configure basic intervention properties, such as monitoring days) and monitoring status (whilst active, one cannot configure the flow of the intervention and makes the intervention available to users on the mobile app).

Selecting to edit an intervention reveals several different tabs to make adjustments, as shown in Figure 2.6, which has the "Micro Dialogs" tab open. In this tab, an intervention administrator is able to add new messages to dialogues, and how the responses from
Exploring and Expanding the Functionalities of the MobileCoach Platform

Figure 2.5: MobileCoach web application main intervention screen.

a participant is to be interpreted. The message in the top of the list is sent first, then it sends the messages in descending order. A message can be constructed in many different ways. They can be configured to demand a response from a participant, offer multiple choice answers and be sent as a command (then they are invisible for the participant). An example of how a message can be designed as a command is shown in Figure 2.7, where the show-web command is entered in the text area (the "German:" keyword is to differentiate between those who selected "Deutsch" or "English" as their language of choice). An intervention administrator also has to check the box that specifies that the message is a command, shown by the highlighting in the figure. These are just some of the available options for designing messages. It is also possible to design different set of dialogs. Each dialog can consist of multiple messages and different logic than in other dialogs. In Figure 2.6, the "Onboarding" dialog is shown.

An intervention administrator is able to use a so-called decision point in a given dialog, as shown in the bottom of the "Onboarding" dialog in Figure 2.6. Here, an intervention administrator is able to make a decision on which dialog to jump to next, depending on the calculations that will be done. In Figure 2.8, a decision point has been made, where it will always switch to the "getSteps" dialog. An intervention administrator is also able to let JavaScript code to be run in a decision point, this can be useful for more heavy and specific calculations.
Exploring and Expanding the Functionalities of the MobileCoach Platform

Figure 2.7: An example of how a message is designed as a command. Here, the show-web command is used. This will open up a WebView in the MobileCoach application.

Figure 2.8: MobileCoach web application designing a decision point in a dialog. This particular decision point is designed to always go to the ”getSteps” dialog.

The logic behind the messages created in the ”Micro Dialogs” tab can be configured in the ”Rules” tab, shown in Figure 2.9. In this tab, an administrator is able to specify a wide variety of rules for interventions. They are able to specify which calculation should be done on a daily basis, periodic basis (lets say every hour or so), how to act on unexpected responses from a participant and what to do with the expected responses. A rule can also be used for switching between dialogs. For example, when a participant has gone through the welcome screen, the ”tutorial”, an intervention administrator is able to specify which dialog to start at that point. Another approach could be to make a specific dialog to start every day at a specific time, this dialog could be used for asking a participant how many glasses of water they have drunk during the day.
The rule calculations are often done with the help of variables defined for a specific intervention. These variables can be declared and defined in the "Variables" tab, as shown in Figure 2.10. Each variable has to be specified with a "$" in the beginning of its name. The values of the variables can be read at a later time when logic needs to be checked, like checking if a participant have taken a certain number of drinks on a given day. To do this, an intervention administrator can design a message to expect a number as a response from a participant. This number is to be stored in a variable that can later be used to, for example, change dialog, depending on the rules for that intervention.

2.1.4 Mobile Application

As of September 2019, the MobileCoach team released a mobile application for the MobileCoach platform. This application is developed for both iOS and Android devices with the React Native framework (more on React Native in section 3.1. Prior to this, a participant could only interact with the platform by regular SMS messages. Now with the
mobile application, a participant of an intervention is able to not only save money, but also to have everything accessible in a single environment (as far as the MobileCoach platform is concerned). The mobile application allows a participant to chat with a chat bot and also to have direct contact with their caretaker as well as upload media for the caretaker to take part of. The chat bot can be used to send predefined messages, as well as responses, to and from the participant.

By having a mobile application, many doors with opportunities open up for the MobileCoach platform. It allows for greater monitoring of the participant, which in return helps to determine the health state of a participant. As for this thesis, the ability to read the amount of steps a participant has taken during a day will enable the platform to better understand the health status of a participant as well as open up more doors for future work.

### 2.1.5 Chat Bot

Once a user has gone through the welcome screen of the mobile application, the main screen when opening the application is the chat with the chat bot. This chat bot is operated via the web application that is available at `https://IP_OF_DOCKER_SERVER/MC/admin`, where a set of predefined rules can be configured by an intervention administrator (where the administrator is not required to have any programming knowledge), depending on the given intervention. In certain situations, this static chat bot may not be the optimal way of communicating with a participant. Therefore, there is an option for communicating directly with a caretaker in another tab. This thesis only focuses on the chat bot and the caretaker chat is therefore outside the scope of this thesis.

The configuration of the chat bot consists of manually defining exactly what kind of message the bot is about to send to a participant. The possible answers that a participant can give is usually also predefined, which gives the caretakers more control over the information flow. It is possible to let a participant answer in free text, but those messages may be harder to process for the bot. Also, it may cause issues when none of the possible responses a participant can choose from does not correspond to what the participant would like to respond with.

The chat bot can trigger a various set of code sequences, which gives the caretakers a lot of freedom when it comes to designing their own interventions. This is done by sending a message as a command to, which is invisible for the participant, and triggers specific actions on the mobile application. These commands consists of, but are not limited to - opening a WebView and share a link. The aim of this thesis is to introduce two new set of commands available for the bot to handle. One command will be to prompt a participant to authorize MobileCoach to Google Fit, and the other command will let the bot query for user step data from Google Fit.

### 2.1.6 Chat Messages

The MobileCoach mobile application communicates with the server by the use of messages in the format of JSON objects. Listing 1 displays the layout of a message that is sent from the server to the mobile application. There are some key properties that are of
importance in this message. They are the following:

- **id** - Specifies which sequence of message this is (i.e. the first message is sent with an id of 1, and a response to that particular message also has an id of 1, which increases by the number of messages sent).

- **status** - Specifies where the message was sent from, which in this case has the value "SENT_BY_SERVER". Another value this field can have is "ANSWERED_BY_USER".

- **author** - Tells where this message originated from, which in Listing 1 is the server and therefore has the value "SERVER".

- **type** - Specifies the type of the sent message. In the example showed in the figure, the message is of type "PLAIN". Another possible value this property can have is the "COMMAND" value, which is the message type of interest in this thesis, which indicates that an intervention administrator has selected the message as a command.

- **server-message** - Marks the visible content of the message for the participant, which in this case is "Hello! I am your digital coach.". This is the text that a participant will see in the chat of the mobile application.

- **answer-format** - Specifies what a participant can respond to the message, where the "type"-property specifies which response a participant can give, whereas the "options"-property displays the possible responses from the participant. In the example, a participant is given only one option, "Okay, thank you".

- **expects-answer** - Has a value of true or false, depending on if a participant is expected to answer on the message that was sent, which has a value of true in Listing 1 since the participant has been given answer options defined by the answer-format property.

```
{
  "id": 7,
  "status": "SENT_BY_SERVER",
  "type": "PLAIN",
  "content": "",
  "server-message": "Hello! I am your digital coach.",
  "format": "plain",
  "answer-format": {
    "type": "select-one",
    "options": ["Okay, thank you", "0"]
  },
  "message-timestamp": 1575965542650,
  "expects-answer": true,
  "can-be-cancelled": false,
  "last-modified": 1575965542650,
  "sticky": false,
  "deactivation": false,
  "client-status": "OPEN_QUESTION",
  "client-id": "s-7",
  "client-read": false,
  "author": "SERVER",
  "client-version": 0
}
```

Listing 1: JSON message sent from the server to the mobile application.
In Listing 2, a participant has responded to the message that the server sent in Listing 1. It has a similar structure, containing the same properties, like `id`, `type`, `server-message` and `answer-format`. The properties that have changed, however, are the properties of `status`, `client-status`, `user-message` and `user-text`. The `status` property has changed values to "ANSWERED_BY_USER", which defines that this message has been answered by a user. `client-status` has changed values to "ANSWERED_AND_PROCESSED_BY_SERVER", which indicates that the server has received the message and interpreted the contents. The `user-message` and `user-text` property indicates which option the user decided to answer with and which text was related to that option, respectively.

```
{
    "id": 7,
    "status": "ANSWERED_BY_USER",
    "type": "PLAIN",
    "content": "",
    "server-message": "Hello! I am your digital coach.",
    "format": "plain",
    "answer-format": {
        "type": "select-one",
        "options": ["Okay, thank you", "0"]
    },
    "message-timestamp": 1575981846277,
    "expects-answer": true,
    "can-be-cancelled": false,
    "last-modified": 1575981871132,
    "sticky": false,
    "deactivation": false,
    "client-status": "ANSWERED_AND_PROCESSED_BY_SERVER",
    "client-id": "s-7",
    "client-read": true,
    "author": "SERVER",
    "client-version": 7,
    "fake-timestamp": 1575981847124,
    "user-message": "0",
    "user-text": "Okay, thank you",
    "user-timestamp": 1575981871485
}
```

Listing 2: Structure of the message that is answered by a participant.

A command message is shown in Listing 3, specified by the "COMMAND" value in within the `type` property. The `server-message` property specifies which command is to be executed, which in this case is the "show-web" command. A command message can not be answered by a participant, as shown by the `expects-answer` property. This property is always `false` for command messages.
2.1.7 Security Aspects

As the MobileCoach platform needs to collect many different types of data from a participant, security is a topic that probably most people would take into consideration before using such a platform. For starters, the MobileCoach team, as previously mentioned, is stationed in Switzerland, and is therefore not part of the EU. This means that they have not yet considered the GDPR data laws as their top priority. Another reason for this is that it is usually themselves handling the interventions and can therefore provide the necessary information for all participant’s before the interventions begin.

Despite this, basic security is in place on the platform. This security consists of encrypting the data stored in the database as well as the communication between different components of the ecosystem. Notwithstanding, the MobileCoach team can not guarantee sufficient security for the platform since it is up to the designers of a given intervention on how the security is to be handled, since it is an open source platform and changes to the code are not particularly difficult. It is like a car manufacturer can not guarantee that their car will not crash. They have implemented specific techniques to avoid crashes, but they can not be prevented if the driver drives in a reckless manner.

As of writing this thesis, the MobileCoach platform supports identification of participant’s by handing out QR-codes to participant’s to scan to get a hold of where to install the application. Participant’s are also provided with a ”patient code”, which is used for identifying each participant for the caretakers. This is a sufficient method for the time being since the MobileCoach team usually makes the interventions themselves.

2.2 Related Work

The MobileCoach platform is not the first one to bring health behavioral interventions in digital form. Other platforms with similar goals to MobileCoach include ginger.io13.

13https://www.ginger.io/
LifeGuide\textsuperscript{14}, Minddistrict\textsuperscript{15} among others. But every platform is different by bringing their own pros and cons to the table, and MobileCoach is striving for bringing their best approach on digital health interventions.

Ginger.io offers a user to chat directly with one of their coaches when they seek treatment via their own mobile application. These sessions may be done via both chat and video. Outside of this, Ginger.io offers self-taught activities that focuses on certain areas, such a stress, anxiety and breathing, among others.

LifeGuide have gathered over 30 different digital health interventions and considered what was good about them and constructed a Person-Based Approach that helps form interventions that are more accessible, engaging and cost-effective.

Minddistrict is available on the web as well as mobile devices and offers a service of individual recovery. This treatment is offered via different types of modules, where each module represents a given personal trait that a client may want to improve on.

MAX\textsuperscript{16}[15], a digital health literacy coach for children with asthma. This platform is an extension of the MobileCoach platform with a specific focus of 10-15 year old children with asthma. MAX allows for collaboration between healthcare providers, the participant and their parents. Since MAX is an extension of the MobileCoach platform, it also implements the support for identifying participant’s by a given QR-code as well as a participant code.

The Google Fit API is widely known among developers when it comes to recording fitness activity from mobile users. Google has been very eager to facilitate the integration process of Google Fit as much as possible whilst developing other applications. Which has lead to that many other applications have adopted this API. For example, the diet app Lifesum\textsuperscript{17} uses Google Fit to record how many steps a user has taken in order to determine how many calories they have burned for the day, and Pokémon GO\textsuperscript{18} uses Google Fit to record the amount of steps a player has taken in the past week to give them rewards. Pokémon GO has also contributed to increasing the activity of US citizens by an average of 1473 steps[16] on a daily basis.

\textsuperscript{14}https://www.lifeguideonline.org/
\textsuperscript{15}https://www.minddistrict.com/ehealth-platform
\textsuperscript{16}https://www.c4dhi.org/projects/health-literacy-children-asthma/
\textsuperscript{17}https://developer.android.com/stories/apps/lifesum-health
\textsuperscript{18}https://pokemongolive.com/post/adventure-sync
Chapter 3
Theory

This section brings up relevant theory that is useful in order to advance towards the goals of this thesis. It brings up the React Native framework and the Google Fit API. It will also cover user authentication with OAuth 2.0 and why it is necessary. It will additionally be explained how OAuth 2.0 can allow for an application to communicate with an API for accessing user data (with the user’s consent). Lastly, it will be described how this methodology can be used to make the MobileCoach mobile application communicate with the Google Fit API.

3.1 React Native

Developed by Facebook, React Native\(^1\) is an open source JavaScript framework used to develop applications for desktop and mobile devices (both iOS and Android). React Native builds on the same principles as ReactJS\(^2\), which is a JavaScript library, also developed by Facebook. The difference between ReactJS and React Native is that ReactJS is designed to develop user interfaces for specific applications, whereas React Native is used for building cross-platform applications.

The purpose of React Native is to avoid the complicated and cumbersome process of developing two different apps for Android and iOS, which takes more time as well as require more experience and expertise since each application for respective OS are developed in different programming languages. When developing an application in React Native, an environment will be built for both iOS and Android where users on the different OS’s will have a similar experience, but sometimes it may require some further tweaking due to different behaviours of the operating systems\(^{17}\). React Native applications are written in JSX, a mixture of JavaScript and XML syntax. This gives a major advantage for many programmers, especially web developers, since JavaScript is a broadly used programming language suitable for many different applications.

The MobileCoach mobile application has been developed in React Native and is therefore highly relevant for this thesis since it is necessary to continue development in it in order to integrate the platform with Google Fit. React Native has also been used to develop many popular mobile applications, such as Instagram, Tesla, Skype, SoundCloud,

---
\(^1\)https://facebook.github.io/react-native/
\(^2\)https://reactjs.org/
among others\textsuperscript{3}.

### 3.2 Google Fit

As the name suggests, Google Fit\textsuperscript{4} is being developed by Google and was initially released in 2014. Google Fit is a health-tracking platform for both Android and iOS devices that collect user activity via a wearable or mobile device. Some of the data that Google Fit is able to record is steps, activities and the amount of calories burned during certain activity. Steps is the data of interest in this thesis. The development of Google Fit has been done in cooperation between Google, WHO and the American Heart Association\textsuperscript{5}, resulting in a platform that strives for improving the health of the user. What makes Google Fit relevant for this thesis is that it provides developers the ability to record the amount of steps a user has taken during a given time interval.

Google Fit offers a well-documented SDK\textsuperscript{6} for developers that helps facilitate the integration of the Google Fit API into their own applications. The Google Fit API consists of multiple, more specific, API’s. The one of interest for this thesis is the History API, which is used for performing bulk operations on fitness data, such as adding, deleting and reading. This API will be used when the MobileCoach application is querying for the amount of steps taken by a participant during the specific day.

The developer is able to make their app take use of all data that Google Fit has stored for a particular user, with the user’s consent of course. In order for a developer to use Google Fit for their own purposes, an OAuth 2.0 client ID needs to be in place. A user is able to specify which data that a given app will get access to on their Google Fit account, where they are also able to remove this access by visiting the permissions site for their Google account\textsuperscript{7}.

### 3.3 User Authentication with OAuth 2.0

In modern applications, it is very common to use some kind of user authentication when a developer wants their app to access certain information about a user on one of their other accounts (for example, Facebook, Google, Twitter). For this, OAuth\textsuperscript{8} was created and is now an industry standard that lets applications authenticate users without the application having to know the password of the user, or even know which user the authentication is to be made for. The most widely used version of OAuth is version 2.0.

In the early 2000’s, when a given application wanted access to specific information that resided in a given user’s account from another corporation, the user would have to type in their username as well as their password for that account in order for that application to have access to it. This was a terrible idea, not only because the user would have to...\textsuperscript{9}

---

\textsuperscript{3}https://facebook.github.io/react-native/showcase
\textsuperscript{4}https://www.google.com/fit/
\textsuperscript{5}https://www.heart.org/
\textsuperscript{6}https://developers.google.com/fit/
\textsuperscript{7}https://myaccount.google.com/permissions
\textsuperscript{8}https://www.oauth.com/
hand over both the username and password, but the application would also be granted full control over that account. This was totally unnecessary when a certain application only needed access to the calendar, but got access to email and contacts as well as having the ability to send and delete email messages. OAuth was created to help solve this problem, letting an application access a certain part of user’s accounts, without knowing the password for those accounts.

The process of using OAuth requires a few different components in order for it to work as intended. The included components are and play the following roles:

- **The user** - The user that is to allow a given application to access certain information about them that is stored on one of their other accounts.

- **The client** - Used by the user to interact with the application, could be a computer or a smartphone.

- **The application** - The component that wants to access certain information about the user. Usually shows a dialog to the user where it asks for the user’s consent if the application is allowed to access the data.

- **OAuth server** - A server that generates OAuth tokens. These tokens work as proof for the application to show for the API that it has been allowed to access user information.

- **API** - The resource to provide user information to the application. It checks whether the OAuth token provided by the application is valid. If it is, the application can access the requested data.

These components and the communication between them are visualized in Figure 3.1. The user interacts with the client, which in return communicates with a given application. The application wants to get access to certain information about the user on one of their accounts from another association, like contacts on their email account. The application itself can not get access to this information by talking to the API directly, it first needs confirmation from the user to access the desired data. Therefore, it asks the user to give the application access to the information by letting the user confirm for the OAuth server that the application indeed is authorized to access the given data about them. The user tells the OAuth server that a given application is allowed to access their data. The OAuth server then returns a temporary code for the application to use to display for the API. The client sends this temporary code back to the application. Now the application can contact the OAuth server and provide the temporary code, and its secret, if it has one. This is a proof of that the temporary code the OAuth server handed out comes from the right user. Hence, an access token is generated and sent to the application. The application can now use this access token to communicate with the API and prove that it is allowed to access the information of the user.
A great metaphor for the OAuth authentication process was described by Aaron Parecki during his lecture on API Days\(^9\), where he discusses how to secure API’s with OAuth 2.0. The whole process can be seen as when checking in to a hotel. In the reception, the receptionist hands over a keycard to the customer. Here the receptionist is the OAuth server, the customer is the user and the keycard is the access token. This access token can then be used to open a specific room at the hotel. The unlocking of the door symbolizes access to information in the API. The door itself does not care who put the keycard in, but it knows that the keycard has access to the room. Just as the process shown in Figure 3.1, any personal information of the user is never exchanged.

As for the case of this thesis, it is the MobileCoach Android application that is supposed to communicate with the Google Fit API, the flow of retrieving and OAuth access token can be shown as in Figure 3.2. The OAuth server component is now a part of the API component (Google Fit) since it is more sufficient for larger API’s to have integrated OAuth servers.

---

\(^9\)https://speakerdeck.com/aaronpk/securing-your-apis-with-oauth-2-dot-0
Figure 3.2: Intentional flow of authorizing the MobileCoach Android application to access a participant’s steps taken stored in Google Fit.
Chapter 4

Method

This section will bring up the method specification of the thesis, where it will be described what has to be done, methodologically. After that, the tools that have been used for this thesis will be brought up, as well as discuss their roles. Later, it will be described what has to be done in order to achieve the final results, which consists of acquiring an OAuth client ID, authorizing MobileCoach to access Google Fit data, extract that data (steps specifically) and to integrate this functionality into the chat bot. And lastly it will be specified how the results will be validated to help confirm that this thesis indeed hit the intended goals.

4.1 Method Specification

Before the integration of Google Fit into MobileCoach can be done, an OAuth client ID is required in order to communicate with the API. This client ID is necessary in order to advance the progression of the thesis by allowing a participant to authorize the MobileCoach mobile application to retrieve the steps taken by that participant.

It will first be tested if a user can authenticate the MobileCoach application to access Google Fit, and later if it can fetch user step data. When these two criteria are met, the two functionalities will be implemented into the chat bot.

4.2 Selection of Tools

For the development work for this thesis, the text editors Visual Studio Code\(^1\)(VSC) and Android Studio\(^2\)(AS) were used. Both of these text editors provide great development environments, where VSC is an all-round editor while AS, as the name suggests, has Android development as focus.

The development also requires that the integrated functionality can run on an Android device. For this, AS offers a tool for emulating an Android device, called Android Virtual Device (AVD). A developer is free to select from a wide variety of Google devices to emulate, as well as select which version of Android to run on the selected device. During this thesis, an emulated Pixel 3 running Android Oreo (version 8.0.0) was used. The

\(^1\)https://code.visualstudio.com/
\(^2\)https://developer.android.com/studio
AVD provided in AS was chosen because it is the most popular emulator for Android development and is also the same emulator that the MobileCoach team has used during their development. Figure 4.1 shows the emulated Pixel 3 running the MobileCoach application. When the application seemed stable on the emulated Pixel 3, the code was later loaded into a physical Android device. This device consisted of a Samsung Galaxy S7 edge (SM-G935F), also running Android Oreo (version 8.0.0). As of writing this thesis, Android 10 is the latest version of Android and would therefore be the optimal version to run on the emulator and physical device. But version 8.0.0 was the latest version available on a physical Android device during the work on this thesis, hence the reasoning for using that specific version.

By default, when building a React Native application, it is automatically installed on a running AVD. Allowance of development software to be installed on a physical Android device is not enabled by default. One can enable this by enabling USB-debugging for their Android device. This is done by going to the settings of the device and locate the Build Number section. Tapping on this section 7 times enables "Developer mode" for the device. A user can now find a "Developer Options" section at the bottom of the settings menu. In this menu, one can enable USB-debugging. Now a developer is allowed to build their developed software on their physical Android device if it is connected via USB.

4.3 Method Description

4.3.1 Acquiring OAuth Client ID

As mentioned in section 3.3, the Google Fit API has merged the OAuth server with the API server. This allows Google to provide their own service to hand out OAuth client ID’s, which is provided on their API console website. In order to retrieve such an ID, some fields are required. First, the application that wishes to use OAuth has to be registered on Google’s API console website. Second, the API that will be used for the given application.

3https://console.developers.google.com/getting-started
needs to be specified (in this case the Fitness API) as well as where this API will be called from (Android), and additionally, what kind of data will be accessed. Thirdly, a name of the application needs to be specified, this field is just for private use to help distinguish different projects. Another required field is to provide the SHA-1 signing-certificate fingerprint as well as the package name that is used in the Android code of the application (AndroidManifest.xml file). Once this is done, an OAuth client ID is automatically acquired and the application is ready to authenticate itself using OAuth. This is the method of choice since Google themselves hand out OAuth client ID’s and it is therefore more convenient since this thesis will be utilizing the Google Fit API.

### 4.3.2 Authorizing MobileCoach to Access Google Fit Steps Data

Once the OAuth client ID is set up, the mobile application is theoretically ready to communicate with the Google Fit API (the flow as shown in figure 3.2). But additional software has to be developed in order for this flow to take place.

A part of the required software consists of installing the npm package react-native-google-fit\(^4\). This package for React Native provides a sufficient set of functions in order to let a user authenticate a given app to Google Fit as well as query data about that particular user. One of the implemented interfaces in the package is the GoogleSignInApi\(^5\) interface, which provides the familiar Google login-screen for the users of the application. This package is not from Google themselves, this is because that the documentation provided for the Google Fit API only takes up Java examples (for Android development), and since this thesis works with the React Native framework, JavaScript is the language to go for.

### 4.3.3 Querying Step Data of A Participant

When calling the function for authorizing the application to Google Fit, it takes a scope as an argument. A scope is a specification of which data is to be accessed within an API. For this thesis, it is only of interest to read the fitness activity, specifically the steps, so therefore only the scope https://www.googleapis.com/auth/fitness.activity.read will be used. Full specifications on which data is accessible via this scope is shown on the Google Fit API site\(^6\). After the scope has been selected and the user has given permission to access their account data, it is time to start recording the steps taken, even without the user having to have Google Fit installed on their device. At demand, the step data for a given user may be queried via the Google Fit API by specifying the time span that is of interest.

Listing 4 displays how one can make use of the react-native-google-fit npm package and use its functions to authorize (with a scope of only reading data) as well as query for steps. The options variable specified in the getDailyStepCountSamples-method is a JSON object, which holds the latest and earliest date to fetch step data from. Processing of the response from Google Fit (which will be stored in the response variable) will be done in the arrow function.

---

\(^4\)https://www.npmjs.com/package/react-native-google-fit

\(^5\)https://developers.google.com/android/reference/com/google/android/gms/auth/api/signin/GoogleSignInApi

\(^6\)https://developers.google.com/fit/rest/v1/authorization
import GoogleFit, { Scopes } from 'react-native-google-fit';
GoogleFit.authorize(Scopes.FITNESS_ACTIVITY_READ);
GoogleFit.getDailyStepCountSamples(options, (error, response) => {});

Listing 4: Code snippet showcasing a possible implementation of how to authorize a participant as well as query for their steps.

4.3.4 Integrating Authorization and Step Retrieval For the Chat Bot

The results achieved from the methodology described in subsection 4.3.2 and subsection 4.3.3 will be used to implement two additional commands for the chat bot. The first command will be used to let a participant authorize MobileCoach to read the step data that Google Fit possesses. The second command will be used to actually query that data. The data retrieved from this command is to be sent to the server client, where it will be processed to determine if they are on the right track to improving their step total. These two commands will be available for an intervention administrator to use when designing a message to a participant, if a given intervention is in need of participant step data.

4.4 Validation of Results

Validating that the intended results have been met can be measured in different steps. Firstly, it needs to be confirmed whether the user can indeed authorize the MobileCoach application to query for fitness information of that particular user. As a follow-up, it will be checked if the MobileCoach application can query data and get a response. These two steps will be made via the chat bot.

Lastly, it needs to be verified if the steps retrieved for the MobileCoach platform indeed matches up with what Google Fit has stored, to ensure that it is the correct data. This will be done by having the standalone Google Fit application installed on the same device that runs the mobile application of MobileCoach. From here, one can check if the number of steps displayed in the Google Fit application matches up with the number of steps that the MobileCoach application retrieved via the Google Fit API. If the numbers match, one can be certain that the information has been queried correctly.
Chapter 5

Results

The result of the work from this thesis has lead to that the MobileCoach platform is able to let participant’s authenticate MobileCoach to Google Fit to read fitness data in forms of steps. These steps are also able to be sent to the MobileCoach server, enabling interventions to act accordingly. The integration of Google Fit into the MobileCoach platform is shown in Figure 5.1, where the MobileCoach mobile application is the only component in the MobileCoach platform that communicates with the Google Fit API.

The results enables for an intervention administrator to utilize two commands whilst designing an intervention. The same methodology as shown in Figure 2.7 is used in order to use these commands. The two commands that has been implemented are the following:

- **authenticate-google-fit** - When this command is sent to a participant, it opens up a dialog where the participant is able to authorize the MobileCoach application to access Google Fit data.

- **get-steps-google-fit** - If a participant has authorized MobileCoach to Google Fit, this command fetches the steps that particular participant has taken during the current day. The number of steps is stored in a variable on the MobileCoach server.

![Figure 5.1: Visual representation of how the architecture of the MobileCoach looks like based on the results of this thesis.](image-url)
Figure 5.2 displays how an intervention administrator can use one of the implemented commands whilst creating a dialog message for a given intervention. An intervention administrator only have to alter two elements within a message to be able to use the authentication command. First, the administrator have to type in the phrase "authenticate-google-fit" in the text box, as shown in the top of Figure 5.2. The “German” keyword only distinguishes how the message should look like for participant’s that chose Deutsch as their language. Currently the platform does not distinguish participant’s who chooses Deutsch or English. The second thing the administrator has to do is to mark the message as a command, which is shown in the middle of the figure. When a participant is to be prompted to authenticate MobileCoach to access Google Fit data, a command message is sent as shown in Listing 5. Here it is specified that the message is of type “COMMAND” and that the specific command is "authenticate-google-fit” as specified by the “server-message” field. This triggers what is shown in Figure 5.3.

When an intervention administrator has set up the command for authorization, the chat bot will send the command to a participant. The authentication process is shown in Figure 5.3. Here, two additional messages have been created prior to the authentication message, as shown in Figure 5.3a, which gives the interaction with the bot a more natural impression. Once a participant presses the "Sign in to Google Fit” dialog, the command that the administrator created is sent by the chat bot. This opens up a dialog where a participant is able to choose which one of their accounts is to authenticate the MobileCoach mobile application to access their Google Fit data (Figure 5.3b). The participant is guided through three pages of text, where each page corresponds to different areas of Google Fit and what data each area holds. Lastly, as shown in Figure 5.3c, a participant is given an overview of which data MobileCoach is to be allowed to access. A participant can easily filter out the areas he or she seems irrelevant that MobileCoach should have access to. Depending on
Listing 5: Structure of a command message that triggers the prompt for a participant to authenticate MobileCoach to Google Fit

which data a participant authorized the MobileCoach application to access, MobileCoach may now query for steps of that participant.

Figure 5.3: Visual representation of the authentication process. The chat bot first sends a message that will open a dialog, where a participant can choose which account that is to authenticate MobileCoach to access Google Fit data. Lastly, a participant can confirm which data MobileCoach is allowed to access.

The second command that an intervention administrator now can use that has been implemented is the "get-steps-google-fit" command, as shown in Figure 5.4. This command is used by the intervention administrator in the same way as the "authenticate-google-fit" command, despite now it requires two variables, the first variable is the earliest day that steps are to get fetched from, and the second variable is the latest day to fetch steps from. These two variables are intended for an intervention administrator to calculate us-
ing JavaScript code in a decision point. Now when the command is sent to a participant, the step data is collected automatically, compared to the "authenticate-google-fit" command where a participant had to sing in to their Google account. This interaction with the chat bot can be made more intuitive by letting the bot first give a heads-up and let a participant confirm that MobileCoach can get their step data. But an intervention administrator could design a given intervention to get participant steps in the background (as long as the application is active), without the participant having to make any interaction with the chat bot. This chat flow can be seen in Figure 5.5. In Figure 5.5a, it is shown how a participant specifies their daily step goal (5000 in this case), and the participant is prompted to press the button which will make MobileCoach query for their steps taken for the current day. The follow-up chat is shown in Figure 5.5b, where the chat bot says that the total number of steps the participant has taken today is 531, which is lesser than their goal of 5000 steps. This triggers a decision point (in the web application) made for the current dialog, and it switches to a dialog that is designed for when the current step count is lesser than the goal of a participant. In this case, the bot sends messages with the intent of encouraging the participant to make a push for it.

Figure 5.4: Screenshot of how an intervention administrator can use the chat bot to get step data from a participant.
Figure 5.5: Display of the process of where steps are asked to be fetched and once they got fetched, respectively.

Listing 6: The command message for retrieving steps between two dates.

```json
{
  "id": 17,
  "status": "SENT_BY_SERVER",
  "type": "COMMAND",
  "content": "",
  "server-message": "get-steps-google-fit
2019-12-09T20:39:59.026Z
2019-12-10T20:39:59.049Z",
  "format": "plain",
  "message-timestamp": 1576010402019,
  "expects-answer": false,
  "can-be-cancelled": false,
  "last-modified": 1576010402019,
  "sticky": false,
  "deactivation": false,
  "client-status": "RECEIVED",
  "client-id": "s-17",
  "client-read": false,
  "author": "SERVER",
  "client-version": 0
}
```

What happens when a participant presses the button showed in Figure 5.5 is that a command message is sent as shown in Listing 6 from the server, where the two specific dates are sent with the command "get-steps-google-fit". These two dates mark the start and end date, respectively. These dates have to follow the ISO 8061 standard¹.

Verification of the achieved results consisted of looking if the amount of steps the MobileCoach platform gathered was equal to the desired amount.

¹https://www.iso.org/iso-8601-date-and-time-format.html
Coach platform queried agrees with the steps displayed for the corresponding day in the Google Fit application. This is shown in Figure 5.6a, where the steps are marked inside a red circle. This can be compared to the steps retrieved in Figure 5.6b, where the retrieved steps via MobileCoach is also marked in a red circle. It is shown that these two values are the same, which means that the integration was done properly.

Figure 5.6: A comparison of the steps a participant has taken according to the MobileCoach application and the Google Fit application.
Chapter 6

Discussion

The achieved results corresponds to the goals that were set out to be achieved for this thesis. A fundamental understanding of the MobileCoach platform was acquired in order for the integration of Google Fit into the platform to be possible. A participant is now able to authorize the MobileCoach application to access their fitness data stored in Google Fit. This allows for the MobileCoach application to query for step data and the chat bot can give different types of responses depending on how many steps a participant has taken.

The final implementation was done in such a way that each time participant steps are fetched, the last reading of the steps is overwritten. This applies for both the mobile and the web application. A benefit from this is that the app does not become more "fat" over time, i.e. storing a bunch of step readings that may not ever be used again. However, there might come such a situation where one would want to read previous step readings, but this is not possible because of the way it was implemented and can therefore be seen as a flaw of the current implementation.

After this thesis, the current architecture of the MobileCoach platform corresponds to what was shown in Figure 5.1. It is believed that the decision of only being able to communicate with the Google Fit API through the mobile application was the right one. This due to the nature of the OAuth process, which was described in section 3.3, which requires that the application directly needs to communicate with the desired API. During the course of this thesis, it was discussed whether it would be possible to let the MobileCoach server communicate with the Google Fit API instead, but conclusions where drawn that this could not be the case. It is also believed that the achieved implementation for this thesis could not have been done in many other ways. This due to how the MobileCoach platform is designed on how the mobile application communicates with the server.

The chosen validation method for checking if the right amount of steps was collected is safe to say was the best approach possible. This because since both values comes from the same source, it would only be logical if the two values would be the same in the MobileCoach application and in the Google Fit application. If however they would not be the same, one would be certain that something went wrong whilst collecting step data with MobileCoach.
6.1 Implemented Chat Bot Commands

The final work enables an intervention administrator to easily utilize the two added commands for the chat bot. The administrator only has to type in "authenticate-google-fit" in the text field of a message, and mark it as a command in order for it to ask a participant for authorization against Google Fit. And by querying for steps, the administrator only has to construct another message that instead has "get-steps-google-fit" command as well as two variables that specifies the start and end date of a given time interval. These two variables would have to be calculated in a decision point. From the perspective of an administrator, this is really easy to use when designing an intervention.

However, it is believed that the design of interventions utilizing commands could become even more efficient for intervention administrators whilst designing them. Currently when designing a command message, there are no options/preview of which commands are available to use. In this sense, the available commands would become more recognizable. Instead, an administrator needs to be aware of the different commands they can use. It would also be even more beneficial if a short description of the different commands could be seen in the MobileCoach web application.

6.2 User experience

From a user perspective, the authorization process is believed to be smooth and of the same experience as on every other application that requires the user to sign in to their Google profile. This conclusion can be drawn since the npm package used in this thesis implements the GoogleSignInApi interface, which is used as a standard when signing in to Google profiles. Once a participant has authorized MobileCoach to Google Fit, they do not have to do anything in order for MobileCoach to get their step data. This aspect could be seen as a lesser good user experience given that an intervention can be designed in such a way that a participant is not aware of when MobileCoach will collect their step data. This issue arises due to the way that the intervention system in the platform has been implemented and the work achieved in this thesis could not have altered that functionality.

However, there are some aspects that are believed to have been able to improve the user experience. One aspect would be to visually represent the goal of steps and amount of steps a participant has taken. For example, a progress bar would be sufficient, where the bar would fill up more and more the closer a participant gets to their goal. Additionally, it is currently not possible for participant’s to decide when MobileCoach should check their daily steps. This functionality would help make the interventions even more personalized since not all individuals have the same circadian rhythm. One solution to this would be to display the number of steps MobileCoach has registered during the day, where a participant could also be able to specify when MobileCoach should check their steps.

As for user integrity, which is of highest interest in today’s society and is no longer only of interest for developers, but also for users. In this thesis, the implementation was managed to be done in such a way so that a user never have to expose their identity while authorizing MobileCoach to Google Fit, thanks to OAuth 2.0, as described in section 3.3. By ensuring the integrity of the user, the application will appear more attractive to current and new users.
6.3 What Could Have Been Done Differently?

Given that the implementation of Google Fit was done in the React Native framework, it would probably have been possible to also integrate the functionality for iPhone devices. However, it is not certain how many adjustments, if any, would have to be made to the code in order for it to be running properly on an iOS device, since the final implementation utilized Google Fit, which might not be as optimized on iOS compared to Android. The reasoning for why this did not happen was that this thesis initially focused on the Android OS, and did therefore not prepare the necessary material for development on iOS before it was too late. The finished implementation could have also been tested with an Android smartwatch, but was not available during this thesis.

The initial months of the thesis was spent trying to understand the MobileCoach platform based on public information such as published papers, the MobileCoach website and the MobileCoach BitBucket repository. There was not much progress made during these first months, but the work was sped up once contact with the MobileCoach team was made. It would therefore have saved a lot of time if the MobileCoach team would have been approached from the beginning of this thesis, instead of trying to make the integration without their help.

The work that was done in this thesis was initially aimed towards the belief that a participant should be able to enter their own step goal, and this goal would be compared with the number of steps the participant has taken during the day. Another approach to this could have been to let a participant enter their current daily activity. Based on this, and some medical research, the participant would be recommended which step goal would be suitable for someone on their activity level, but they would have the opportunity to upper/lower this goal.

The current integration of getting participant steps requires that the mobile application is connected to a MobileCoach server. The reasoning for this was that the functionality of retrieving steps could be available for the chat bot to examine. Another way to do this, without having contact with the chat bot for retrieving steps, could be to use a Toast message that is often used for displaying brief, auto-expiring windows of information to a user. In this case one could use a Toast message to display the amount of steps the platform received from Google Fit.

The period of which the MobileCoach platform retrieves steps from is adapted for only getting the steps a participant has taken on the current day. The reasoning for this was that it would be of highest interest to know steps on the current day, rather than any other day. The way of which the implementation of fetching step data was made, and by the way the platform is designed, changing this could be a struggle.

6.4 Future Work

For future work, the MobileCoach team themselves would like to implement a functionality for measuring the intensity of the steps a participant has taken, not only the number of...
steps. This thesis has taken the first step in that direction for the MobileCoach platform. Step intensity would be of great benefit for the caretakers of MobileCoach because of the fact that an individual who lives a more active lifestyle would not feel that a given amount of steps would be as fatiguing as a more inactive individual would. This would help the MobileCoach platform to make better decisions for a participant since it would contribute to more data points.

Currently there is no way for a user to authenticate themselves on the MobileCoach platform, except using the "patient code" method described in subsection 2.1.7. This approach is not scalable when the platform starts spreading more globally. Instead, the intervention could start with that the user would go through a process where a user could authenticate themselves with a service similar to the Swedish BankID\(^2\). BankID is an electronic identification system where an individual enters their personal code number into a third-party service, and then they have to enter their password for the BankID system. Why this kind of service would be desired for a platform such as MobileCoach would be to link a participant with their identity in a secure manner. Data collected from the MobileCoach platform could be stored in an archive that is shared with caretakers in a hospital, which would enable for faster hospital treatments if the caretakers had immediate access to the data that the MobileCoach platform would have collected previously instead of spending precious time examining the participant for symptoms.

Today when an intervention administrator designs an intervention, they have to go through and create a lot of different messages and decision points in order for participant’s to have a good user experience. This could be improved by making the chat bot more intelligent. With the help of machine learning, the chat bot could instead learn how to identify patterns that are iconic for participant’s who, lets say, are inactive, and which methods are the most effective in helping them become more active by walking more. The chat bot could also learn how to interact with the participants in a more human-like way by studying thousands of different chat sessions it will have with participants. There are a lot of possibilities in this aspect, but it would require lots of data. By implementing machine learning into the chat bot, the MobileCoach platform would come closer to their objective of offering a more cost-efficient way of creating and maintaining health interventions.

Chapter 7

Conclusions

The purpose of this thesis was to integrate the MobileCoach platform with Google Fit. It was achieved by first acquiring an OAuth client ID for the mobile application. The mobile application was modified to use that ID to let a participant authorize the MobileCoach mobile application to access their Google Fit step data. The step data could later be queried from Google Fit and stored in the MobileCoach server. The work from this thesis enabled MobileCoach to query step data from Google Fit. But it also implemented the opportunity to query for other health-related data as well. The additional data could consist of burned calories during activities, such as biking and hiking.

From the results of this thesis, it has opened up for future work. The MobileCoach team would like to measure the intensity of the steps a participant has taken. And this thesis enables the first part of that functionality. This is relevant because only measuring the amount of steps an individual has taken does not necessarily translate to how healthy they are. For example, the same amount of steps for a obese individual may be more intense compared to a more fit individual. What was achieved in this thesis could be used as a good foundation for implementing the ability to record step intensity from a participant.

To conclude, results from this work can enable for more efficient DHIs on the MobileCoach platform whilst also opening up doors for future work of measuring step intensity of a participant, making DHIs even more efficient and effective.
Bibliography


Jonas Fockstedt

Ema Krcic