User-controlled Privacy for Personal Mobile Data

by

Sharon Myrtle Paradesi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Engineer in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Author .....
Department of Electrical Engineering and Computer Science
August 15, 2014

Certified by .....
Lalana Kagal
Principal Research Scientist, Computer Science and Artificial Intelligence Lab
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Theses

User-controlled Privacy for Personal Mobile Data

by

Sharon Myrtle Paradesi

Submitted to the Department of Electrical Engineering and
Computer Science
on August 15, 2014, in partial fulfillment of the
requirements for the degree of
Engineer in Computer Science

Abstract

Smartphones collect a wide range of sensor data, ranging from the basic, such as location, accelerometer, and Bluetooth, to the more advanced, such as heart rate. Mobile apps on the Android and iOS platforms provide users with "all-or-nothing" controls during installation to get permission for data collection and use. Users have to either agree to have the app collect and use all the requested data or not use the app at all. This is slowly changing with the iOS framework, which now allows users to turn off location sharing with specific apps even after installation.

MIT Living Lab platform is a mobile app development platform that uses openPDS to provide MIT users with personal data stores but currently lacks user controls for privacy. This thesis presents PrivacyMate, a suite of tools for MIT Living Labs that provide user-controllable privacy mechanisms for mobile apps. PrivacyMate aims to enable users to maintain better control over their mobile personal data. It extends the model of iOS and allows users to select or deselect various types of data (more than just location information) for collection and use by apps. Users can also provide temporal and spatial specifications to indicate a context in which they are comfortable sharing their data with certain apps. We incorporate the privacy mechanisms offered by PrivacyMate into two mobile apps built on the MIT Living Lab platform: ScheduleME and MIT-FIT. ScheduleME enables users to schedule meetings without disclosing either their locations or points of interest. MIT-FIT enables users to track personal and aggregate high-activity regions and times, as well as view personalized fitness-related event recommendations. The MIT Living Lab team is planning to eventually deploy PrivacyMate and MIT-FIT to the entire MIT community.

Thesis Supervisor: Lalana Kagal
Title: Principal Research Scientist, Computer Science and Artificial Intelligence Lab

Acknowledgments

Completing the work for this degree in one year was no easy feat and I am thankful to a number of people, both at MIT and outside, for supporting me through this endeavor.

Dr. Lalana Kagal, my research supervisor, has been extremely understanding and considerate throughout the process of this degree and research. I greatly enjoyed working with her and appreciated the technical expertise and insight she provided during the course of this thesis. I sincerely thank Prof. Samuel Madden and Elizabeth Bruce from the MIT Big Data Initiative at CSAIL (bigdata@CSAIL) for funding my research and for giving valuable feedback. I thank Brian Sweatt for mentoring me beyond just providing technical help during the implementation. I am thankful for his help and guidance that enabled me to grow into a better software developer. Along the same lines, I thank Dr. Ilaria Liccardi for her help in writing technical papers and for stimulating scientific thought. Myra Hope Eskridge and Laura Watts from the IS&T Department and Albert Carter provided valuable development guidance and help with the UI/UX design for which I am thankful. I thank Oshani Seneviratne, Fuming Shih, Daniela Miao, Andrei Sambra and other members of the Decentralized Information Group (DIG) for their input over the course of this project. Marisol Diaz (from DIG) and Susana Kevorkova (from bigdata@CSAIL) have been very helpful in coordinating presentations, roundtable discussions, and general support. In addition to Dr. Lalana Kagal, I am indebted to Prof. Hal Abelson, Prof. Leslie Kolodziejski, Janet Fischer, and Prof. Dina Katabi for their patience with me and for their guidance regarding the degree options after my Masters degree. Outside of CSAIL, I am thankful to Emily and Marcus Gibson, Michelle Chang, Becky Mieloszyk, Sunday Trevino, Kimberlee Thorburn, Melanie Vaughn, and friends from the Cambridgeport Baptist Church for their support, prayers and encouragement.

Last, but not the least, I am greatly thankful to my family for showing me unconditional love, support and encouragement. My father, Suresh Paradesi, mother, Dr. Mary Gollapalli, nephews, sister-in-law and brother, Matthias, Emmanuel, Esther, and Martin Paradesi have been a great blessing to me. Above all, I am humbly thankful to my God for His grace, mercy, and love.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Smartphones and smart devices collect a vast array of sensor data. This data ranges from the obvious (location, accelerometer, Bluetooth, and so on), to the more advanced, such as heart rate. This chapter presents the shortcomings of the access control mechanisms commonly available for such sensor data on smartphones. It explains the need for and describes a four-pronged privacy mechanism for users' personal data stores (PDS) that is user-controlled, called PrivacyMate. PrivacyMate is developed within the MIT Living Lab project, led by the MIT Bigdata@CSAIL initiative. The chapter concludes with an overview of the rest of the thesis.

1.1 Motivation

The sensor data collected from smartphones can be used to infer health, social, and behavioral patterns of users. For example, research has shown that location information can be used to discover different types of personal behaviors and details, such as activity patterns [12] [17], profile behaviors [6], and likes and dislikes [18]. Though such data analyses enable service providers to create personalized services for the user, they present some serious risks to privacy.

Researchers have discovered that only four spatio-temporal data points are sufficient to uniquely identify individuals in a set of de-identified data [5]. However, people are often unaware of the nature and extent to which their information is collected and used since the

apps installed on smartphones can silently collect data and send it away from the phone even when the device is idle [19]. Further, in most commercial location-based services, users do not have the ability to control (modify, permanently delete, or limit the use of) their data as these app and device providers collect data and store them on their own servers and could potentially share them with third parties. Figure 1-1 shows the data collection, storage, and retrieval model on most commercial mobile apps.

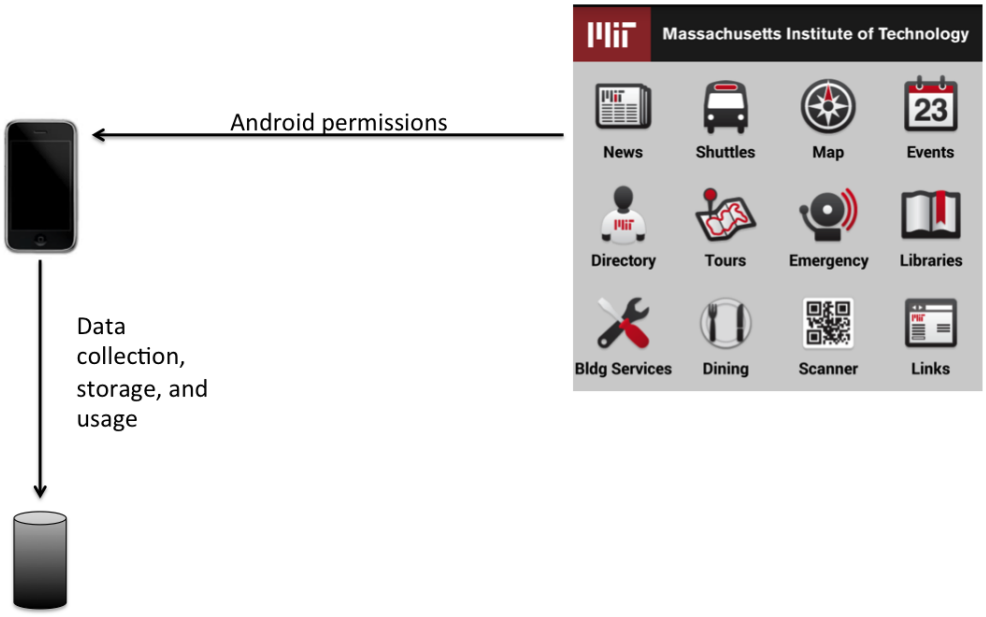

Figure 1-1: *Current access control models in mobile devices are "all-or-nothing". Users must give an app permission to access all the data requested or not avail themselves of the service.*

This business model conflicts with most people's preferences. Published research indicates that people generally want to control who can retain their personal sensor data [1]. Moreover, people want to control the granularity of the data that is being collected by apps and device providers, which is not possible with commercial apps. As shown in Figure 1-2, mobile apps provide users with "all-or-nothing" privacy controls during their installation for data collection and use. Users have to either agree to share all the requested data or not use the service. The premise of this thesis is that it is crucial to allow users to decide the

granularity of this data because the default fine-grained collection of their sensor information could lead to privacy violations [8].

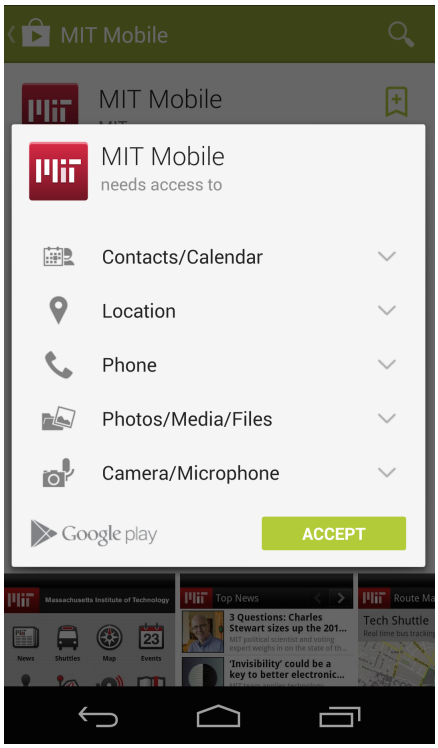

Figure 1-2: *Permissions for data required to install the app "MIT Mobile" on an Android phone.*

1.2 Related Work

This section describes related work from the aspect of other user-controlled privacy mechanisms, specifically for mobile personal data.

A recent study [14] has found that the "all-or-nothing" access to applications does a remarkably poor job of meeting the participants' self-reported preferences. In fact, user preferences are so intricate that even determining how much contextual factors can affect people's decisions about data disclosure is difficult [19]. Additionally, default profiles, although not expressive, seem easy for users to understand and adapt to. However, participants in a study [21] who used the default settings ended up sharing more of their location information compared to other participants. Thus, the drawbacks of placing a cognitive burden on users to create policies to protect their location data should be carefully considered.

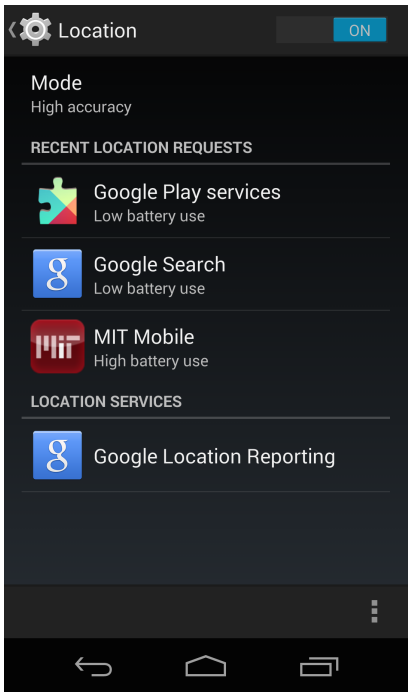

Figure 1-3: *Controls to specify sharing location with apps: Android*

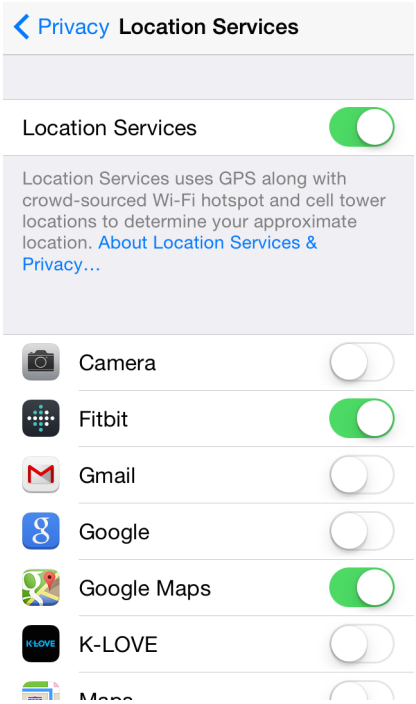

Figure 1-4: *Controls to specify sharing location with apps: iOS*

The iOS framework seems to have a little more flexibility than the Android platform in this regard [10]. The iOS platform allows users to specifically turn off location sharing with specific applications (as shown in Figure 1-4), unlike what is possible in Android (as shown in Figure 1-3). Thus, even though the users may agree to the data collection initially during installation of an app, they have the option to turn off sharing their location information with that particular application later on. PrivacyMate aims to extend the model of iOS and allow users to select or deselect various types of data (more than just location information) from being collected by apps. Existing engineering efforts in this area include SensorSafe [2] and Webinos [16].

Chakraborty et al. [2] provided a flexible user interface for fine-grained and context-dependent access control and employ PDS-specific (e.g., Django) rule-based sharing mechanisms. For Web-based applications, Lyle et al [16] developed a XACML-based framework that provides a single policy enforcement mechanism for a user's "personal zone" – the devices they own and use. Rather than expect users to create rules or policies, PrivacyMate enables users to define permissions using visual controls via the the *opt-in to data collection*,

context definition, and *opt-in to data aggregation* settings provided for a given app. Specifically, PrivacyMate focuses not only on what data is used, but also what data should be stored in the user's personal data store in the first place. Further, PrivacyMate does not have to consider the notion of *data sharing* because it focuses solely on *first use*. In other words, the code of any app resides wholly within a user's PDS machine.

This thesis demonstrates how to build privacy-preserving location-based apps on the openPDS platform [7]. openPDS is an alternate and decentralized platform that enables users to store their mobile personal data (sensor, location, call logs, and other mobile information) on their own machines or on trusted servers. Tools for privacy-preserving distributed computing are not novel [4], and there are other decentralized privacy-preserving and open architectures such as ipShield [3], πBox [15], Koi platform [11], and Open mHealth [9]. However, apps created using our approach have the ability to preserve users' privacy by utilizing the *question and answer* and *group computation* techniques of openPDS. These techniques enable the apps to function without sharing users' raw or fine-grained personal information with other participants or the apps themselves. Yet, the problem with openPDS is that it does not provide user-controllable privacy mechanisms (access control) in order for users to specify their privacy preferences.

1.3 Contributions of this thesis

This thesis presents PrivacyMate, a suite of tools designed to provide user-controllable privacy mechanisms for mobile apps on personal data stores. Although PrivacyMate is currently integrated into openPDS within the MIT Living Lab platform, it can potentially be built on top of any decentralized architecture or even on mobile platforms (e.g., Android or iOS). PrivacyMate aims to enable users to maintain better control over their mobile personal data. Privacy preferences can be specified in PrivacyMate by creating temporal or spatial contexts and indicating what data users are comfortable sharing with apps. This mechanism uses "opt-in by choice" settings and spatio-temporal contexts.

The MIT Living Lab project aims to integrate and visualize data collected for and by the MIT community. To achieve that, apps (known henceforth as "labs") are built on the

Living Lab project in the MIT Mobile app. Using PrivacyMate, users can design and enforce context-specific access controls for various labs in the Living Lab project.

We incorporate the privacy mechanisms offered by PrivacyMate into two labs built on openPDS: ScheduleME (also called as "Meetup") and MIT-FIT. ScheduleME is a privacy-preserving alternative to the mainstream scheduling apps. It enables users to schedule meet-ups without explicitly disclosing either their location(s) or their point(s) of interest. MIT-FIT is a privacy-preserving alternative to the mainstream fitness apps. MIT-FIT appears to be the first fitness app built on a privacy-preserving decentralized architecture that provides visualizations based on quantified activities to enable behaviors that help users achieve their fitness goals.

Note that PrivacyMate is part of the MIT Living Lab platform. Thus, it does not have to be included as a library within each newly created lab. When developers build a new lab, PrivacyMate's functionality would automatically be included in their lab. Chapter 3 shows the procedure to create a new lab on the MIT Living Lab platform. MIT-FIT and PrivacyMate have been beta-tested and will be launched in an upcoming release to the MIT campus.

1.4 Thesis Overview

The rest of this thesis is structured as follows.

- **Chapter 2** describes the existing infrastructures that PrivacyMate, ScheduleME, and MIT-FIT build upon. Specifically, it discusses openPDS, MIT Mobile app, and DataHub, which are part of the MIT Living Lab platform.

- **Chapter 3** describes the design decisions and technical implementation of Privacy-Mate. A tutorial is provided that shows how to build new labs on top of openPDS and PrivacyMate.

- **Chapter 4** describes building ScheduleME as a sample lab on openPDS and Privacy-Mate. ScheduleME aims to enable users to schedule meetups without explicitly sharing either location information or points of interest.

- **Chapter 5** describes building MIT-FIT as a sample lab on openPDS and PrivacyMate. MIT-FIT aims to provide high-activity details by time and location to create awareness in the users. It also recommends fitness-related activities (as advertised by MIT's recreation-promoting center) based on the user's activity patterns.

- **Chapter 6** demonstrates how PrivacyMate and MIT-FIT work together by describing the workflows that users would go through and the functionality of MIT-FIT with and without PrivacyMate. It also discusses the usability and accessibility testing conducted by the Usability team of the IS&T Department at MIT.

- **Chapter 6** concludes the thesis and discusses future work.

Chapter 2

Existing Infrastructure

Since this work was developed as part of the MIT Living Lab project, it is imperative to first consider the already existing infrastructure of the project. This chapter outlines its technical infrastructure, with an emphasis on the openPDS platform. Other infrastructures that were inspirations for portions of this work are also described. Finally, the chapter describes a roundtable discussion that we conducted among a sampling of the MIT community to better understand the requirements for Quantified-Self apps and devices.

2.1 MIT Living Lab Architecture

The MIT Living Lab project was started by the bigdata@CSAIL initiative and provides the ability to access, visualize, share, and use the data generated by and for the MIT community. The goal is to allow MIT itself to become more data-driven.

The MIT Living Lab project is built on the following components: (i) openPDS (server acting as a personal data store), (ii) DataHub (backend repository for the personal data), and (iii) Android/iOS (mobile clients). Figure 2-1 shows the architecture diagram of the Living Lab project and the stack of various technologies used in building it. These component platforms are then explained.

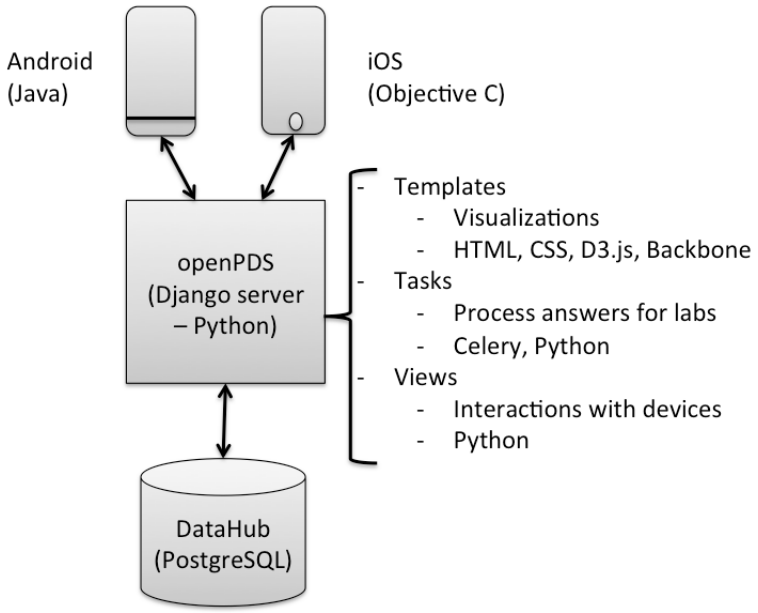

Figure 2-1: *Technical infrastructure of the MIT Living Labs platform.*

2.1.1 openPDS

openPDS is an open-source platform that enables users to store their personal sensor data on their own machines or on trusted servers. This creates a decentralized data storage and processing solution for mobile personal data. Unlike the access control model shown in chapter 1, the data collection, storage, and retrieval model on openPDS (shown in Figure 2-2) enables users to be the actual collectors of their own data and gives them the option to share their data with apps for use. This decentralized approach is quite opposite to the centralized, rigid approach that makes the app providers the data collecting agents. Under this new framework, users would not have to disclose their raw data to third parties or to app providers. In this way, openPDS preserves the privacy of users' private data.

Technically, openPDS is a Django-based webserver that provides (i) data connectors to store, fetch, and query the personal data; (ii) views to interact with the mobile clients and devices; and (iii) templates to render visualizations after processing the computations (functionalities) of various labs. The source code is hosted on GitHub[1] and is written primarily in scripting (Python) and Web-based (HTML, D3.js, backbone MVC framework, and CSS)

[1]https://github.com/HumanDynamics/openPDS

languages.

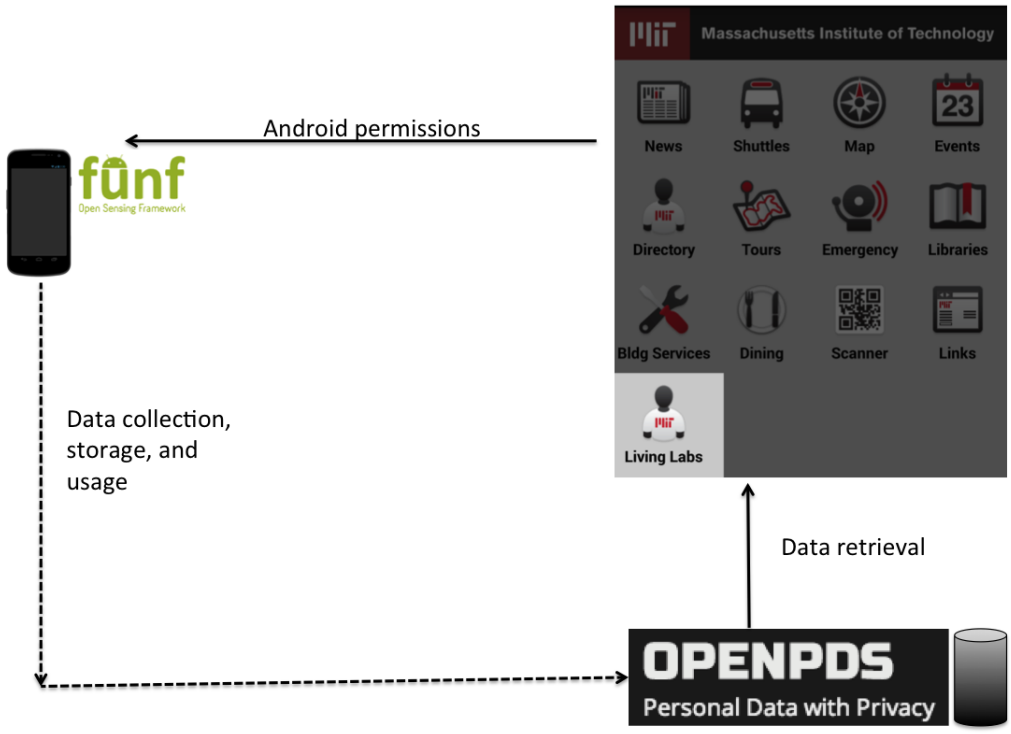

Figure 2-2: *Proposed access control model using openPDS. It enables users to specify what personal data can be accessed by specific labs.*

2.1.2 MIT Mobile App

The MIT mobile app was developed by the MIT IS&T Department[2] for both the Android and iOS platforms. The code for the Android platform has been modified to specifically integrate the MIT Living Lab project. This version of the mobile app has already been populated with the following labs: Social Health Tracker, My Places, and ScheduleME (also called Meetup). The goals of these labs are as follows:

- *Social Health Tracker* aims to provide sociometric analyses of people's behavior ("social health") along the activity, social, and focus axes.

[2]https://ist.mit.edu/

- *My Places* aims to identify the "work" and "home" locations of people by analyzing where they spend most of their waking and sleeping hours.

- *ScheduleME* aims to enable people to schedule meetups without revealing either their locations or their points of interest.

Technically, the MIT Mobile Living Labs app is an open-source Android client. The source code is hosted on GitHub[3] and is written primarily in Java. It incorporates the Funf library[4] which is another open-source platform that provides the MIT Mobile Living Lab app with a background service to collect sensor and other mobile information from the device. The MIT Mobile Living Lab app also incorporates the Google Play Service[5] and the Android ActionBar Sherlock library[6] to provide a better user experience.

2.1.3 DataHub

DataHub is an open-source platform that aims to provide a user-friendly interface through which users can interact with their databases. It provides a GitHub-like versioning control for the backend schemas and data. Technically, DataHub provides Web and command-line interfaces to PostgreSQL (relational) databases. The source code is hosted on GitHub[7] and is primarily written in Python and JavaScript.

2.2 A Closer Look at openPDS

PrivacyMate and MIT-FIT leverage the openPDS platform to incorporate their various features. Therefore, it would be helpful to have a closer look at the design of openPDS, with a particular emphasis on its privacy-preserving characteristics. openPDS comprises the following two privacy-preserving mechanisms: "Question and Answer Framework" and "Group Computation."

[3]https://github.com/HumanDynamics/MIT-Mobile-for-Android
[4]http://www.funf.org/
[5]http://developer.android.com/google/play-services/index.html
[6]http://actionbarsherlock.com/
[7]https://github.com/abhardwaj/datahub

2.2.1 Question and Answer Framework

As mentioned previously, openPDS users do not have to disclose their raw mobile personal data to app providers or third parties. This is accomplished through the *question and answer framework* of openPDS. One of the requirements set by openPDS is that any lab hosted on it should have all the necessary code residing on the computing space of the user's PDS machine. In other words, a lab cannot send a user's raw data to an external location for further processing or for providing results of computations. Within the scope of this limited processing, openPDS consists of two APIs (Application Programming Interfaces) for data access – an internal API and an external API. The internal API grants purpose-based authenticated access (using OAuth[8]) to the labs residing on openPDS. This authentication prevents the labs from directly accessing a user's raw mobile personal data and from sending the data outside the PDS machine. Instead, a lab can ask "questions" of a user's PDS instance such as "Where was the user location at 10am in the last two weeks?" or "What is the activity level of the user when present at (42.3617434,-71.0906687)?" Though these sample questions were provided in English, the actual questions asked by the labs are scheduled as tasks and written as Python code. A user's PDS instance would then access his or her raw mobile personal data and compute the "answers" to those questions. The external API grants the lab an authenticated access (again, using OAuth) to the answers computed by openPDS. Thus, the labs never directly see, share, or use the users' raw mobile personal data. They can only access the processed results that answer questions they seek to understand or analyze for the user.

2.2.2 Group Computation

When labs ask "questions" of a user's PDS instance, openPDS processes the raw data of the corresponding user. However, for "questions" involving data from a segment of (or all) users, openPDS has to query the respective PDS instances to compute an answer. This process is called *group computation* and enables labs to provide group or aggregation analyses. Figure 2-3 shows a walkthrough of the interactions and control flows involved in the group

[8]http://oauth.net/

computation process.

The openPDS ecosystem consists of a "ring" of PDS instances and a registry server. The registry server keeps track of the authentication and instance details of the openPDS users. Processing starts at an initiator's PDS instance. The initiator can be either purposely or randomly chosen by the lab. The intermediate contribution (running total) is then transmitted from that PDS instance to the next one in the ring of PDS instances for further processing. This process continues (Figure 2-3 [steps 5(a), 5(b) and 5(c)]) and once all the instances finish processing over the incoming intermediate results, the final "answer" is obtained. The initiator's PDS instance then broadcasts the result to those of the other participants (Figure 2-3 [step 6]). Finally, the PDS instances would notify the users of the results (Figure 2-3 [steps 7]).

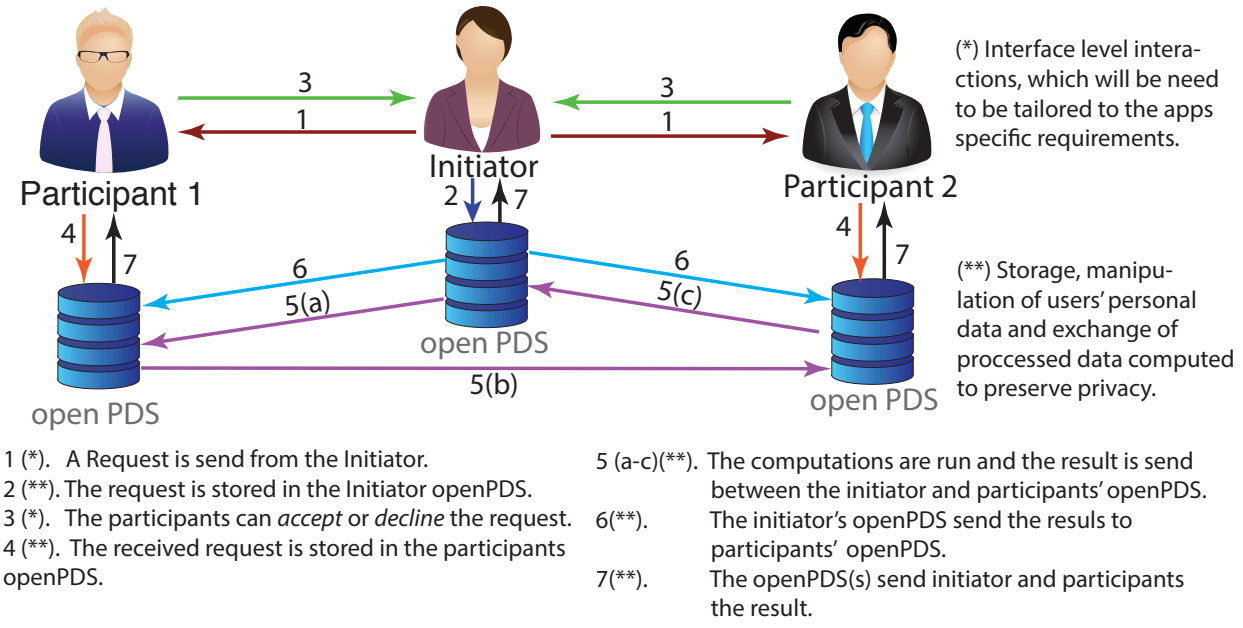

Figure 2-3: *Example of the interactions, storage and manipulation of users' personal data and exchange of processed data between personal PDS instances underlining the privacy-by-design structure of the framework. [20]*

2.3 What's Missing?

The infrastructures described in the previous three sections are extensive, but currently lack the following functionalities and capabilities:

- The MIT Living Lab project currently does not include user-controllable privacy mechanisms.
- The MIT Mobile Living Labs app currently lacks a quantified-self lab that could promote active and healthy living within the MIT community.

2.4 Requirements Gathering

We conducted a requirements gathering study (roundtable discussion) to better understand the experiences of a sampling of MIT affiliates with Quantified-Self apps and devices. We used the takeaways as part of our requirements-gathering in the design of the MIT-FIT lab.

2.4.1 Procedure

We had a one-hour-long roundtable discussion led by a facilitator. The participants were asked whether they used quantified-self apps or devices previously. If they had, they were then asked to share their experiences. If they had not, they were asked for their reasons. Based on their replies, follow-on questions were asked. The questionnaire followed during the roundtable discussion is listed in Appendix 8.1. This was a casual gathering and we received very candid responses.

2.4.2 Participants

We primarily focused on recruiting students for the roundtable discussion, but allowed non-students to participate as well. Some of the participants were initially recruited from the bigdata@csail student mailing list. The MIT Department of Athletics, Physical Education & Recreation (DAPER)[9] had asked participants in its annual survey to indicate whether they would like to be contacted for a research study. Thus, later in our recruiting process, we contacted those who opted in via the DAPER survey. Overall, we had nine participants in the roundtable discussion (7 students – both graduate and undergraduate students, 1 MIT affiliate, and 1 staff).

[9]http://web.mit.edu/athletics/www/

2.4.3 Takeaways

Following are the major takeaways from the roundtable discussion.

- Participants who were students seemed generally comfortable sharing data with MIT. In contrast, the staff member was not comfortable doing so and raised concerns related to legal and insurance issues. We construe this as a validation for the "opt-in to data collection" setting.

- Regarding privacy settings for data collection and use, one student said that they would "want to turn it on", instead of "actively turning it off." We construe this as a validation for our "opt-in by choice" settings.

- The privacy model in Foursquare was preferred by a participant. In that model, only friends that you choose to share your location with know where you are (and only when you share). Others are not aware of your location at any time.

- Some of the participants stated that displaying the purpose of use of data collected was important to them. The reason given was that the users do not know what the app providers would be doing with the data they collect from the users.

- One participant stated that, at a minimum, apps should turn off sharing their data passively with others and should not post their data to social networking sites like Facebook automatically.

- Monitoring sleep was important for some of the participants. One participant mentioned having friendly competitions with labmates over their "sleep efficiency."

- Leaderboards that listed individuals did not seem interesting to the participants. They seemed to prefer competing in groups (as part of their dorms or housing) against other such groups.

- Regarding analyses, daily check-ins or updates seemed to be an overload. Participants seemed to prefer weekly or monthly check-ins in the form of push notifications.

- A few participants considered negative reinforcements to be more effective than positive ones. Some examples given were: collecting money at the outset and then donating it (or a portion of it) to a political cause that a user opposes or to the other participants of the app, if that user does not meet his or her goal.

2.5 Summary

This chapter has discussed the existing infrastructure in the MIT Living Lab project and highlighted what is missing. It also discussed the roundtable discussion held to better understand the requirements for a quantified-self app for the MIT Living Lab project.

Chapter 3

PrivacyMate: User-Controllable Privacy Mechanisms for Living Labs

PrivacyMate is not a standalone system, but rather is a suite of tools that provides user-controllable privacy mechanisms for the MIT Living Labs project. This chapter provides the technical details of PrivacyMate. It starts with the functional architecture of PrivacyMate and then discusses the design decisions and implementation details of the privacy mechanism.

3.1 Functional Architecture of PrivacyMate

Figure 3-1 shows the functional architecture of a portion of the MIT Living Lab project relevant to this thesis. The data and control flows between the client (Android device with the Funf service) and the openPDS server are shown along with those between the openPDS server and the labs residing on the server. PrivacyMate enhances the current functionality of the MIT Living Lab project by providing user-controllable privacy mechanisms. Users can create preferences (consisting of settings and context) to specify what data they would like to collect and contribute to a specific lab, and also the contexts in which they are comfortable with the data being used. These settings are sent to openPDS along with the data collected by Funf. Further, when labs ask openPDS "questions" about a particular user's data and receive a processed "answer," the corresponding processing takes the user preferences into consideration, thereby enforcing those preferences.

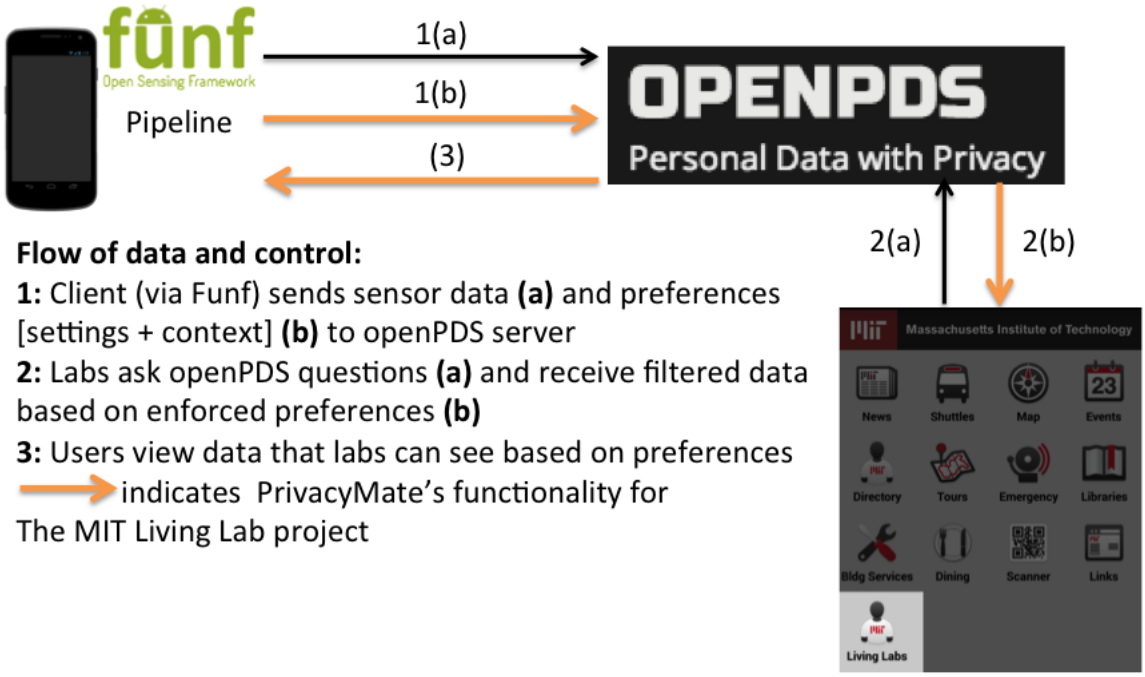

Figure 3-1: *Portion of the MIT Living Lab platform with PrivacyMate's functionality superimposed.*

3.2 PrivacyMate's Privacy Mechanism

When it comes to user-controllable privacy mechanisms, two issues need to be considered: *What controls are displayed?* and *What concerns and functions do those controls address?* This section aims to answer these two questions, starting with the second one, for the suite of tools provided via PrivacyMate.

For personal data stores, the major privacy concerns are *data collection*, *data aggregation*, and *data use* and these constitute the functions targeted by the proposed controls. *Data collection* refers to the collection of the mobile personal data (sensor and other mobile data via Funf, or even external data via third-party webservices or APIs). *Data aggregation* refers to the processing of a user's data in the context of or in conjunction with the data of other users. In other words, this refers to the notion of group computation described in Chapter 2. *Data use* refers to the individual processing of a user's data (not along with the data of other users). The most straightforward processing of an individual's data is for data use and the privacy concern there is not so much about the data itself, but rather about

the *metadata* of the data under consideration – in other words, the *context* of the collected data under which a user would feel comfortable having the lab use his or her data. Since openPDS inherently does not allow sharing of the raw data with either the app providers or third parties, we do not consider the notion of *data sharing*.

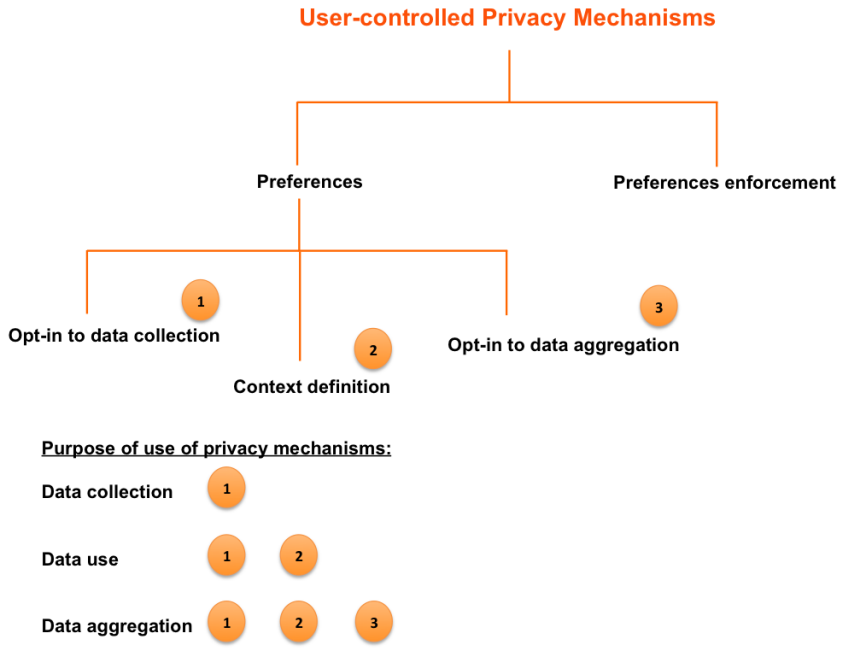

Figure 3-2: *The user-controlled privacy mechanism provided by PrivacyMate in the Living Lab platform.*

To address the privacy concerns just mentioned, PrivacyMate offers the following three controls: *opt-in to data collection*, *context definition*, and *opt-in to data aggregation*. These three controls constitute the access control (preferences) provided by PrivacyMate. As Figure 3-2 shows, the preferences, along with their enforcement, form the four-pronged privacy mechanism offered by PrivacyMate. The access control (preferences) enables users to make an informed decision about what data they are willing to let specific labs collect, access, and use. PrivacyMate provides default settings to help users get set up quickly. Users who prefer to create custom controls can do so using the full range of the features offered by PrivacyMate. These four-pronged privacy mechanisms are briefly introduced below and explained in further detail in the following sub-sections.

1. **Opt-in to data collection:** controls to specify what data a user would like to be collected or used by a specific lab.

2. **Opt-in to data aggregation:** controls to specify whether a user is interested in sharing his or her data to be used in the group computations performed by a specific lab.

3. **Context definition:** controls to specify the spatio-temporal context (metadata) within which a user is comfortable having a specific lab use his or her data.

4. **Preference enforcement:** enforcement of the user-defined preferences by the PDS instance.

Example. Figures 3-3 and 3-4 show the preferences created by a sample user. According to these preferences, the user has chosen to have both activity and location data collected and used by the MIT-FIT lab. Usage is indicated to be limited to the temporal and spatial specifications of the *MIT* context (pre-defined, but editable by the user).

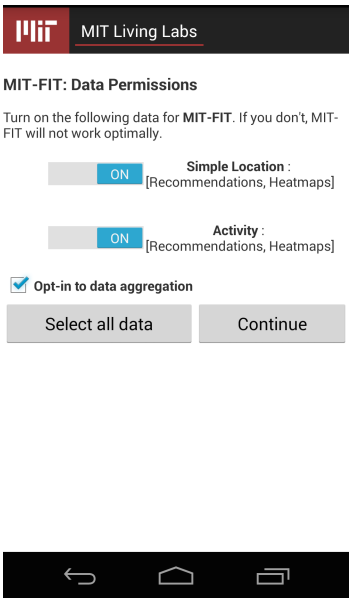

Figure 3-3: *Opt-in to data collection control. This screenshot lists the probes (along with the purpose that the corresponding data would be used for) that a particular lab needs. Also note the opt-in to data aggregation control (toggle switch).*

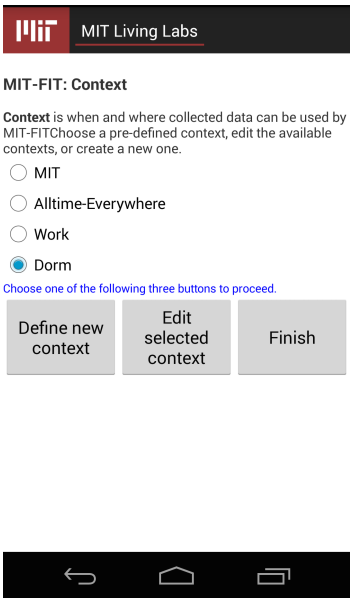

Figure 3-4: *The context definition control. This context home screen shows the two pre-defined contexts "MIT" and "Alltime-Everywhere." Users can define new contexts and edit existing ones.*

3.2.1 Opt-in to Data Collection

These controls inform users about the data (location and other sensor data) that are required by a particular lab along with their purpose of use. For instance, Figure 3-3 shows that the MIT-FIT lab requires the activity and simple location data to function. These preferences for data collection and use are stored both locally (in the SQLite database on the device, for data collection) and remotely (on the openPDS server, for enforcement). To make it easy for the users, we also provide an option for users to select the data globally for all labs using the Global Settings (as shown in Figure 3-5). These settings enable users to turn data collection on or off for all the probes as well as being able to selectively turn on only data that are required by the labs installed on that user's PDS instance.

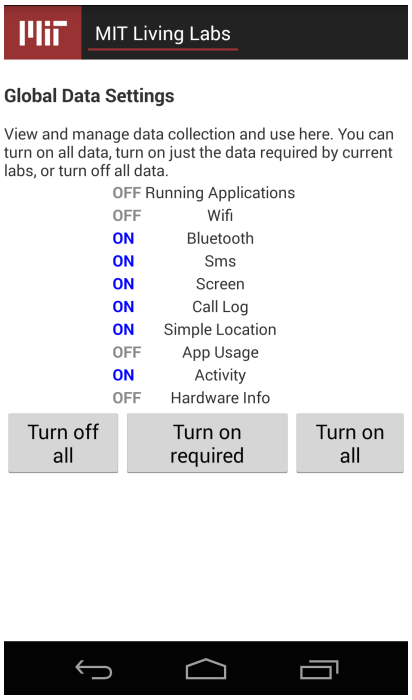

Figure 3-5: *Global listing of all the probes that Funf collects for the MIT Living Lab project*

Since global settings apply to all labs installed on a user's PDS instance, the switches on this screen are "read-only." We provide the nuclear options of de-selecting and selecting all data for data collection from the device. However, from a privacy standpoint, it might make sense to only collect data if there is at least one lab installed on the user's PDS that requires that data. Thus, the option of selecting only required data is also provided.

Data is collected from various sensors on the device using the Funf library. Internally, Funf constructs a "pipeline" that specifies the schedule of data collection and the probes from which data is permitted to be collected on that schedule. This pipeline is invoked by the Funf manager, which runs continuously as a background service (broadcast launcher/receiver) on the device. PrivacyMate computes the union of the probes permitted by the user across all labs to determine the new list of probes for data collection. This new list is passed as a modified pipeline to the Funf manager so that only the data explicitly selected by the user are collected.

3.2.2 Opt-in to Data Aggregation

Figure 3-3 also shows the opt-in to data aggregation control. Using this control, users can specify whether they would like to allow labs to use their data for group computations. When users select this option, their PDS instance would include their data in group computations and pass the intermediate results along to others in the ring of PDS instances.

3.2.3 Context Definition

Context refers to spatio-temporal constraints (metadata) specified by users that govern what data they would like the lab to use. These are specified on the client device, but stored on the openPDS server. To simplify the process of defining contexts, PrivacyMate provides two pre-defined contexts by default: MIT and Alltime-Everywhere as shown in Figure 3-4. That screenshot is the context "home screen" that lists the pre-defined and user-defined contexts available for use. As their names may suggest, the *MIT* context defines the MIT campus as its spatial constraint and 10am through 6pm on weekdays (working hours) as its temporal constraints. The *Alltime-Everywhere* context, on the other hand, does not specify any location and selects all the times (24X7) as its temporal constraints. As seen in Figure 3-4, users also have the option to create new contexts or edit both the pre-defined as well as the user-defined contexts. Figures 3-6 and 3-7 show how to specify the temporal and spatial constraints for the pre-defined "MIT" context. The context editing screens for user-defined contexts provide users an additional option to delete that context. As indicated by the radio

buttons in Figure 3-4, only one context may be associated with any given lab.

Locations specified in the spatial specification of a context may potentially be more sensitive than the sensor data collected, and can raise concerns like "why does this user consider non-indicated (other) locations to be sensitive?" Thus, PrivacyMate does not share the locations specified in the contexts with the labs. The internal API uses the context as a filter to determine whether the lab is allowed to access a specific data object.

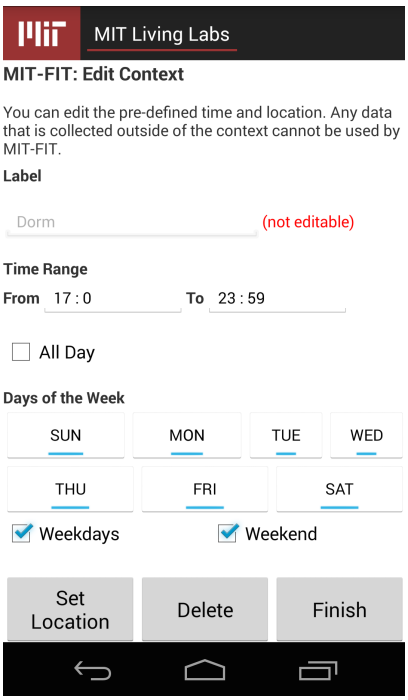

Figure 3-6: *Temporal specifications of the context definition.*

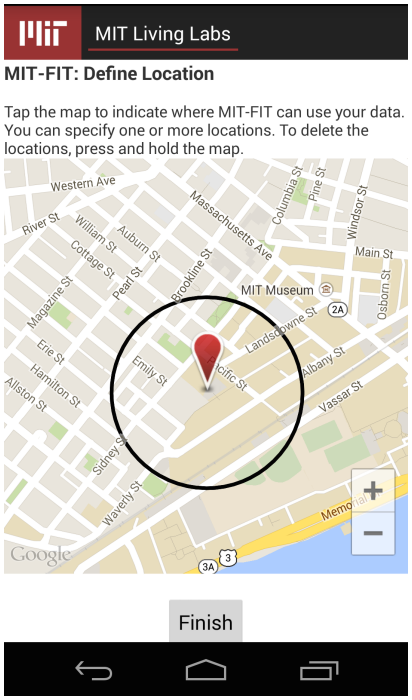

Figure 3-7: *Spatial specifications of the context definition.*

3.2.4 Preference Enforcement

When a lab requests openPDS to answer a "question," openPDS collects the necessary data using the internal API and determines whether the lab is allowed to access each data object. Using the preferences created by the user, openPDS accomplishes the preference enforcement as follows.

1. **Opt-in to data collection:** if a lab requests a particular type of data (such as, Wi-Fi), but the user did not enable its collection, the PDS instance returns a *None* object

to the lab. If that type of data is available on the user's PDS, the context associated with this lab is investigated next.

2. **Context definition:** the PDS instance checks the temporal and spatial specifications of the context associated with the lab under consideration as follows:

   - **Temporal specifications:** the timestamp associated with a particular data object (e.g., call log or Bluetooth data) is checked to determine whether the time of collection of that data object lies within the start and end times specified in the context associated with the lab. Further, the day of the week of the timestamp of the data is also compared with the days selected in the corresponding context to determine permissibility. If these two conditions are met, the temporal specification of the context is deemed to be satisfied.
   - **Spatial specifications:** If no location is specified in the context associated with a lab (as in the "Alltime-Everywhere" context), the lab is allowed to access any data specified by the *opt-in to data collection* control that satisfy the temporal constraints of that context. Otherwise, the location data at the timestamp corresponding to a specific data object is retrieved. This location is checked to see whether it (the latitude/longitude coordinate pair) lies within a 500-meter radius of any of the location(s) specified in the corresponding context. If the latitude/-longitude pair satisfies this condition, the spatial specification of the context is deemed to be satisfied. The distance computation between two latitude/longitude coordinates needs to consider the spherical shape of the Earth. Thus, PrivacyMate employs the *Haversine formula* to compute the distance between the location associated with a particular timestamp (at which a data object was collected) and the location specified in the context definition. Using the Haversine formulation, and considering the radius of the Earth (r) to be 6372.8 Km, the distance between two points (ϕ_1, λ_1) and (ϕ_2, λ_2), is

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

If a data object satisfies both the temporal and spatial specifications of the context associated with a lab, that lab is granted access to that data object and the PDS instance of that user can use that data object when computing the "answer" for the "question." Since location data is required to perform these checks for the spatial constraints, we treat the location probe as a special data probe and always turn it on for data collection. If the *opt-in to data collection* control for a user indicates that the user does not want location data to be used by a particular lab, the corresponding lab will not have access to the location data via openPDS. However, openPDS will still perform the spatial checks mentioned previously for the other data that are allowed by the user for that lab.

3. **Opt-in to data aggregation:** if a user does not select the *opt-in to data aggregation* control, openPDS would not use that user's data for group computations. However, the data permitted by the checks for *context definition* and *opt-in to data collection* controls would still be used for individual use (e.g., to render visualizations of the user's data back to the user).

3.3 Tutorial: How to build a new lab on openPDS and PrivacyMate

To create a new lab, a developer needs to make changes to the MIT mobile app and openPDS server. PrivacyMate's functionality is part of the framework and therefore a developer does not have to build additional code to integrate his or her code. This section discusses the steps needed to develop a lab and presents brief code snippets. A fully-fleshed walkthrough to develop a sample lab is hosted on the openPDS repository wiki on GitHub[1].

3.3.1 On the openPDS server

The processing and visualizations of the lab have to be built on the openPDS server. The following four steps describe this procedure.

[1]https://github.com/HumanDynamics/openPDS/wiki/Tutorial:-Creating-a-new-lab

Step 1 Create a Python task in a file called *yourlabname_tasks.py* (replace "yourlabname" with the name of your lab) in the *oms_pds* directory. Tasks typically consume either raw or processed data using the internal API and produce processed data for the lab's visualizations (consumed via the external API). The code snippet for the MIT-FIT lab's *findActiveLocations* task is provided in Listing 3.1. The dependent task, *recentProbeDataScores* is shown in Listing 3.2 and provides data that *findActiveLocations* uses internally.

Listing 3.1: MIT-FIT task and associated helper function

```
def activeLocationsComputation(internalDataStore):
    activityAnswerList = internalDataStore.getAnswerList("
        recentActivityProbeByHour")
    # compute the result using your logic
    return locationPoints

@task()
def findActiveLocationsTask():
    profiles = Profile.objects.all()
    for profile in profiles:
        # provide user profile, app name, lab name, and token
        internalDataStore = getInternalDataStore(profile, "Living␣
            Lab", "MIT-FIT", "")
        # use the values returned from the helper function
        values = activeLocationsComputation(internalDataStore)
    # compute location_frequencies using your logic
    for key  in location_frequencies:
        location_value = { "lat": key[0], "lng": key[1], "count":
            location_frequencies[key]}
        location_frequencies_list.append(location_value)

    for profile in profiles:
```

```
        internalDataStore = getInternalDataStore(profile, "Living
            Lab", "MIT–FIT", "")
        # saving the processed answers to the database
        internalDataStore.saveAnswer("activeLocations",
            location_frequencies_list)
```

Listing 3.2: recentProbeDataScores task and associated helper function

```
def probeForTimeRange(probe, internalDataStore, start, end...):
    probeEntries = internalDataStore.getData(probe, start, end)
    # compute processed results using your logic
    if probe == 'LocationProbe':
        location = { "start": start, "end": end, "centroid":
            centroids }
        return location
    elif probe == "ActivityProbe":
        activity = { "start": start, "end": end, ..., "high":
            highActivityIntervals }
        return activity
    return None


@task()
def recentProbeDataScores():
    profiles = Profile.objects.all()
    # perform the task for all users
    for profile in profiles:
        # provide user profile, app name, lab name, and token
        internalDataStore = getInternalDataStore(profile, "Living
            Lab", "MIT–FIT", "")
        # use the values returned from probeForTimeRange
```

Step 2 Add the task file to the *celery* scheduler in the *oms_pds* directory along with a schedule of when to run the tasks. Listing 3.3 demonstrates how to run the *recentProbeDataScores* task twice every hour at the 8 and 38 minute marks.

Listing 3.3: Adding the *recentProbeDataScores* task to *celery*

```
"probe-summaries": {
    "task": "oms_pds.probedatavisualization_tasks.
        recentProbeDataScores",
    "schedule": crontab(hour="*", minute="8,38")
},
```

Step 3 Create the visualization UI using HTML, CSS, and JavaScript on top of the *backbone.js* MVC platform. Create the HTML templates in the *oms_pds/templates/visualizations* directory. Create the JavaScript files in the *oms_pds/static/js* directory. Use *backbone.js* to process the model and provide data to the JavaScript variables to render or manipulate. Listing 3.4 shows the HTML snippet, while Listing 3.5 shows the outline of the JavaScript snippet needed to create the visualization.

Listing 3.4: MIT-FIT HTML for High-Activity Locations

```
<!-- Extend backbone layout and add script files -->
<script>
$(function () {
        //calling the AnswerListMap JavaScript function
    window.answerListMap = new AnswerListMap("
        recentSimpleLocationProbeByHour", "
        recentActivityProbeByHour", "answerListMapContainer", true,
         "user");
});
</script>
<div id="answerListMapContainer"></div> <!-- div for result -->
```

Listing 3.5: MIT-FIT JavaScript for High-Activity Locations

```
window.AnswerListMap = Backbone.View.extend({
    el: "#answerListMapContainer",
    initialize: function (locationAnswerKey, activityAnswerKey,
       mapContainerId, autoResize, entity) {
        _.bindAll(this, "render", "renderPlaces");
        this.render();
        this.locationAnswerLists = new AnswerListCollection([],{ "
           key": locationAnswerKey });
        this.locationAnswerLists.bind("reset", this.renderPlaces);
        this.locationAnswerLists.fetch();
        // repeat previous 3 lines for activityAnswerKey
    },

    render: function () {
        //render to answerListMapContainer div
        this.map = new google.maps.Map(document.getElementById("
           answerListMapContainer"), myOptions);
   },

    renderPlaces: function () {
        var locationEntries = (this.locationAnswerLists && this.
           locationAnswerLists.length > 0)? this.
           locationAnswerLists.at(0).get("value"):[];
        //process locationEntries and create visualization

    },

});
```

Step 4 Add the path to the HTML visualizations to a *oms_pds/visualization/urls.py* file on the server. This enables Django to locate the visualizations for rendering. Listing 3.6 shows how to add the *mitfit_user_location* page to the urls.py file.

Listing 3.6: Adding the visualizations location to the urls.py file

```
urlpatterns = patterns('oms_pds.visualization.views',
        (r'^mitfit/userlocation$', direct_to_template, { 'template
            ' : 'visualization/mitfit_user_location.html' }),
    )
```

3.3.2 On the MIT Mobile Living Lab App client

The mobile client has to be updated to allow users to view the newly-created lab in the MIT Living Lab app.

Step 1 Add the lab to the *pds_strings.xml* file. The following parameters of the lab should be added: lab name, data requirements, and URL (matching the path specified in the urls.py file in step 4 in the previous sub-section). Listing 3.7 shows how to add the "High-Activity by Location" page to the *pds_strings.xml* file.

Listing 3.7: Adding the MIT-FIT lab to the mobile app

```
{
        'name': 'MIT-FIT',
        'visualizations': [
                {'title': 'High-activity Locations', 'key': '
                    mitfit/userlocation', 'answers': ['
                    activeLocations'] },
        ],
}
```

3.4 Summary

This chapter has described the four-pronged privacy mechanism offered by PrivacyMate. Note that these privacy mechanisms are not included as libraries. Instead, these four mechanisms are integrated as part of the MIT Living Lab platform and available to any lab (including those that may be built in the future).

Chapter 4

Implementation of ScheduleME on PrivacyMate

This chapter presents the ScheduleME lab that was implemented in collaboration with other researchers [20]. Through this lab, we demonstrate how to build privacy-preserving labs that perform group computations on the MIT Living Labs project. ScheduleME uses mobile and sensor data for performing group computations in privacy-preserving ways. ScheduleME differs from other scheduling apps by not forcing users to reveal either their current or desired physical locations or inexact points of interest.

This chapter discusses the design decisions and technical details of the ScheduleME lab as well as a comparison of ScheduleME with other commercial and research-based scheduling apps. We discuss how users can achieve privacy assurances on ScheduleME without sacrificing the desired utility obtained in other apps.

4.1 ScheduleME Design: Goals

The goal of ScheduleME is to provide a privacy-preserving way to enable users to schedule meetups without revealing either their current or desired physical locations (latitude/longitude coordinates) or their inexact points of interest (such as landmarks).

4.2 ScheduleME Implementation: Group Computation

The user interface of ScheduleME is shown in Figure 4-1. A user who wants to schedule a meetup is called an "initiator." An initiator starts the workflow by selecting the people to invite. This is accomplished by entering email addresses of those people (Figure 4-1(a)). All users on openPDS are authenticated by the registry server using their MIT Kerberos ID[1] and thus openPDS knows their MIT email addresses. The invited people can either approve, deny, or delete the request. All who approve the incoming request are considered as "participants" in this workflow. Once openPDS receives notification from all invited people, the initiator's PDS instance randomly selects a location for each hour of the day from the bounding box of locations computed based on his or her location history, as collected via Funf. This suggested location is relayed to the neighboring PDS instance in the ring of PDS instances (maintained by the initiator's PDS instance and obtained from the registry server). The neighboring PDS instance then determines the centroid between the incoming suggested location and its own random location suggested from the bounding box for that hour of the day (Figure 4-1(b)). The neighboring PDS also determines the distance from the centroid to its own suggested location and adds this to a running tally of distances for each hour of the day. The neighboring PDS instance then sends the newly computed centroid and the running distance score for each hour to the next PDS instance in the ring. Once all the PDS instances in the ring (consisting of the PDS instances of users who have both opted in using the *opt-in to data aggregation* and have accepted the invitation for the meetup) have finished processing, the result reaches the initiator's PDS instance. The initiator selects the centroid with the lowest distance score as the meetup location. Because this workflow requires the PDS instances to pass a centroid as an intermediate result, a participant does not have to reveal his or her historical or current location or points of interest.

[1] https://ist.mit.edu/start/kerberos

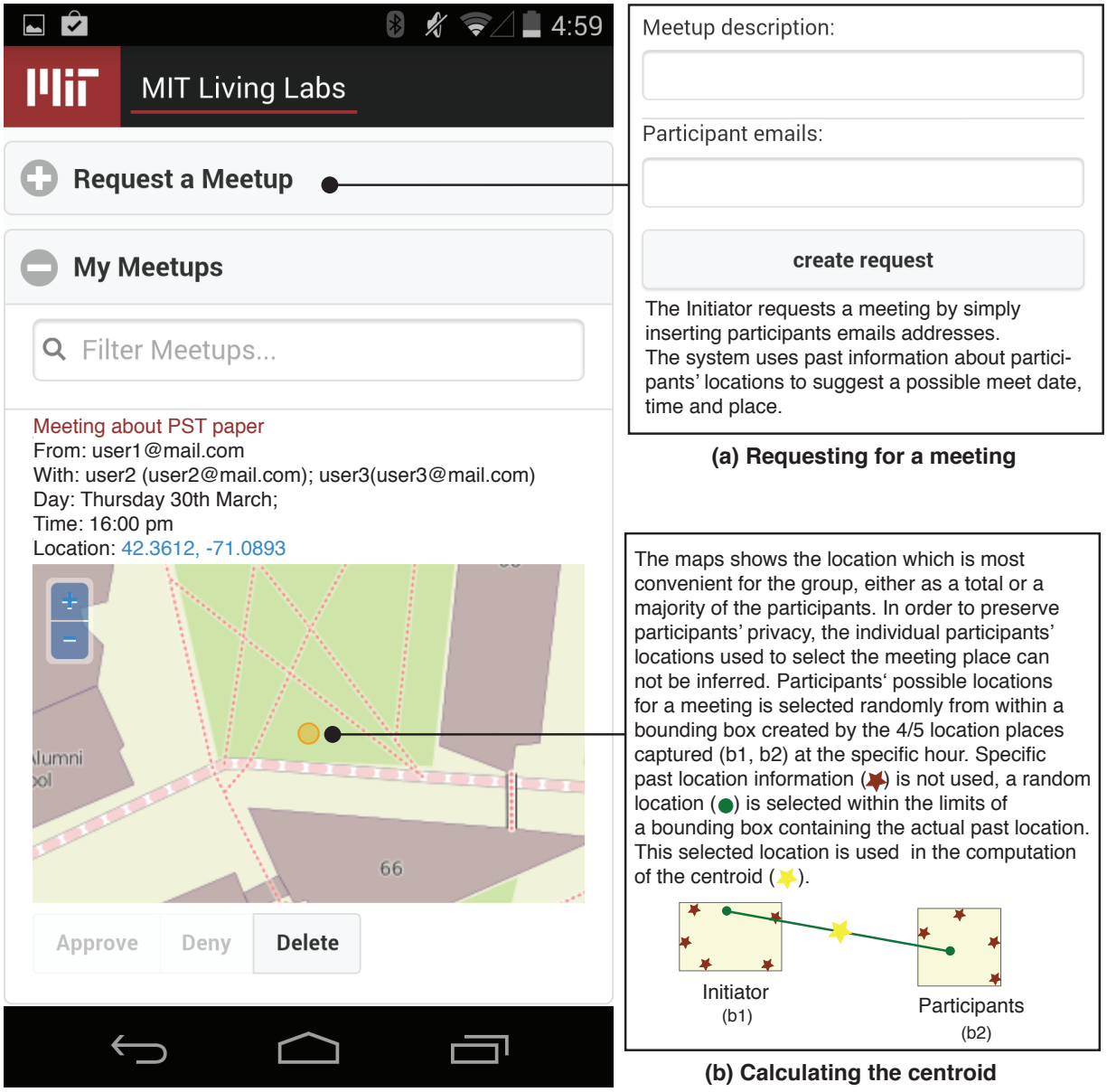

Figure 4-1: *ScheduleME app created using openPDS: showing (a) the interface to request a meeting; (b) the possible results of a group computation, explaining how the actual personal location information is preserved and the computed answer is shared among users' PDS to maintain participants' privacy. [20]*

4.3 Related Work

This section discusses how ScheduleME differs from commercial and research-based scheduling apps.

4.3.1 Factors used in comparison

We focus on the following five factors for comparing ScheduleME with the other apps:

- **Visibility:** who can view a user's location and with whom users can share their location;
- **Real-time location tracking:** whether the app allows users' locations to be tracked in real time;
- **Privacy techniques:** privacy-preserving mechanisms provided by the app;
- **Log-in access:** whether an app requires users to create a new account or allows them to use their existing social network profiles;
- **Framework structure:** whether the app's framework is centralized or decentralized.

4.3.2 Comparison with ScheduleME

Table 4.1 lists the apps (both commercial as well as research-based) that provide location-based services. *Blendr*[2] and *Skout*[3] allow serendipitous discovery of people by recommending people based on a user's proximity to them, thereby allowing strangers to view a user's *inexact* locations. *Glympse*[4], *Owntracks*[5] and *Miataru - be found*[6] allow users to share their location information with other users. *Waze*[7] and *GPS Plus*[8] provide crowd-sourced and personal traffic navigation, respectively. *Glympse*, *Waze*, and *GPS Plus* enable users to share their location information with others for a user-specified amount of time. After the time expires, the contacts would no longer be able to view that user's location. *Tag - You're It*[9] and *PrivateMeetUp* [13] provide a similar functionality as ScheduleME and use location information to help coordinate meetups. *Tag - You're It* allows users to privately broadcast

[2]http://blendr.com/
[3]http://www.skout.com/
[4]https://www.glympse.com/
[5]http://owntracks.org/
[6]http://miataru.com/
[7]https://www.waze.com/
[8]https://itunes.apple.com/us/app/gps-plus-location-commute/id528698859
[9]https://itunes.apple.com/us/app/tag-youre-it/id829344553

their location and view a suggested route to meet with "tagged" friends. In *PrivateMeetUp*, users do not share actual location information, but send approximate distances to points of interest to their peers.

Table 4.1: *Commercial & research location-based apps, showing the visibility within each app, whether real-time location tracking is possible, the privacy techniques used to safeguard users' location information, type of log in access (F. Facebook, G. Google+, T. Twitter, FS. foursquare, O. Others), and the underlining structure of data storage / communication. [20]*

	APP NAME	VISIBILITY (SHARING)	TRACKING	PRIVACY TECHNIQUES	LOG IN ACCESS		FRAMEWORK STRUCTURE
					NEW	S.N.	
COMMERCIAL	**Blendr**	Public	No	Reveals inexact location	Yes	F.	Centralized
	Glympse	F. T.	Yes	Time-based (can expire early)	Yes	F., T.	Centralized
	GPS Plus	Private	No	Share past history (up to 24 hours old)	Device	N/A	Centralized
	Miataru	Private	Yes	Can use own server to store data	Device	N/A	Decentralized
	Owntracks	Private	Yes	Can use own server to store data	Device	N/A	Decentralized
	Skout	Public	No	Separate communities for adults and teens	Yes	F., G.	Centralized
	Tag	Private	No	Send location and privately view route to meetup	Yes	F.	Centralized
	Waze	F., T., FS.	Yes	Send ETA via email/SMS	Optional	F.	Centralized
RESEARCH	**PrivateMeetUp**	F.	No	Only disclose approximate distance to a particular location	N/A	F.	Decentralized
	ScheduleME	None	No	individual private store, question and answer framework, group computation	Yes (email)	N/A	Decentralized

We now discuss the content of Table 4.1 according to the five factors just defined. Visibility is inferred based on who can view a user's location. Private indicates that only contacts based on the device's contact list are granted permission. Public visibility means that any one using the app can view the user's location (at some level of granularity). For apps that allow social network contacts, but not the general public, to view a user's location, the social networks allowed are listed. Of the ten apps, *Glympse*, *Waze*, *Owntracks*, and *Miataru - be found* allow others (as specified by the visibility column) to track a user's

location. *Glympse* restricts the tracking privileges by time, with a maximum time limit of four hours. As shown in the table, different apps have different means of authenticating users when using their respective services. Some apps allow users to register a new account on their platform while others allow users to use their social network profiles as an alternative to creating a new account. *Miataru - be found*, *Owntracks*, and *GPS Plus* identify users by their device IDs. *Waze* allows users to use their service anonymously. However, to share their locations with contacts, the anonymous users would need to log in.

Since ScheduleME is built on top of openPDS, the key differentiator in the privacy mechanisms is that ScheduleME does not allow either the app or contacts (as in peer-to-peer communication) to view or track a user's raw location data. Thus, unlike *PrivateMeetUp* and *Tag - You're It*, ScheduleME enables users to schedule meetups without requiring them to reveal either their actual locations or points of interest. By either extending the functionality or adding new features, ScheduleME can provide functionalities comparable to those of the apps listed in the table.

- *Skout* and *Blendr*: unexpected or serendipitous discovery of users can be accomplished by comparing the bounding boxes of users and alerting them when an overlap occurs.

- *Waze* and *GPS Plus*: traffic navigation can be simulated by collecting and using additional sensor data (like activity, Wi-Fi, etc.). By combining such data from multiple users, it is theoretically possible to simulate traffic in a crowd-sourced fashion.

- *Glympse*, *Miataru - be found*, and *Owntracks*: tracking of users can be simulated in a privacy-preserving manner by randomly selecting a location from the bounding boxes of the top locations visited by a user and collected over the past few days. This location can be broadcast to the contacts as an approximate location where the user may be found at this hour.

Thus, we see that it is plausible to extend ScheduleME and provide similar user experiences and functionalities to those of other apps while keeping users' data private.

4.4 Summary

This chapter has discussed the implementation of ScheduleME, with a particular emphasis on the group computation functionality. The privacy mechanisms of ScheduleME were then compared with those of other commercial and research-based apps. We showed how it was feasible to create similar functionalities within ScheduleME without compromising the user's privacy.

Chapter 5

Implementation of MIT-FIT on PrivacyMate

This chapter presents the MIT-FIT lab which was implemented as part of this thesis. Through this lab, we demonstrate how to build a privacy-preserving fitness-related lab on the MIT Living Labs project. The MIT Living Labs initiative is planning to release the MIT-FIT lab in the coming months. MIT-FIT uses mobile and sensor data for performing individual and group computations in privacy-preserving ways. MIT-FIT differs from other fitness apps and devices which store users' data on the app provider's servers and can potentially share the data with third parties.

This chapter discusses the design decisions and technical details of the MIT-FIT lab as well as a comparison of MIT-FIT with other commercial fitness-based apps. We discuss how users can achieve privacy assurances on MIT-FIT without sacrificing the desired utility obtained in other apps.

5.1 MIT-FIT Design

The MIT-FIT lab was conceived out of discussions involving some of the following questions for health and wellness: How can we better promote fitness and wellness? Is exercise correlated with performance in other aspects of student/employees performance? What are the patterns of use of MIT's athletic facilities? The roundtable discussion further informed

our design and implementation decisions and helped us identify a few design and functional requirements for the lab. Possible extensions to the lab are listed in Chapter 7 as future work.

5.1.1 System Architecture

The system architecture of MIT-FIT is shown in Figure 5-1. MIT-FIT currently requires the *activity* and *location* data for its functionalities. These data can be captured from the mobile devices via Funf or potentially fed to the system from APIs provided by the manufacturers of wearable devices. It is conceivable to collect additional types of fitness and wellness data (such as calories burned during a workout) from the wearable devices. Each participant's PDS instance computes answers to questions requested by MIT-FIT through openPDS' *Question and Answer* framework. Individual and group computations of the answers take place on and among the PDS instances of different participants. The results of the computations (visualizations and recommendations) are then sent to MIT-FIT to be displayed to the participant.

5.1.2 Goals

MIT-FIT aims to allow users to (i) track personal and community activity levels and (ii) interact with others by sharing and subscribing to campus-wide fitness-related events.

MIT-FIT makes the following novel contributions to the research on participatory sensing for fitness:

1. Enables privacy-preserving computations and visualizations of data in the context of fitness. During and after computations, other participants and even the service itself would not have access to any of the participant's raw data.

2. Provides rudimentary spatio-temporal analytics. MIT-FIT delivers basic analytics that are comparable to those offered by the mainstream fitness apps.

3. Motivates the need to investigate richer dimensions of *Quantified Self* [22]. Specifically, MIT-FIT motivates quantifying activities, relationships, and places for fitness.

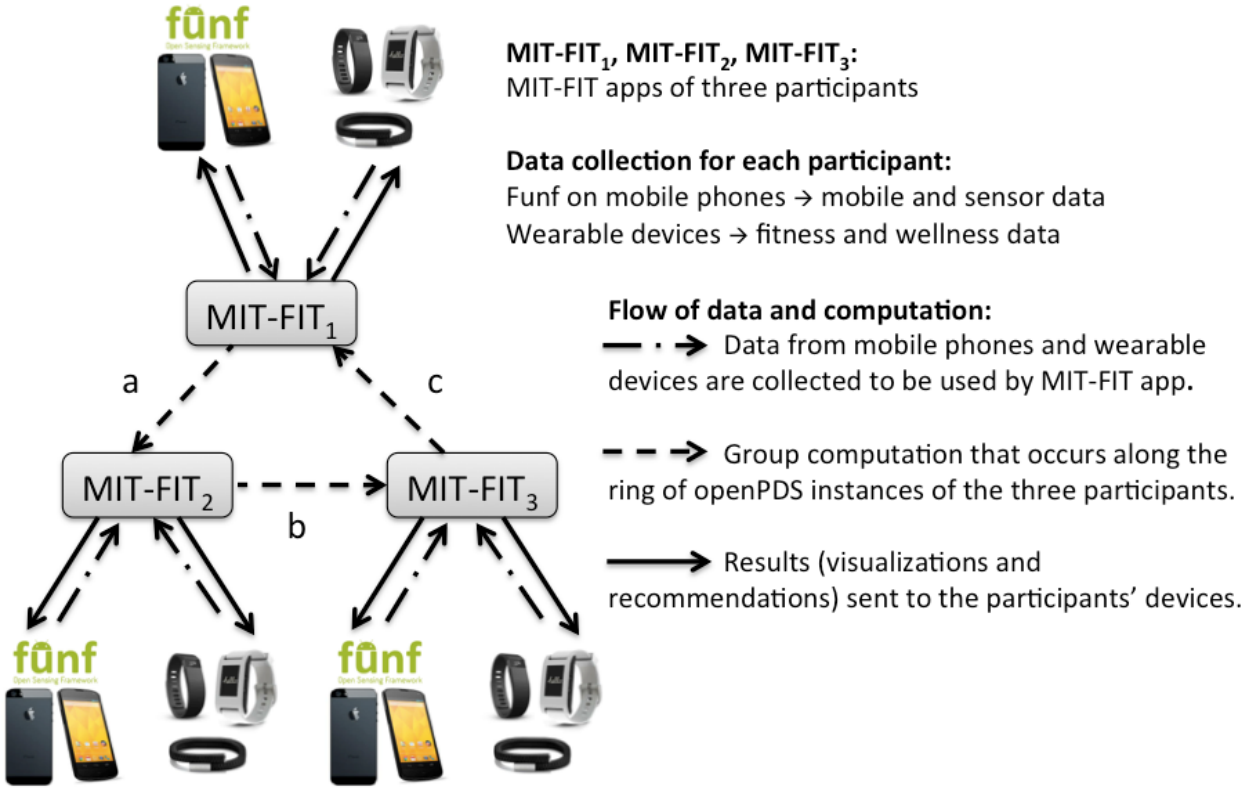

Figure 5-1: *MIT-FIT system architecture.*

5.2 MIT-FIT Implementation

5.2.1 Features

MIT-FIT aims to provides the following features to the MIT community (faculty, students, and staff) to help them adopt or maintain a healthy and active lifestyle.

1. **Analyze activity patterns by location and time** Using the location and activity sensor data from the participant's device, MIT-FIT identifies places and times of high activity for that participant. These regions are then rendered using a heatmap (Figure 5-2) while the times are displayed as a bar chart (Figure 5-4). MIT-FIT also performs a group computation to determine aggregate places of high activity for all the participants and renders the corresponding places in a separate heatmap (Figure 5-3).

2. **Fitness-related recommendations** Upon receiving a list of fitness-related events from MIT's recreational center, MIT-FIT can sort those events based on whether a

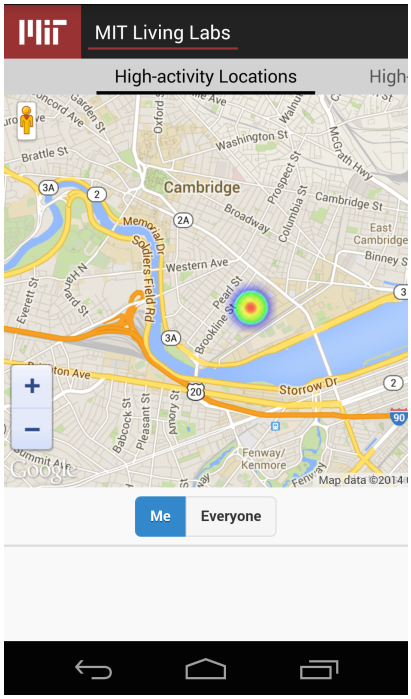

Figure 5-2: *Locations of high activity for the user.*

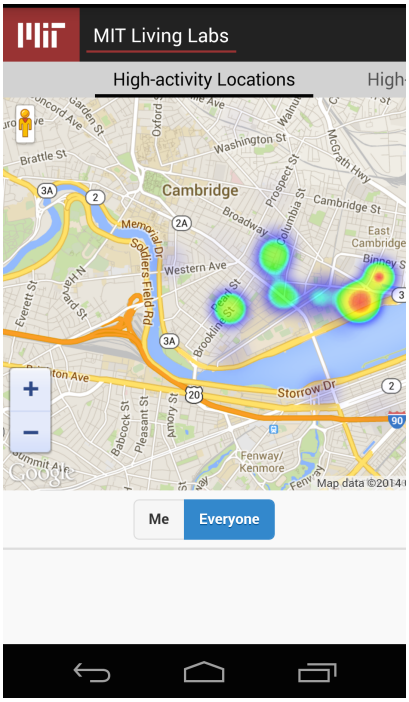

Figure 5-3: *Locations of high activity for all users.*

particular user is inherently active at that particular time or location. This sorted list would result in personalized activity recommendations for the user. Figure 5-5 shows the screenshot of such recommendations for a user.

5.3 Related Work

This section discusses how MIT-FIT differs from commercial fitness-related apps.

5.3.1 Factors used in comparison

We use the same format as in Chapter 4 to compare the privacy-preserving mechanisms of MIT-FIT with other commercial apps. However, for the fitness-based quantified-self apps, we can ignore *Real-time location tracking* and *Framework structure* in the comparison. Since the field of quantified-self relies primarily on fetching as much sensor data as possible for analyses, as expected, all the apps listed in Table 5.1 track users' data. Also, the providers of the apps inherently offer centralized solutions.

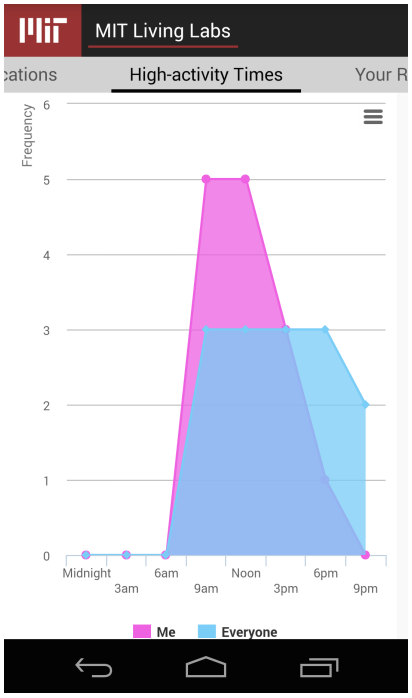

Figure 5-4: *Frequency of high activity times for the user.*

5.3.2 Comparison with MIT-FIT

Table 5.1 lists the commercial apps that provide fitness-related services. *Fitbit*[1] and *Pebble*[2] are primarily smart watches/devices that also offer supporting mobile apps. *Nike+*[3], *Moves*[4], *RunKeeper*[5], and *Strava*[6], on the other hand, are primarily mobile apps. *Fitbit* provides a wide variety of tracking and visualization services for fitness, nutrition, and sleep. *Pebble* and *Moves* primarily track movement, while *Nike+*, *RunKeeper* and *Strava* specialize in running and biking activities.

Even though we ignore tracking and framework structure in this comparison, the following are interesting exceptions to the status quo. *Moves* offers the following time durations as options for users to turn off tracking: 1 hour, 3 hours, up till next charge, or completely. Further, some fitness apps and devices such as *Pebble*, *Moves*, and *Fitbit* allow users to access

[1] http://www.fitbit.com/
[2] https://getpebble.com/
[3] https://secure-nikeplus.nike.com/plus/
[4] http://www.moves-app.com/
[5] http://runkeeper.com/
[6] http://www.strava.com/

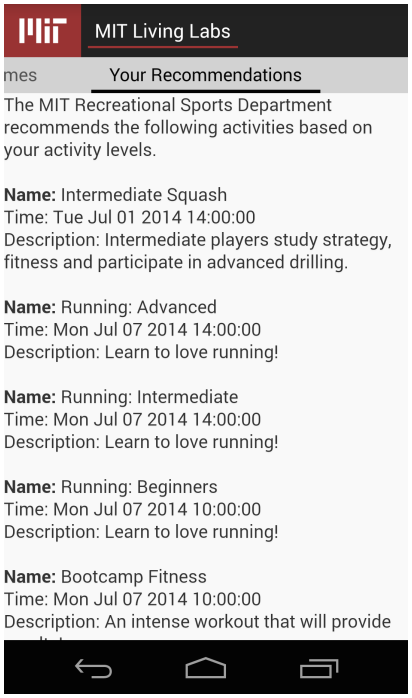

Figure 5-5: *Personalized recommendations of fitness-related events to users.*

APIs or provide links to download their data.

Most of the apps that permit sharing user data and analyses on social networks enable sharing on Facebook. *Nike+* and *RunKeeper* also allow sharing on Twitter (with *Nike+* further allowing users to share on Pinterest). Almost all apps allow users to register new accounts to access their service. *Pebble* and *Moves* use the user's device IDs to authenticate the users and allow them to use their respective app or smart-device. *Nike+*, *RunKeeper*, and *Strava* allow users to log in with their social network profiles.

Moves provides anonymous usage of the app, while *Strava* enables users to anonymize their names by initializing their lastname in leaderboards. Additionally, *Strava* has the option to only allow "*Strava* athletes" to follow and see photos, and only allow approved followers to view activities on profile. These privacy features are available via *Strava*'s "Enhanced Privacy" switch which is turned off by default. *Pebble* and *Fitbit* both provide APIs to allow users to download and access their own data or to allow developers to provide services for users who share their data. As expected, both *Pebble* and *Fitbit* have safeguards in the form of well-defined policies for privacy. Additionally, *Pebble* allows developers to

APP NAME	VISIBILITY (SHARING)	PRIVACY TECHNIQUES	LOG IN ACCESS	
			NEW	S.N.
Fitbit	F. T.	policy safeguards for developer and API access	Yes	F. G.
Nike+	F. T. P.	grouping to share data and setting up private challenges	Yes	F.
Pebble	F.	policy safeguards for developer and API access	Device	-
Moves	Private	can use app without account	Device	-
RunKeeper	F. T.	grouping to control sharing of data and analyses	Yes	F.
Strava	F.	"Enhanced Privacy"	Yes	F.
MIT-FIT	None	individual private store, question and answer framework, group computation	Yes	N/A

Table 5.1: *Commercial & research quantified-self apps, showing the visibility within each apps, the privacy techniques used to safeguard users' location information, and the type of log in access (F. Facebook, G. Google+, T. Twitter, FS. Foursquare, P. Pinterest, O. Others)*

develop companion apps for the "*Pebble* appstore" and its policies cover this additional scenario as well. While most apps allow sharing data and providing leaderboards, *Nike+* allows users to create "events" (e.g., a 5k run) and share them privately with contacts on the app. It thus provides this unique feature while taking privacy into consideration. Finally, *RunKeeper* allows grouping of contacts (only me, friends, everyone) to help users selectively share their data and analyses.

By either extending the functionality or adding new features, MIT-FIT can provide similar functionalities to the apps listed in the table.

- *Pebble*, *Moves*, and *Fitbit*: since these apps provide an API to access users' data, the data generated and collected by them can be downloaded to users' PDS instances and similar analyses can be computed.

- *Nike+*, *RunKeeper*, and *Strava*: by using Android's *DetectedActivity* class, we can extend Funf to identify the type of activity a user is involved in (such as running, walking, driving, etc.). It would then be feasible for MIT-FIT to identify and map a user's run, hike or bike ride.

Thus, we see that it is plausible to extend MIT-FIT and provide similar user experiences and functionalities as other apps while keeping users' data private.

5.4 Summary

This chapter has discussed the implementation of MIT-FIT, with a particular emphasis on the system architecture. The privacy mechanisms of MIT-FIT were then compared with those of other commercial apps. We showed how it was feasible to create similar functionalities within MIT-FIT without compromising the user's privacy.

Chapter 6

User Experience with PrivacyMate and MIT-FIT

This chapter describes how a user would interact with PrivacyMate and MIT-FIT. The chapter starts with the various workflows a user might go through and then discusses the effect of using PrivacyMate's privacy mechanisms on MIT-FIT.

6.1 Workflows of User Experience

When interacting with PrivacyMate and MIT-FIT, users go through the following workflows: the first time set-up (walkthrough) and subsequent setting update.

6.1.1 First time set-up (walkthrough)

When a lab is installed on the user's device and PDS instance, the user is greeted with a walkthrough that introduces the lab and guides the user through the process of creating his or her access control preferences.

The walkthrough starts with a brief introduction explaining the goals and functionalities of the installed lab. Users are then taken to the settings screen where they see the *opt-in to data collection* and *opt-in to data aggregation* controls. They can select the data that they would like the lab to use and indicate whether or not they would like to participate in

group computations performed by that lab. Next, they are taken to the context definition screen. As shown in Figure 6-1, they see a list of pre-defined contexts (MIT and Alltime-Everywhere). At this point, users can choose one of the following three options: create a new context, edit or delete an existing context, or finish the walkthrough.

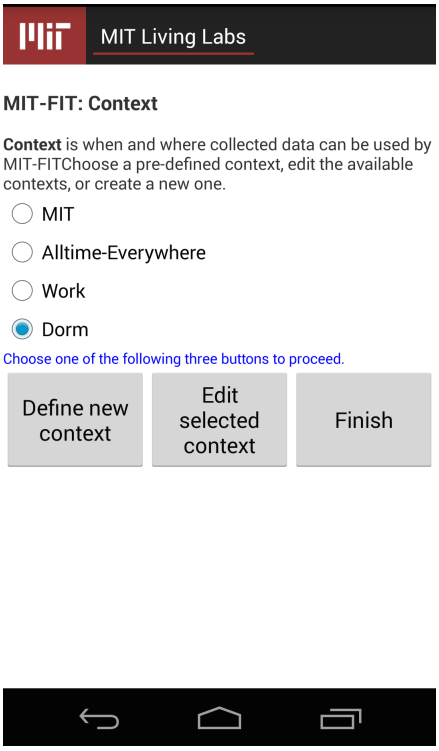

Figure 6-1: *The context home screen showing the two pre-defined contexts "MIT" and "Alltime-Everywhere" and the option to create new custom contexts.*

- **Create a new context:** Users can create custom contexts by specifying their desired temporal and spatial specifications for the context under which they are willing to let the lab use their data. Users are first asked to specify times of day and days of the week to indicate their temporal constraints. Next, if they choose to, they can specify locations by selecting regions on a map to indicate their spatial constraints.

- **Edit selected context:** Users can edit the temporal and/or spatial specifications for any given context (both pre-defined as well as user-defined). Though users cannot delete either of the pre-defined contexts (MIT and Alltime-Everywhere), they can

delete any of the user-defined contexts. PrivacyMate has this restriction because a preference should consist of both settings and context for the enforcement to work.

- **Finish:** If users indicate that they would like to finish the walkthrough, they are redirected to the lab's homepage.

6.1.2 Subsequent setting update

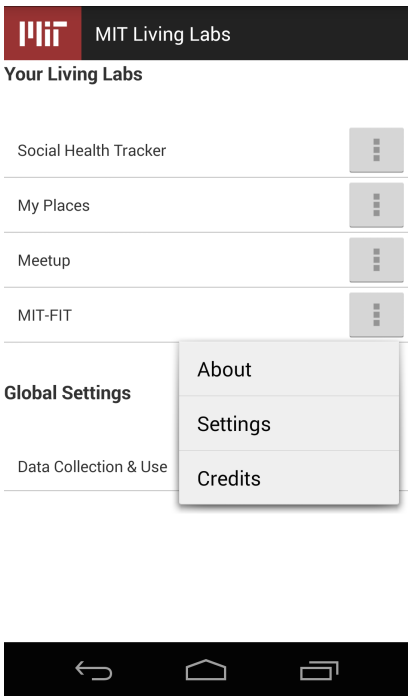

Figure 6-2: *Menu beside each of the labs listed on the Living Labs app. The menu options currently are: about, settings, and credits.*

For subsequent access and use of the access control preferences, PrivacyMate provides a drop-down menu (as shown in Figure 6-2) next to each lab's name on the MIT Living Lab home screen. After clicking on the "settings" item in the drop-down menu, users are redirected to a workflow similar to the *walkthrough*, except that the first introductory page describing the lab is not shown. The introductory page is still accessible for subsequent use via the "About" and "Credits" items in the menu as shown in Figure 6-2.

6.2 Access Control Enforcement for MIT-FIT

After a user creates access control preferences during the walkthrough for the MIT-FIT lab, openPDS enforces those preferences on the PDS instance. When the MIT-FIT lab (code) on the user's PDS instance asks "questions" of the server, these preferences are used as a filter to determine whether a given location or activity data object is allowed to be used by MIT-FIT. These checks are hidden from the users and, from their perspective, they only view the results displayed by MIT-FIT.

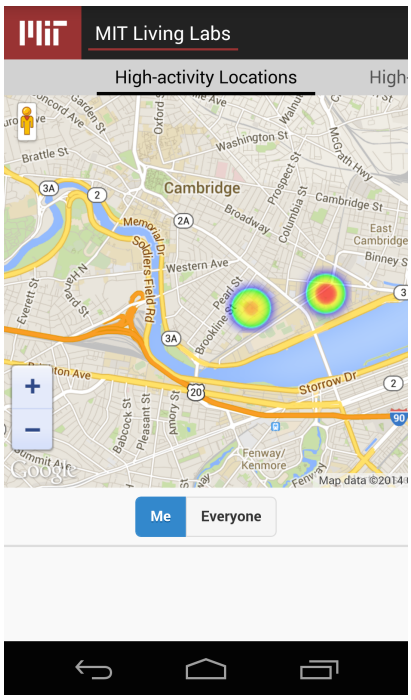

Figure 6-3: *High-activity locations for a user:* Alltime-Everywhere *context*

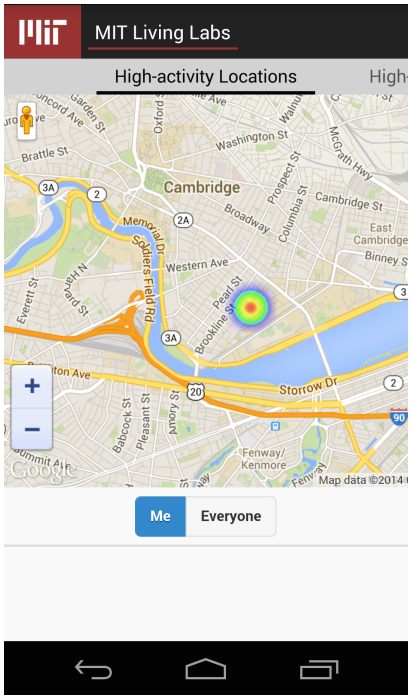

Figure 6-4: *High-activity locations for a user:* Dorm *context*

In order to demonstrate the differences in functionality that different access control preferences can cause, we create two preferences for a user and evaluate MIT-FIT on both. Both preferences allow the required data (location and activity) to be collected and used. However, one of them (*preference1*) uses the *Alltime-Everywhere* context, while the other (*preference2*) uses the *Dorm* context. The MIT-FIT visualization of high-activity locations is shown for both the preferences in Figures 6-3 (*preference1*) and 6-4 (*preference2*). As can be observed from the screenshots, the more relaxed context (*preference1*) causes the lab to obtain more data from the user's PDS instance than the comparatively more restrictive

context (*preference2*). Further, note how the visualization obtained from the filter corresponding to *preference2* is centered around a specific location. This is because the spatial specification of *preference2* indicated that region as permissible (as shown in Figure 3-7). Thus data collected by Funf when the user was present in other locations was blocked by the filter corresponding to *preference2* for the MIT-FIT lab.

6.3 Usability Consultation and Semi-Structured Interviews

We tested the usability of PrivacyMate over the following two phases: *usability consultation* and *semi-structured usability interviews*.

Phase 1 – Usability consultation: We visited the MIT IS&T Usability team [1] for a consultation about the usability of the user controls for privacy. Their overall suggestion was to conduct a user study to understand the potential issues in the usability of the framework.

Phase 2 – Semi-structured usability interviews: We then conducted a few semi-structured usability interviews.

6.3.1 Semi-structured usability interviews

The details of these interviews are presented next.

Goal

The goal of the semi-structured interviews was to better understand the usability of the user controls for privacy and to obtain suggestions for improving them.

Procedure

We started the interviews with a verbal "walkthrough" of the lab to each participant. We then asked them to perform three specific tasks. The participants were then asked to rate their interaction with the framework when accomplishing each specific task. Their rating

[1] http://ist-prod-web-1.mit.edu/usability/

was based on a five-point Likert scale with the following options: *Very difficult*, *Difficult*, *Neutral*, *Easy*, *Very easy*. They were then asked to justify their feedback for each rating. Finally, we asked them for any final comments and feedback.

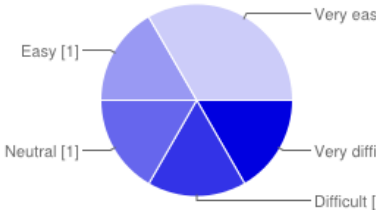

Figure 6-5: *Five-point Likert scale ratings for task 1*

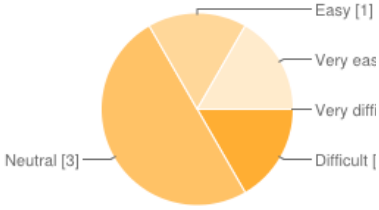

Figure 6-6: *Five-point Likert scale ratings for task 2*

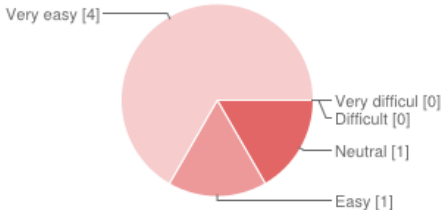

Figure 6-7: *Five-point Likert scale ratings for task 3*

Feedback

We now discuss the feedback we received for the three tasks during the semi-structured interviews.

Task 1: *Scenario: Suppose that you want to allow Social Health Tracker to collect and use all required data. How would you accomplish this task?*

Three participants mentioned that the "three-dot icon" to access the settings was ambiguous. Most of them tried to access the settings by clicking on the Social Health Tracker lab name present in the list of labs (see Figure 6-2). However, that action took them to the home screen of the lab's functionality and not the data permissions screen. After investigating the screen for sometime, most of the participants were able to figure out how to access the data permissions and context screens. While performing task 1, a couple of participants asked about the opt-in to data aggregation functionality. Participant P1 asked "Is it sharing my data? I don't want my data to be shared." In addition to this additional control present on the screen, the wording on the screen appeared to be unclear. Two participants mentioned that they did not understand the wording. Participant P3 mentioned about not willingly reading text on mobile devices unless required to. One suggestion was to make the UI more intuitive to navigate and understand without requiring users to read and parse the text. Participant P6, on the other hand, said that they "experienced very similar app settings and therefore could do it." Thus, depending on what the participants focused on,

they had varying experiences about usability of the data permissions screen and we received a somewhat uniform distribution of feedback for this task (Figure 6-5).

Task 2: *Scenario: Suppose that you only want Social Health Tracker to use your data when you are at home. How would you accomplish this task?*

Two participants asked if they could select more than one context for a given lab at the same time. Two other participants mentioned that they are at home at different times during the weekends compared to weekdays, but the UI does not allow them to indicate that. The overall feedback about task 2 was that the concept of context was not directly apparent and that no information was provided on the "Data Permissions" screen that users could access and set their context after pressing the "Continue" button. Specifically, participant P3 mentioned "... [context] is hidden. It can be discovered by accident or remembered." Participant P3 further mentioned a downside that the temporal-spatial parameter values of the various contexts were not visible until the temporal and spatial screens were visited. The lack of indication that the context created was indeed applied to the lab was also pointed out. For a context's spatial parameter, participant P5 mentioned that it "would be nice to give an address and have the map drop a pin, [thereby doing] something similar to how people find places on Google maps." Further, participant P5 also mentioned that enabling the users to specify the diameter of the location circle would be helpful. Thus, the majority of the responses were "Neutral" (Figure 6-6) since creating the context of "home" was somewhat usable but felt that the controls should be improved.

Task 3: *Scenario: As you may have noticed, there are other apps available in addition to Social Health Tracker. Suppose that you wanted to allow data collection and use by all apps at the same time (instead of doing it app by app). How would you accomplish this task?*

Task 3 about the global settings seemed to be more intuitive to the participants with four of them saying that the task was "Very easy (Figure 6-7)." Participant P1 mentioned that "this task was easier because I found Global Settings right at the beginning." Thus, the fixed sequence of tasks could have enabled some participants to accidentally discover the Global Settings. Participant P2 liked this UI better than the one presented for the previous tasks because "this [showed the] data collection and use [controls] on the same screen, which is good." Participant P4 found this task to be easier than the previous two because the

number of steps required to accomplish this task was less than what was needed in the previous two tasks. An interesting observation made was that only participant P3 used the "Turn on required" option while all the other participants used the "Turn on all" option to turn on data collection and use for all the labs. Participant P5 mentioned that the task itself was easy, but the controls were confusing because (i) no feedback about the action's success, and (ii) no specification about what data was being collected was provided after the action was performed. Participant P3 commented that once "Turn off all" was selected, there was no way to turn on one or two specific data probes except by visiting the individual labs. Participant P3 mentioned about not knowing "what applications have what data" since information about which labs needed those data was not presented.

6.4 Summary

This chapter has discussed the user interactions with PrivacyMate and MIT-FIT. It also demonstrated the difference in the visualizations rendered by MIT-FIT when different contexts were defined. Finally, semi-structured usability interviews were presented along with the feedback we received.

Chapter 7

Conclusion

This thesis has described a way for the MIT Living Lab project to incorporate user-controllable privacy mechanisms for mobile personal data. Specifically, PrivacyMate accomplishes this task through its suite of tools via the *opt-in to data collection*, *context definition*, and *opt-in to data aggregation* preferences and their subsequent enforcement. We also described the roundtable discussion conducted to gather requirements for the development of a fitness-related lab, MIT-FIT. The design and implementation of the ScheduleME and MIT-FIT labs were then discussed. Finally, the MIT-FIT lab was tested to understand how it would function with different types of context definitions. The MIT Living Lab team is planning to eventually deploy PrivacyMate and MIT-FIT to the entire MIT community.

7.1 Future Work

Looking forward, the following improvements would be beneficial to PrivacyMate and MIT-FIT:

- Users should be enabled to apply multiple contexts at the same time to a lab. This would need careful checks for boundary conditions.
- Since labs can potentially inherit or extend the code of other labs, we need to investigate the issue of loose coupling versus inheritance, especially with regards to data collection and usage.

- Based on the roundtable feedback, we need to (i) investigate various means of forming groups via dorms or housing and (ii) integrate data from other QS apps and devices into openPDS and labs.
- Basing recommendations on relationships and places, rather than just activity, could be investigated.
- MIT-FIT functionality might be improved by classifying and considering different types of activities. The heatmaps and charts could then also be filtered based on the type of activity.
- It would be good to conduct another usability study on a larger scale.

Chapter 8

Appendix

8.1 Requirements Gathering: Roundtable Questionnaire

Following are the questions that we selected and asked the focus group participants (see Section 2.4 above):

1. Circle the top three words that convey the meaning of wellness to you. activity, fitness, sleep, play, work, stress, health, productivity, well-being, mindfulness

   (a) Look at the the list provided in the last page. Which of the quantified-self apps or devices do you currently use or have used in the past? (circle those that apply)

   (b) For what purpose(s) did you use those apps or devices? List name of the app/device and purpose.

2. What motivates you to use those apps or devices?

3. Describe how you are using those apps or devices today?

4. Have you stopped using any apps or devices? If so, why?

5. What kinds of privacy concerns do you have with self-tracking apps and devices?

6. Imagine it is 6 years from now, i.e., the year 2020. What would your perfect quantified-self app or device look like and do?

1. Circle the top three words that convey the meaning of wellness to you. activity, fitness, sleep, play, work, stress, health, productivity, well-being, mindfulness

2. Look at the the list provided in the last page. Have you ever thought about using any of those quantified-self apps or devices? (circle those that apply)

3. What has stopped you from using a quantified-self app or device so far?

4. Suppose that a quantified-self app would be released to the MIT community in Fall 2014. What would incentivize you to use that app? Examples: financial, social, fitness goals Please elaborate.

5. What are the kinds of privacy concerns you may have with self-tracking apps and devices?

6. Imagine it is 6 years from now, i.e., the year 2020. What would your perfect quantified-self app or device look like and do?

Type of QS object	List of objects
Gadgets & Sensors	Amiigo, Autographer, Automatic, Beddit, Bedscales, BodyMedia FIT, CarePredict, Cubesensors , Fitbit, Green Goose, HAPIfork, iHealth, Jawbone Up, Looxcie, Lumoback, Med Gadget, Metromile, Misfit Shine, Memoto, Moov, Muse, Nike +, Sano Intelligence, Pebble Smartwatch, Sensoria Smart Sock, Trace, Vicon Revue, W/ME, WakeMate, Withings, Zendrive, Zeo
Apps	Argus, Average Sleep, Bodywise, Cardiio, Digifit, Daytum, Endomondo, Eventflow, Everyday, Ginger.io, Heyday, In Flow, Instant Heart Rate, ITrackMyTime, Kennedy, Lifelapse, Lume Personal Tracker, Map My Fitness, MapMyRun, Moodpanda, Mealsnap, Momento, Moves, Mymee, Nudge - Healthy Living Score, One Second Everyday, OptimizeMe, QuantLove, Quotidian, Quentiq, Reporter, Rseven, RunKeeper, Saga, Shadow, Sleep bot, Sleep Genius, Sleep Cycle, Sleep Time, Stress Check, Tableau Public, Tonic Self Care Assistant, Weight Record, Zen Log
Web Services	Beeminder, Bedpost, Everylog, HonestBaby, Mercury App, Microsoft HealthVault, Moodscope, RescueTime, Sen.se, Slife, Traqs.me, uMotif
Data Aggregation Services & APIs	Bodytrack, Carepass, Exist.io, Fluxtream, Human/API, JoyMetrics, Kantify You, Manybots, Matchup.io, My Fitness Pal, Open mHealth, ProjectAddapp, QuantifiedAPI, Sing.ly, SocialSafe, Syncmetrics, ThinkUp, Tictrac, Zenobase
Smart Journals	Day One, Diaro, Everyday.me, Memories: the Diary, Momento, Narrato, Step

Table 8.1: *Quantified-self apps, devices, etc. The list was accumulated from various sources.*{[123]}

[3]http://lifestreamblog.com/lifelogging/
[3]https://play.google.com
[3]https://itunes.apple.com

Bibliography

[1] Debjanee Barua, Judy Kay, and Cécile Paris. Viewing and controlling personal sensor data: what do users want? In *Persuasive Technology*, pages 15–26. Springer, 2013.

[2] Supriyo Chakraborty, Zainul Charbiwala, Haksoo Choi, Kasturi Rangan Raghavan, and Mani B Srivastava. Balancing behavioral privacy and information utility in sensory data flows. *Pervasive and Mobile Computing*, 8(3):331–345, 2012.

[3] Supriyo Chakraborty, Chenguang Shen, Kasturi Rangan Raghavan, Yasser Shoukry, Matt Millar, and Mani Srivastava. ipshield: a framework for enforcing context-aware privacy. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages 143–156. USENIX Association, 2014.

[4] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.

[5] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013.

[6] Yves-Alexandre de Montjoye, Jordi Quoidbach, Florent Robic, and Alex Sandy Pentland. Predicting personality using novel mobile phone-based metrics. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 48–55. Springer, 2013.

[7] Yves-Alexandre de Montjoye, Samuel S Wang, Alex Pentland, Dinh Tien Tuan Anh, and Anwitaman Datta. On the trusted use of large-scale personal data. *IEEE Data Eng. Bull.*, 35(4):5–8, 2012.

[8] Daniel A Epstein, Alan Borning, and James Fogarty. Fine-grained sharing of sensed physical activity: a value sensitive approach. In *Proc. ACM Intl. Joint Conference on Pervasive and Ubiquitous Computing*, pages 489–498. ACM, 2013.

[9] Deborah Estrin and Ida Sim. Open mhealth architecture: an engine for health care innovation. *Science(Washington)*, 330(6005):759–760, 2010.

[10] Drew Fisher, Leah Dorner, and David Wagner. Short paper: location privacy: user behavior in the field. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 51–56. ACM, 2012.

[11] Saikat Guha, Mudit Jain, and Venkata N Padmanabhan. Koi: A location-privacy platform for smartphone apps. In *Proc. of NSDI*, pages 1–14, 2012.

[12] Samiul Hasan, Xianyuan Zhan, and Satish V Ukkusuri. Understanding urban human activity and mobility patterns using large-scale location-based data from online social media. In *Proc. of ACM SIGKDD Workshop on Urban Computing*, page 6, 2013.

[13] Tanzima Hashem, Mohammed Eunus Ali, Lars Kulik, Egemen Tanin, and Anthony Quattrone. Protecting privacy for group nearest neighbor queries with crowdsourced data and computing. In *Proc. of ACM Ubicomp*, pages 559–562, 2013.

[14] Eiji Hayashi, Oriana Riva, Karin Strauss, AJ Brush, and Stuart Schechter. Goldilocks and the two mobile devices: going beyond all-or-nothing access to a device's applications. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 2. ACM, 2012.

[15] Sangmin Lee, Edmund L Wong, Deepak Goel, Mike Dahlin, and Vitaly Shmatikov. πbox: a platform for privacy-preserving apps. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation*}.

[16] John Lyle, Salvatore Monteleone, Shamal Faily, Davide Patti, and Fabio Ricciato. Cross-platform access control for mobile web applications. In *Policies for Distributed Systems and Networks (POLICY), 2012 IEEE International Symposium on*, pages 37–44. IEEE, 2012.

[17] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An empirical study of geographic user activity patterns in foursquare. *ICWSM*, 11:70–573, 2011.

[18] Norma Saiph Savage, Maciej Baranski, Norma Elva Chavez, and Tobias Höllerer. Im feeling loco: A location based context aware recommendation system. In *Advances in Location-Based Services*, pages 37–54. Springer, 2012.

[19] Fuming Shih and Julia Boortz. Understanding peoples preferences for disclosing contextual information to smartphone apps. In *Human Aspects of Information Security, Privacy, and Trust*, pages 186–196. Springer, 2013.

[20] Brian Sweatt, Sharon Paradesi, Ilaria Liccardi, Lalana Kagal, and Alex (Sandy) Pentland. Building privacy-preserving location-based apps. In *12th Annual Intl. Conference on Privacy, Security and Trust*. IEEE, 2014.

[21] Shomir Wilson, Justin Cranshaw, Norman Sadeh, Alessandro Acquisti, Lorrie Faith Cranor, Jay Springfield, Sae Young Jeong, and Arun Balasubramanian. Privacy manipulation and acclimation in a location sharing application. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 549–558. ACM, 2013.

[22] Gary Wolf, A Carmichael, and K Kelly. The quantified self. *TED http://www. ted. com/talks/gary_wolf_the_quantified_self. html*, 2010.