E-MISSION: AN OPEN-SOURCE SMARTPHONE PLATFORM FOR COLLECTING HUMAN TRAVEL DATA

K. Shankari
shankari@eecs.berkeley.edu

Mohamed Amine Bouzaghrane
amine.bouzaghrane@berkeley.edu

Samuel M. Maurer
maurer@berkeley.edu

Paul Waddell
waddell@berkeley.edu

David E. Culler
culler@berkeley.edu

Randy H. Katz
randy@cs.berkeley.edu

Word Count: 5632 words + 6 figure(s) x 250 + 1 table(s) x 250 = 7382 words

Submission Date: October 24, 2018

Abstract

GPS-equipped smartphones provide new methods to collect data about travel behavior, including through travel survey apps that incorporate automated location sensing. However, prior approaches to this have involved proprietary or one-off tools that are inconsistent and difficult to evaluate. e-mission is an open-source, extensible software platform that consists of (i) an app for survey participants to install on their Android or iOS smartphones and (ii) cloud-hosted software for managing the collected data. e-mission collects continuous location data, user-initiated annotations, and responses to contextual, platform initiated survey questions.

New studies can be set up using the existing U.C. Berkeley infrastructure with no additional coding, or the platform can be extended for more complex projects. This paper reviews the requirements for smartphone travel data collection, describes the architecture and capabilities of the e-mission platform, and evaluates its performance in a pilot deployment. The results show that the platform is *usable*, with over 150 installations in a month; *stable*, with over 85% of users retaining it for more than 3 days; and *extensible*, with interface and survey customizations accomplished in a little over a week of work by a transportation engineering researcher. We hope that e-mission will be a useful tool for app-based data collection and will serve as a catalyst for related research.

Keywords: Modeling, Behavior change, GPS, Smartphones, Software platforms, Travel behavior, Travel surveys

INTRODUCTION

The rapid adoption of GPS-equipped smartphones is transforming data collection for travel behavior research and analysis. As of 2016, 77% of all U.S. adults, and 92% of adults under 30, own smartphones ((**author?**) (*19*)). Passive records of smartphones' approximate locations are compiled by service providers (as well as by certain location-based app platforms), but purpose-built travel survey apps remain the best way to obtain detailed data. Travel survey apps allow researchers to collect high-precision GPS traces and pose contextual, on-device survey questions for data validation or to gather supplementary information. Technical challenges include a lack of standardization in smartphone sensing systems, shorter battery life than stand-alone GPS devices, and the complexity of building a comprehensive software platform.

This paper introduces e-mission,[1] an open-source platform for collecting prompted, user-reported, and automatically sensed travel data from smartphones. It consists of an app for survey participants to install on their Android or iOS smartphones and cloud-hosted software for managing the collected data. e-mission improves on existing tools by being entirely open, modular, and extensible. This provides two important benefits: (i) its algorithms for collecting sensor data, managing power drain, and processing GPS traces can be fully documented, benchmarked, and reproduced; and (ii) project-specific modifications to the software are easy to implement and re-use. This is consistent with the case for open computing programs for reproducible research outlined in **(author?)** (*12*). At the same time, e-mission already provides an extensive suite of functionality and can be quickly deployed for new studies that follow a standard template.

Travel survey data collection

Transportation planners and researchers use data from travel surveys to build predictive models of travel behavior and infrastructure needs. Typical surveys collect information about trip origins, destinations, purposes, timing, travel modes, routes, and other related information, using paper-, phone-based, or electronic tools. This human-based data collection may contain errors and biases, but is ideal for understanding people's perception of their own travel ((**author?**) (*25*)).

Technologies like GPS can reduce respondent burden while providing more precise, accurate, and complete records of survey participants' travel. Travel surveys increasingly supplement self-reported information with automatically sensed location data from stand-alone GPS devices (e.g., **(author?)** (*24*), **(author?)** (*17*)). However, these devices have their own drawbacks, such as expense, that make them difficult to use at scale ((**author?**) (*18*)).

e-mission is part of a new category of smartphone-based tools that combine the expressiveness of surveys with the detail and precision of location sensing. Most smartphones now have GPS chipsets, as well as other sensors like accelerometers that can facilitate travel mode detection. Smartphone-based data collection can provide better data quality and better ease of use for survey participants, at lower overall cost than stand-alone GPS devices.

User engagement

Survey participants must be recruited to studies that use the e-mission platform in the same manner as a traditional survey, but the platform includes a number of features designed to reduce enrollment friction and keep participants engaged in the study. e-mission facilitates on-boarding through (i) direct installation of the app from standard app stores, (ii) optional study-specific interface cus-

[1] https://e-mission.eecs.berkeley.edu; https://github.com/e-mission

tomization, and (iii) a clean user experience.

It also includes features to facilitate long-term user engagement through *information provision* or *gamification*, for studies where this is appropriate. Personal travel analytics may appeal to users who are interested in physical activity or environmental sustainability, or just curious about their own mobility patterns. Gamification through personal targets or social competition can make these apps into tools for behavior modification (e.g., **(author?)** (*13*), **(author?)** (*14*)), and experiments in this area are the topic of active research (e.g. in **(author?)** (*4*)). These features can be disabled in cases where they could interfere with a study.

Related work

A 2014 TRB report (**(author?)** (*25*)) provides the most extensive review to date of approaches for collecting and analyzing GPS data to study travel behavior. They identify key challenges for smartphone data collection, including: (i) *market fragmentation* (different mobile operating systems and hardware capabilities make it difficult to collect equivalent data from all survey participants); (ii) *power drainage* (continuous collection of GPS data will rapidly drain a smartphone's battery); (iii) *costly data plans* (cellular data transmission may not be feasible); and (iv) *sampling biases* (ownership of smartphones varies by age, income, and education).

Many studies have used smartphones to collect travel data, typically falling into three categories: (i) automatically generated travel diaries that avoid the errors and biases of self-reporting (e.g., **(author?)** (*5*), **(author?)** (*20*)); (ii) behavior modification based on gamifying travel and providing incentives for particular mode choices (e.g., **(author?)** (*13*), **(author?)** (*14*)); and (iii) understanding human perceptions by building route choice models for active transportation modes such as bicycling (e.g., **(author?)** (*11*), **(author?)** (*2*)).

Technological advancement and increasing market penetration of smartphones are leading to steady progress on each of the four challenges identified in the TRB report. However, we identify a fifth, related challenge: *collection platform robustness*. The complexities of cross-device data collection, power management, and data analysis are best addressed by open, modular, extensible software platforms that encourage widespread adoption. Such platforms can be easily deployed for new projects and reliably benchmarked and adapted as technologies change. Importantly, open-source software can improve reproducibility and provide an opportunity for scholars and practitioners to build a collaborative platform that is controlled by the community.

The remainder of this paper is organized as follows. The next section describes the e-mission platform architecture and data collection capabilities. The subsequent section describes usage and extensibility, and the final sections evaluate pilot deployments, identify future work, and conclude.

SYSTEM ARCHITECTURE AND COLLECTED DATA

The core functionality of the e-mission platform is to *collect* and *assemble* travel data. In this section, we identify key data requirements and briefly describe the architecture of the software platform. Important categories of data are automatically sensed information, user-initiated reports, and platform-initiated requests such as survey questions. For further use, e-mission assembles the raw data into travel diary components, personalized tour models, and other meaningful outputs.

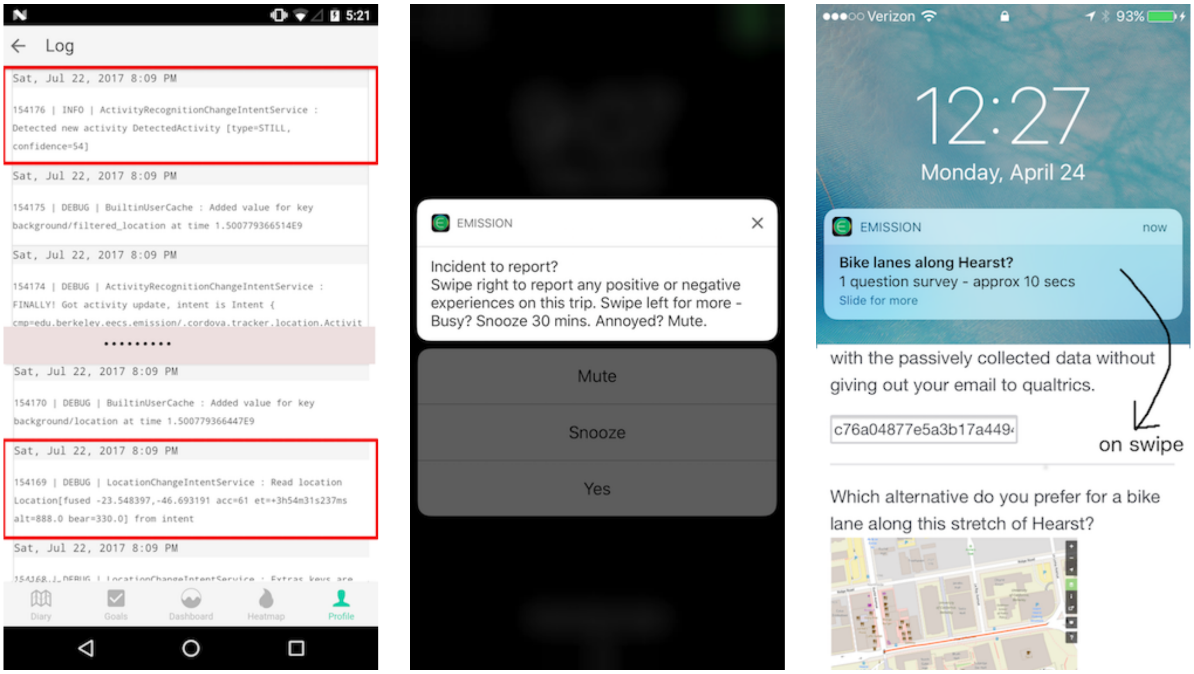

FIGURE 1: Supported data collection types. Left to right: automatically sensed, user initiated, platform-initiated

Categories of human travel data

We divide human travel data into three broad categories, based on the technical requirements and user experience of collecting it from a smartphone(Figure 1).

1. Automatically sensed

This represents data, such as location, accelerometer, or microphone readings, obtained automatically from smartphone sensors without any user intervention. Since this data is obtained automatically, it does not represent a cognitive burden on the user and can be collected in large quantities. However, a naive approach of reading data at high frequency from all possible sensors will lead to significant power drain, and represent its own burden on the user — their smartphone may become unusable during the course of the day. Therefore, the data collection processes need algorithms that can strike a balance between data quality and power drain.

Further, this data is typically not useful in itself; inference algorithms need to be run on top of it to generate useful insights. Multiple inferences can be drawn from the same set of base data — for example, accelerometer data can be used for both road quality (e.g. in **(author?)** (*9*)) and for travel mode detection (e.g. in **(author?)** (*10*)). However, such inferences are inherently inaccurate, so the inference algorithm needs to be able to quantify its uncertainty, and any action on the inference needs to take this into account.

2. User-initiated

This is data that the user is motivated to report based on his/her surroundings. It is typically perceptual and cannot be inferred by sensor data alone. Examples could be: “the sidewalk here feels empty,” “a truck has blocked the bike lane,” and so on. Open-ended perceptual data has not

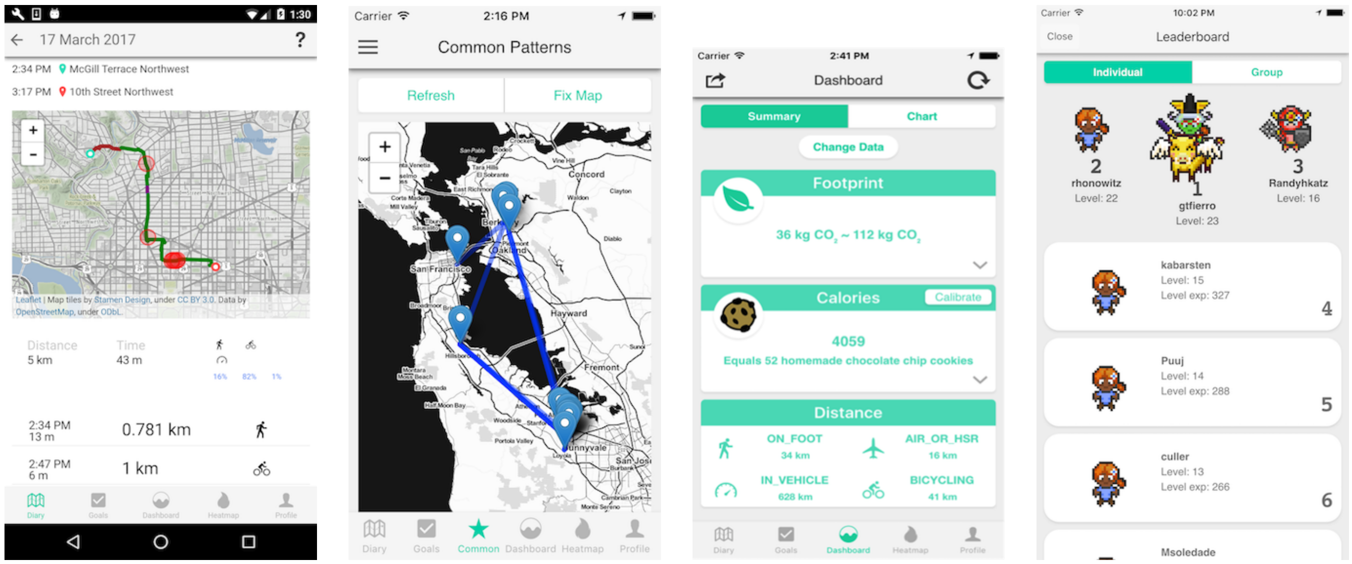

FIGURE 2: Supported outputs. Left to right: travel diary, tour model, statistics, game leaderboard

typically been integrated into transportation engineering studies but its usage is growing, largely because of data generated on smartphones (e.g., **(author?)** (*6*)).

This type of data is currently collected primarily by commercial projects, such as (i) proprietary issue reporting and tracking systems deployed by local public agencies (e.g. SeeClickFix, Comcate, etc.), (ii) proprietary real-time automobile incident reporting (e.g. Waze), and (iii) rating systems for points of interest (e.g. Yelp). Including a qualitative component in travel data collection has the potential to provide a richer understanding of human behavior, while supporting new research areas related to data correctness, bias, and heterogeneity of experience (e.g. the four types of cyclists described in **(author?)** (*8*)).

3. Platform-initiated

This is data that is requested from the user by the platform, such as survey questions. One use of requests is to increase the accuracy of inferred data. Examples of such requests are: (i) to obtain *ground truth* for inferences to boost their accuracy; and (ii) to obtain confirmation of unexpected behavior. However, requesting large amounts of ground truth re-introduces cognitive load on the user. Ground truth acquisition needs to balance accuracy and cognitive load, especially for long-term data collection.

Practitioners can also initiate requests to obtain additional information from a targeted audience. Examples of such requests include: (i) obtaining stated preference data about proposed changes from households that will be affected by them; and (ii) obtain additional demographic information (e.g. bicycle availability) from sub-populations based on their travel patterns (e.g. no recent bicycle trips). These requests are not necessarily tied to sensed or inferred data, and can be fairly complex.

Supported outputs

The e-mission platform can process collected data into a number of standard outputs.

1. Travel diary

This output is the canonical analysis result. Every one of the prior projects from the literature features a basic travel diary. It is also the building block for the other outputs, so should be considered a core component of a smartphone travel data collection platform.

A travel diary is a linked sequence of *trips* between *places*, each potentially split into *sections*. Each section is associated with a travel *mode* and each trip is associated with a travel *purpose* or *activity*. (We suggest using *purpose* to reduce ambiguity because *activity* can have other meanings in a travel context, as in **(author?)** (*27*) and **(author?)** (*1*).)

The mobile systems community (since **(author?)** (*26*)) and travel survey community (since **(author?)** (*23*)) have developed several algorithms for automatic mode inference. These typically use GPS information alone ((*26*), (*23*)) or a combination of GPS and accelerometer data, as in **(author?)** (*21*). They typically support a small number of modes (e.g. walk, bike, car, bus, train) and have accuracies ranging from 70% to 80%. There are fewer algorithms for inferring trip purpose, since it is typically not reflected in sensor readings. Most automated approaches such as **(author?)** (*16*) rely on land use and point-of-interest databases, and the accuracy for locations other than home, work and school is below 70%.

2. Personalized tour model graph

This output analyses the trip diary to generate a graph of the common trips taken by the user. This graph is effectively an amalgamation of tour models, similar to the work in **(author?)** (*22*), but with common locations among the tours represented by single nodes in the graph. Collapsing a long sequence of trips into a single graph allows analysis and modeling to focus on a small number of representative trips. For example, detailed semantic data gathering such as stated preference questions can focus on trips in the tour, and reduce user burden. Generating probability distributions over attributes of the common trips (e.g. start and end times) converts it into a Markov model that can be used to potentially predict future behavior, as in **(author?)** (*7*).

3. Game/motivation

This output uses the travel diary to generate personalized statistics to motivate travelers to think more carefully about their travel patterns and associated choices. Some projects, such as **(author?)** (*15*), calculate personalized calories burned, carbon footprint, and cost for the traveler. Such projects also typically compare the personalized value to various aggregate statistics to reinforce norm setting, as in **(author?)** (*13*). Other projects, such as **(author?)** (*14*) or **(author?)** (*3*), use gamification techniques such as badges, levels, and challenges to encourage long-term behavior change.

Software architecture

e-mission follows a sensor–server–client architecture that is standard for Internet of Things (IoT) applications, where everyday devices are used as digital instruments. In particular, the smartphone app is both the sensor and a client, since personalized information can be viewed on the phone. The server handles communication, long-term storage, data processing and aggregation. While a detailed description of the architecture and the related work is deferred to a forthcoming paper, we sketch the components and their interaction in this section.

The phone app has a hybrid architecture built using the Apache Cordova mobile app framework. Native plugins for Android and iOS (written in Java and Objective-C) sense *location* and

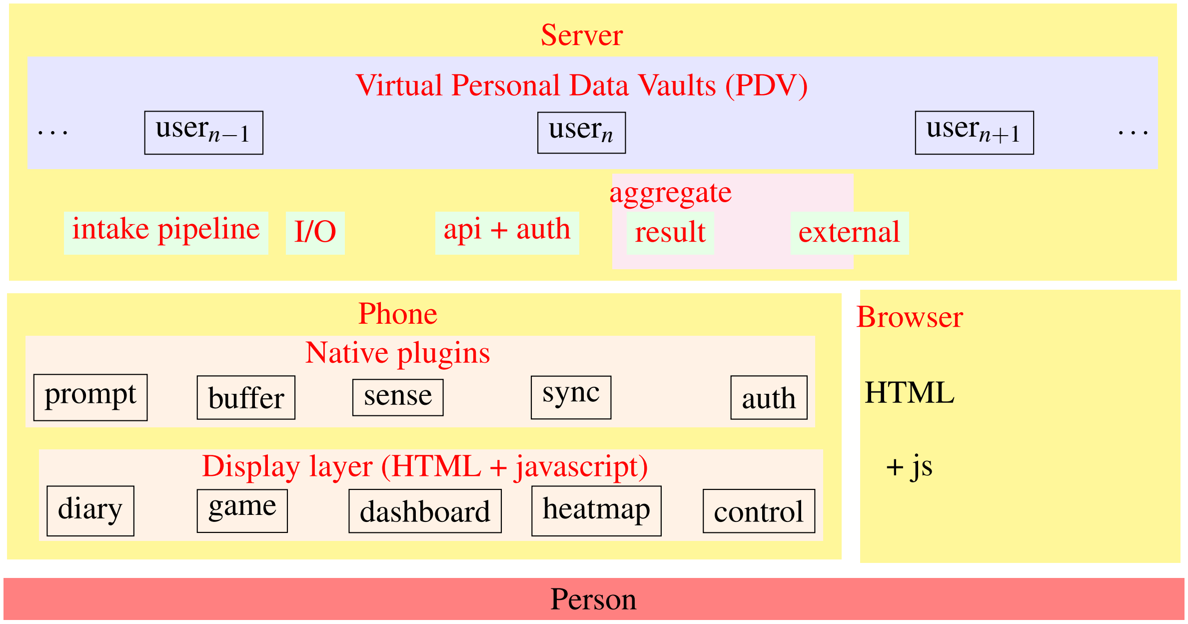

FIGURE 3: Components of the e-mission architecture

motion activity and buffer the data on the phone in a SQLite database. The sensing is automatically turned on at trip start and turned off at trip end to reduce battery consumption. Buffered data is synced to the server after the end of the trip. The trip start and end can also prompt a configurable notification to collect additional information.

The server software is written in Python for ease of extensibility by non-experts. It makes heavy use of Python scientific processing libraries such as scikit-learn, and exposes a REST API for client interaction. It receives data into an input cache, and then saves it into a user-specific section of a shared datastore. The user-specific datastore consists of multiple timeseries, one for each type of object (e.g. background location, manual incident, etc). The newly arrived data is then run through a pipeline that generates the applicable outputs. Travel diary information can also be queried for individual or aggregate statistics.

Finally, certain outputs are displayed as Dynamic HTML views. The view in the phone client can displays personalized information such as the trip diary, the tour model, and the user's current status in the game. A web app provides a visual display of aggregate, non-personal, usage data.

USAGE AND EXTENSIBILITY

In addition to being full-featured, a successful software platform for smartphone data collection must be easy to extend so that it can meet the needs of a variety of projects. Small configuration changes should be easy, and more significant additions to functionality should be achievable using well-defined extension points. Ideally, these changes should be made publicly available for reuse and reproducibility (**(author?)** (*12*)).

Usage without customization

If the standard e-mission interface and functionality meet the needs for a study, the practitioner can simply file a research protocol with her institution's review board (IRB) and specify that she will use the e-mission platform for background location data collection. (This is similar to specifying the use of a platform like Qualtrics to collect survey responses.)

The practitioner would then instruct participants to download the e-mission app from the Android or iOS app stores, and obtain separate consent from the participants according to the method specified in the protocol. This consent would need to include the email address that the participant uses to register in e-mission, in order to confirm which users are associated with the study. At the end of the study, the practitioner would show the consent documents to the e-mission lead researcher [2] and receive a copy of the data from those users.[3] Full in-app consent can be done with simple UI customization; see below.)

Thus, practitioners can collect automatically sensed location and motion activity data without writing any code, simply by directing survey participants to use the app.

Extending the smartphone app

Easy: Customizing the user interface (UI)

Many practitioners will want to customize the user interface of the app: to add a study logo, to add custom consent, or remove unneeded features. This can generally be done with HTML and CSS changes alone, although functionality related to message prompts involves Javascript.

Because the UI is built using web components, it can be updated without deploying a new app to the stores. The e-mission platform supports multiple UI *channels*, meaning that practitioners can ask survey participants to install the standard e-mission app and then switch to the study-specific channel. A channel can be selected in the UI or by following a special URL or QR code. As soon as a user joins the channel, they are presented with study-specific information, consent, and login choices.

Such extensions are shared with the community as new branches on the "e-mission-phone" GitHub repository.[4]

Medium: Extending the phone app using existing plugins

e-mission is built using the Apache Cordova mobile app framework, which allows easy re-use of existing plugins. Functionality like reading a user's calendar or allowing users to take photos can be added in this way. Cordova plugins are controlled using Javascript.

A phone app that has been extended through the addition of new plugins cannot be updated via the UI channels. Instead, a new app would need to be submitted to the stores with a new name and signing key. For iOS, the app must pass the App Store review process. The resulting app would have no obvious connection to the e-mission platform — it could have its own logo, and would be marked as owned by the organization that is submitting it.

Code for such enhancements can be made available to the community by forking the "e-mission-phone" GitHub repository and pushing changes to the fork. Once the project is complete, the enhancement could even be added to the standard e-mission app (in a new UI pane,

[2] K. Shankari, shankari@eecs.berkeley.edu

[3] The standard e-mission consent document is available here: https://e-mission.eecs.berkeley.edu/consent

[4] https://github.com/e-mission/e-mission-phone

for example). This would be done by submitting a pull request to the master branch of the "e-mission-phone" repository.

Hardest: Writing a new native plugin

Some projects may want to use sensing capability that is not currently supported in the Cordova ecosystem, for example by integrating with a sensor that measures stress from sweat, or using ambient noise to determine whether a car trip is shared or not.

This would require writing native code (in Java and Objective-C or Swift) that reads the appropriate sensors, buffers them, and performs the inference either on the phone or on the server. Such projects can reuse the authentication, buffering, and communication components of the e-mission platform. They can also use the notification component to obtain additional information from the user.

Integration with the e-mission platform would allow the new travel data to be placed in a spatio-temporal context without having to re-write the location tracking and post-processing components. On iOS, restrictions preclude most sensors from being read in the background, but using the e-mission platform would allow plugins to attach themselves to the location tracking callbacks in order to read other sensor data.

Such an extension can be shared with the community by structuring the code as a Cordova plugin and publishing it on GitHub. Projects can then add the plugin like any other.

Extending the server functionality

Easy: Adding queries or analyses

Aspects of the server software not related to the core outputs are structured as plugins, where new functionality can be added by simply writing a standalone Python script. Some examples are queries to find users who are targets for platform-initiated surveys or notifications to inform users about things related to their travel patterns. New analyses can be added to e-mission by generating a pull request from a fork of the "e-mission-server" repository.[5]

Medium: Modifying data pipelines

The existing pipelines for creating travel diaries are open to improvement. Practitioners may want to modify the segmentation, smoothing, or mode inference algorithms used by the core platform. (These pipelines are versioned in GitHub and can be reproduced at any point on a practitioner's own machine. e-mission always retains the original raw data alongside any pipeline outputs.)

These improvements will be more complex to integrate into the core platform, because we need to ensure that they are empirically valid and enough of an improvement to make the default. So while these changes can be contributed using a standard pull request, additional testing will be required before the changes can be merged.

Hardest: Running a custom server

Some projects may have data storage and privacy requirements that differ from the core platform and are best achieved by running their own server. Projects that need special external integrations — with an Open Street Maps editor, for example — would also want to run their own server. Projects that modify the core data pipelines could also run a custom server to avoid integrating their changes with the core e-mission platform.

[5] https://github.com/e-mission/e-mission-server

The e-mission server software can run on any Linux, macOS, or other Unix-like system. However, to manage a production backend, you need to be comfortable setting up SSL, obtaining the correct keys for authentication, and monitoring the pipeline logs for errors. Changes to the server software can be shared with the community by publishing the forked code so that it can be used to inform other projects that require similar integrations.

EVALUATION

In this section, we evaluate the performance of the platform in two main areas. First, we evaluate its *usability* and *stability* using metrics from a pilot deployment with more than 100 users. Second, we assess the *extensibility* of the platform through a qualitative and quantitative evaluation of the effort required for a non-expert to modify it.

It is also important to note what we do not evaluate: the travel behavior captured by the platform during the pilot. This is because the goal of the pilot was not to generate generalizable results about travel behavior, but to evaluate the use of the platform as a tool to enable others to generate such results. We also do not currently evaluate the accuracy of data collection or trip diary creation, which will be the focus of a future paper. Finally, our goal with this pilot was to assess the installation process and technical stability under varied user interaction patterns. In the upcoming year, we hope to partner with researchers who want to use the platform in their studies. This will allow us to generate usability metrics across more representative populations.

App usage metrics

The e-mission platform was launched in a pilot deployment on the U.C. Berkeley campus in Fall 2016, covering all the categories of data and outputs described earlier.[6] Participants were not compensated in cash or in kind. The pilot study was linked to an initiative to encourage walking and bicycling to campus. There was no dedicated marketing team or marketing budget associated with the pilot — all publicity was done by researchers associated with the platform. Recruitment was done through email to campus mailing lists, and there was no official endorsement of the pilot as a university initiative.

Installation rate

This metric captures the rate at which users signed up during the pilot. Ideally, we would use metrics from the app stores to measure this, but the iOS store only reports metrics from users who have opted in to share statistics, so it is not very accurate. Instead, we use calls to the profile creation API endpoint [7], which is invoked when a user first launches the app (Figure 4). This is not a perfect metric because it includes app *re-installs* but it is close (Table 1a). We use calls to the game registration endpoint [8] to detect when a user signed up for the game.

This metric is important because recruitment for traditional travel surveys, and for human-subject research in general, is time-consuming and expensive. While this platform does not claim to solve all problems with recruitment, painless installation ensures that there are no barriers to adoption once participants have been recruited.

[6] To aid reproducibility, the Jupyter notebook used to generate these results is available at https://github.com/e-mission/e-mission-eval. The underlying data can be obtained from the corresponding author, subject to restrictions on use.

[7] `/profile/create`

[8] `/habiticaRegister`

These results show that (i) the app was installed by more than 150 participants; (ii) the installations continued for a month after the initial recruitment, presumably through word of mouth; and (iii) the gamification was interesting to only 50% of users (Table 1a).

Length of install

This metric evaluates the *stickiness* of the app by measuring the number of days the app was installed. Since the phone app automatically uploads data to the server periodically, we use calls to the data upload API endpoint [9] as a proxy for the app being active. The install duration for a particular user is thus the length of time between the first and last API call. This does not distinguish between a user having suspended tracking, and having no trips for a particular period, so a user who reported exactly two trips 10 days apart would have an install duration of 10 days. If the last call was during the final two days of the analysis period, we assume that the app was not uninstalled.

This metric is important because longer-term data collection enables researchers to capture variability in daily travel patterns. Most GPS-enabled household travel surveys now cover multiple days: for example, the wearable GPS component of the 2012 California Household Travel Survey (CHTS) spanned 3 days (**(author?)** (*17*)). Since the recruitment for the pilot was ad-hoc and no compensation was provided to participants, we expect that the duration of data collection would be robust if the app is used for classic travel surveys.

The results (Figure 4) are promising. More than 85% of users had the app installed for at least 3 days. 23 users (not shown in the histogram) still had the app installed at the end of the analysis period. Of the users whose install duration is known, half had the app installed for $\approx$ 20 days, and a third had it installed for over a month. There were also 16 users who had the app installed for just one day.

App launches over time

This metric measures user engagement with the app. It tracks two related metrics: *app launches* and *screen switches* (the latter measures how many times a user switched between screens while using the app). App launches are measured by calls to the server API that populates the dashboard, while screen switches are indicated by client stats.[10]

This metric is important because not all the data is obtained through passive sensors. If we want to interact with the user to capture semantic and perceptual data, we need to have a platform that engages with the user and encourages her to provide the information that we seek. Designing for such engagement is challenging, and one goal of the platform is to facilitate it.

However, given the expectations about novelty in user interaction, the results (Figure 5) are promising. First, they show that although app launches go down after the initial install, they never stop completely, and continue even several months after launch. Second, they show that the distribution of app launches across users is highly skewed — 80% of users have opened the app fewer than 5 times, but 10% of users have opened it more than 150 times. Finally, they show that in addition to opening the app, users consistently navigate to other screens, even months after the

[9] `/usercache/put`

[10] `/results/metrics/timestamp` populates the dashboad on app launch. Note that there is also an `app_launched` client stat, but it does not appear to correspond directly to server calls, so we use the more conservative stat in our analysis. The `state_changed` client stat, filtered to remove changes to and from the `splash` state provides the basis for measuring screen switches.

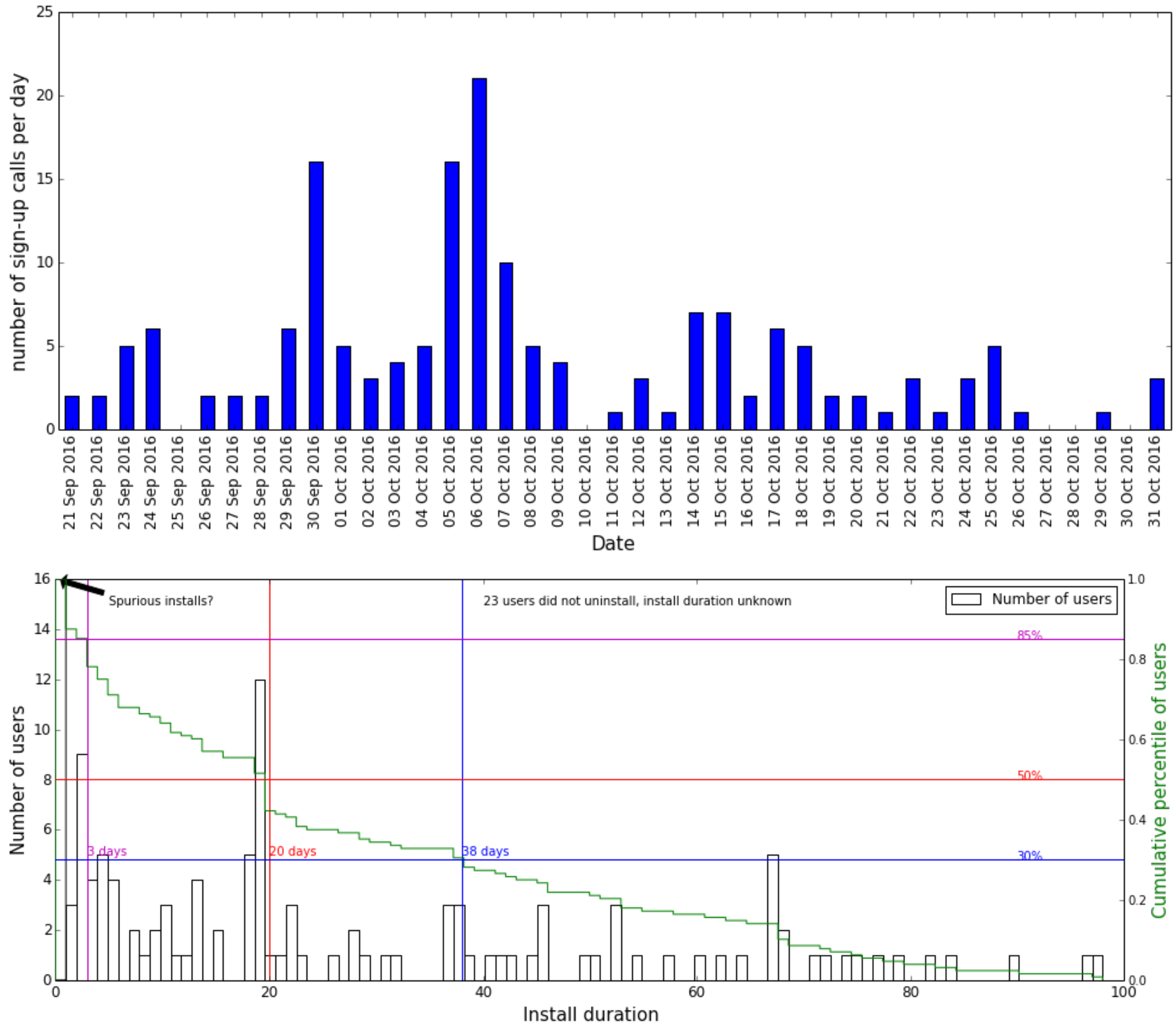

FIGURE 4: Evaluation of the installation rate and duration. Top: Number of calls to the profile creation API per day. This is a resonable proxy for the number of installations, since the profile is created when the user signs in. Bottom: Histogram of user install durations. Install duration is represented by the time duration between the first and last data sync for a particular user. Note that 23 users did not uninstall the app, so their data is not included in the histogram.

	users
Number of calls from unique users	172
Number of sign-ups	170
Number of new clients sending data	151
Number of unique sign-ups for the game	63

(a) Estimates of the number of installations using multiple metrics. Note the ($\approx$ 20) discrepancy between sign-ups and data collection.

Page	Lines changed
CSS Style	+971
Settings	+160
Trip List	-95
Trip Detail	+70

(b) Lines of code modified for each kind of UI functionality changed. Note that the trip list code actually had fewer lines after the changes

TABLE 1: Evaluation metrics for the phone application

install. However, the screen switching is an order of magnitude lower than the app launches. Finally, we see a marked dropoff in both metrics around the end of the study period, which coincides with winter break.

Extensibility metrics

In this section, we evaluate the effort required for a transportation engineering student with no prior front-end experience, specifically in HTML/CSS/Javascript, and who has not worked in app or web development before, to build a custom UI for the app. We use quantitative metrics, such as lines of code, in addition to a brief, open-ended qualitative evaluation of the challenges encountered while completing the task. The results show that less than 1500 lines of code and one and a half weeks of full-time work are sufficient to generate a dramatically different user interface. This includes the learning curve for HTML, CSS and some javascript, and the platform UI in particular. We estimate that the time can be reduced to little more than a day with better documentation or better examples to draw from.

Lines of code

Changes to the UI are shown in Figure 6. These changes consisted mainly of eliminating interface panes, keeping only the main ones (profile and diary). Panes were removed by commenting out the relevant sections of code. The contents of the profile pane were modified as well, along with color schemes, element sizes, and certain icons.

The diary tab gives users access to the list of trips they have taken on each date, and to the details of these trips such as speed profiles and travel modes. This tab was also modified to a new theme that changed the flow and amount of information provided to the user. The "details" page in the modified version displays the trip breakdowns in HTML tables. The new page also includes a button that allows users to fill out a survey about that specific trip. Table 1b illustrates the number of lines of code adjusted for each page in addition to the styles page.

Time required

Another metric to evaluate the amount of effort put into the customization is to measure the amount of time spent understanding the source code, modifying it, and reviewing changes. In this case, getting up to speed with the existing code took approximately 75% of the total time, or 40 hours. Once the source code was understood, implementing modifications was fairly straightforward. Writing new code, debugging it, and testing the results took the remaining 25% of the time, or 15

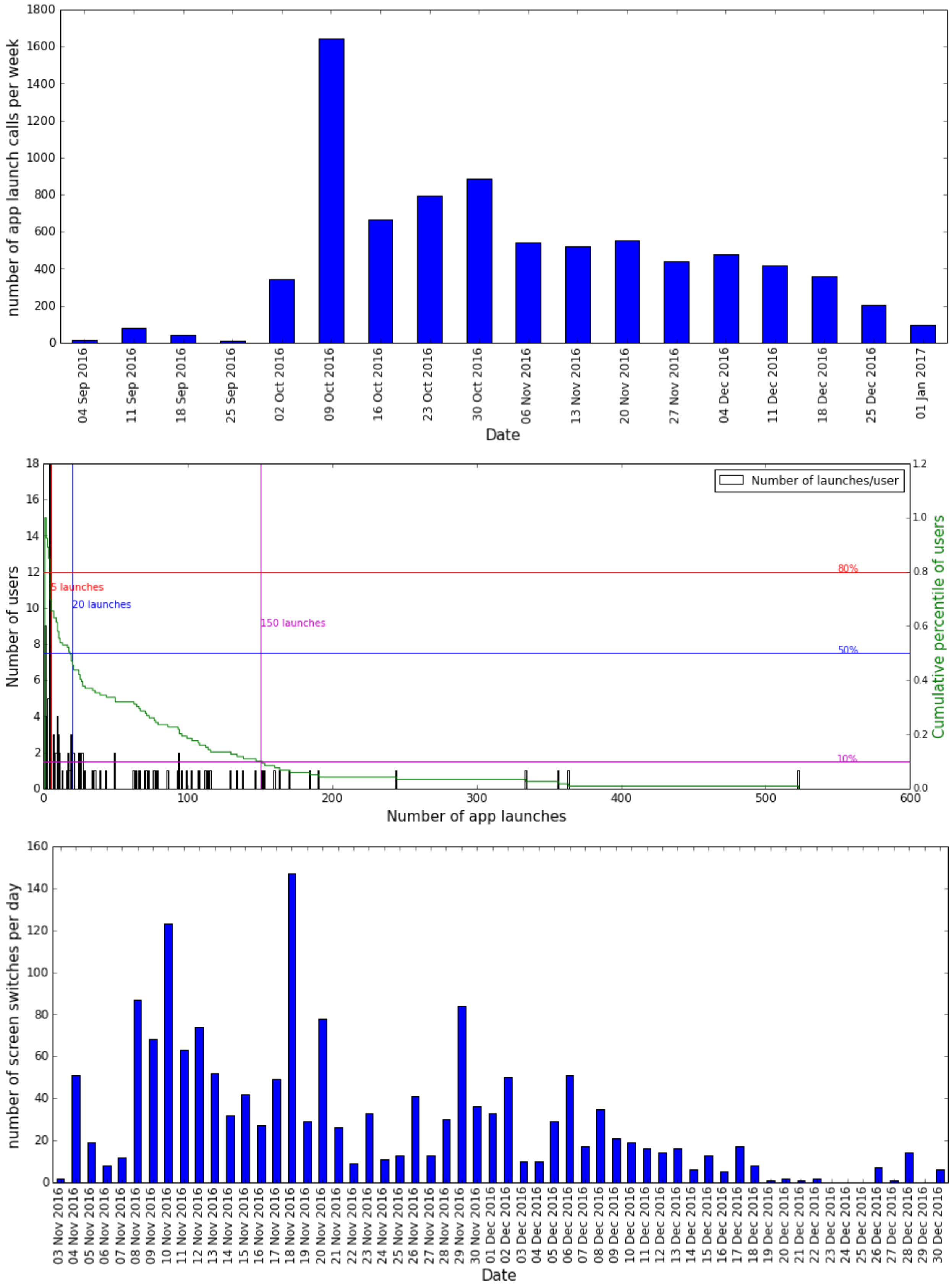

FIGURE 5: Metrics for user interaction and engagement. Top: Number of app launches per week. Middle: Histogram of the number of app launches per user. Bottom: Screen switches per day, starting in November, when we started tracking that stat.

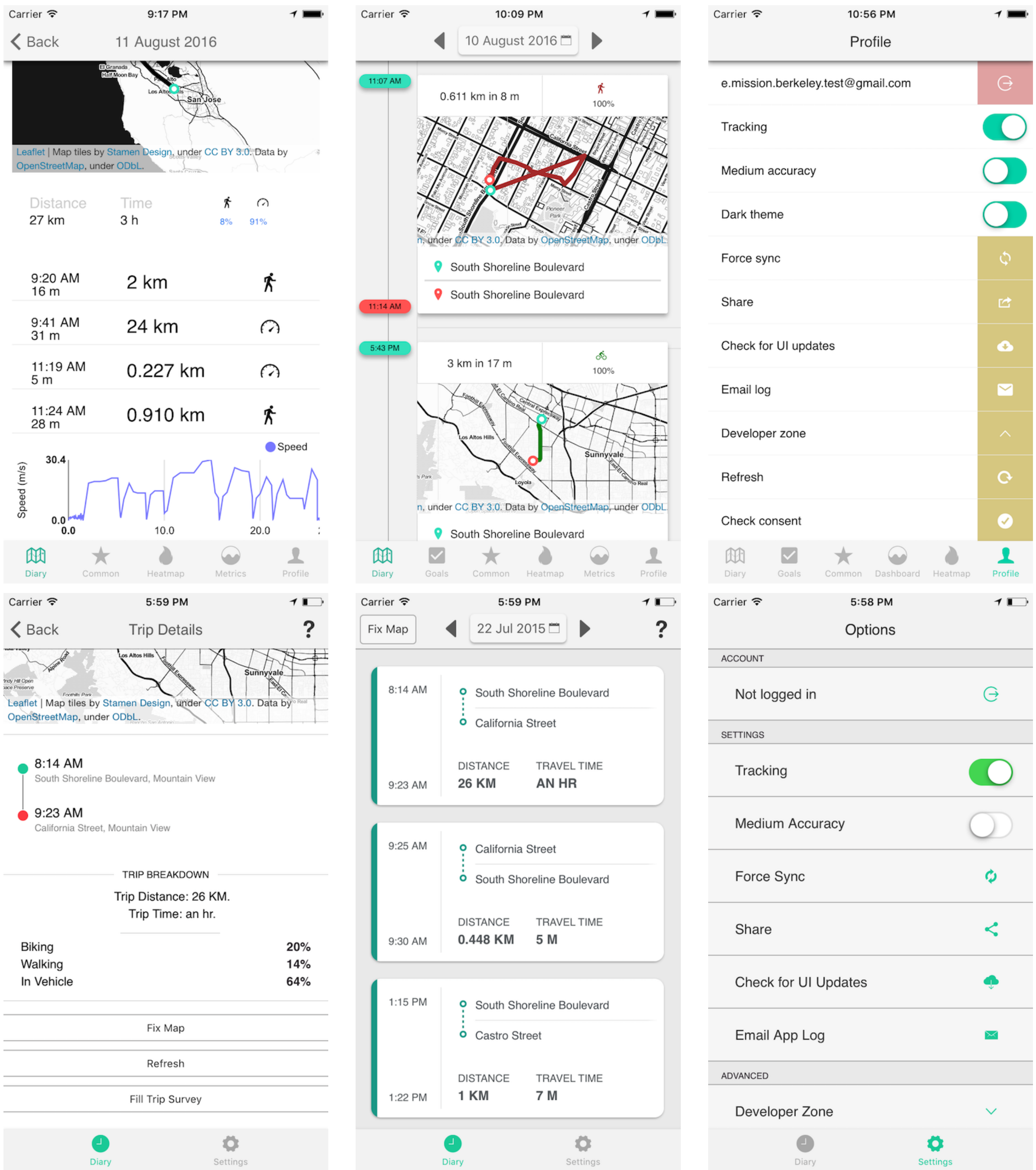

FIGURE 6: Screenshots of customized UI. Top: Base UI on the `master` branch of the `e-mission-phone` repository. Bottom: Customized UI in the `joangroup` branch of the `e-mission-phone` repository.

hours.

Qualitative comments

The major hurdles in customizing the UI were setting up the development environment and understanding the source code. This may have been due to relative inexperience with this type of software development, but improvements to the documentation would assist newcomers in navigating through the project and reduce the time required to make modifications.

CONCLUSION

The e-mission platform[11] aims to make state-of-the-art smartphone travel data collection broadly available to researchers and other practitioners. It supports data collection through (i) background sensing, (ii) user-initiated reporting, and (iii) contextual, platform-initiated survey questions. Its architecture includes native apps for Android and iOS as well as cloud-hosted software for managing the collected data, all of which is modular, extensible, and open-source. e-mission can be used without modification to the interface or functionality simply by instructing participants to download the app (and providing consent documents to the corresponding author, to confirm which users are involved in the study). New UI "channels" can be created with minimal effort, providing study-specific consent forms, branding, and feature combinations. Future development priorities span the data collection and output categories defined earlier. To streamline collection of user-initiated data, we are looking into a "shake-to-report" feature. This would allow users who want to report open-ended perceptual data to shake the phone to report it immediately. In order to guard against false positives, the system can generate a notification for the user to confirm the report. For travel diary creation, we plan to add support for more travel modes, and to evaluate the collected data against various benchmarks. We also plan to improve aspects of the server architecture, including scalability. Additionally, we would like to explore issues of privacy and data ownership, especially in the context of aggregated results.

As described above, improvements to the platform can be contributed by anyone. Valuable independent projects could include an adaptive sampling routine for longer trips or an option to sync data only over WiFi. Finally, we welcome feedback about the potential to assemble general-purpose travel behavior datasets using e-mission. For example, some study participants may be comfortable with a research protocol stipulating that, after a certain time delay, their data be available to outside researchers who agree to restrictions on its use.

[11]https://e-mission.eecs.berkeley.edu; https://github.com/e-mission

REFERENCES

[1] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

[2] J. Broach, J. Gliebe, and J. Dill. Bicycle route choice model developed using revealed preference GPS data. Washington, DC, Jan. 2011.

[3] G. Broll, H. Cao, P. Ebben, P. Holleis, K. Jacobs, J. Koolwaaij, M. Luther, and B. Souville. Tripzoom: an app to improve your mobility behavior. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, page 57. ACM, 2012.

[4] D. Bucher, F. Cellina, F. Mangili, M. Raubal, R. Rudel, A. E. Rizzoli, and O. Elabed. Exploiting fitness apps for sustainable mobility-challenges deploying the goeco! app. *ICT for Sustainability (ICT4S)*, 2016.

[5] Caitlin D. Cottrill, Francisco Camara Pereira, Fang Zhao, Ines Ferreira Dias, Hock Beng Lim, Moshe Ben-Akiva, and P. Christopher Zegras. The Future Mobility Survey: Experiences in developing a smartphone-based travel survey in Singapore. *Transportation Research Record: Journal of the Transportation Research Board*, (2354):59–67, 2013.

[6] C. Coffey and A. Pozdnukhov. Temporal decomposition and semantic enrichment of mobility flows. In *Proceedings of the 6th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, 2013.

[7] M. Dash, K. K. Koo, J. B. Gomes, S. P. Krishnaswamy, D. Rugeles, and A. Shi-Nash. Next place prediction by understanding mobility patterns. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 469–474. IEEE, 2015.

[8] J. Dill and N. McNeil. FOUR TYPES OF CYCLISTS? Examination of Typology for Better Understanding of Bicycling Behavior and Potential. *Transportation Research Record: Journal of the Transportation Research Board*, 2387, 2012.

[9] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

[10] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM Press, 2013.

[11] J. Hood, E. Sall, and B. Charlton. A GPS-based bicycle route choice model for San Francisco, California. *Transportation Letters: The International Journal of Transportation Research*, 3(1):63–75, Jan. 2011.

[12] D. C. Ince, L. Hatton, and J. Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485–488, Feb. 2012.

[13] J. Jariyasunant, M. Abou-Zeid, A. Carrel, V. Ekambaram, D. Gaker, R. Sengupta, and J. L. Walker. Quantified Traveler: Travel Feedback Meets the Cloud to Change Behavior. *Journal of Intelligent Transportation Systems*, 19(2):109–124, Apr. 2015.

[14] A. Jylhä, P. Nurmi, M. Sirén, S. Hemminki, and G. Jacucci. MatkaHupi: a persuasive mobile application for sustainable mobility. pages 227–230. ACM Press, 2013.

[15] V. Könönen, M. Ermes, J. Liikka, A. Lämsä, T. Rantalainen, H. Paloheimo, and J. Mäntyjärvi. Anatomy of automatic mobile carbon footprint calculator. In *Advances in Grid and Pervasive Computing*, pages 84–93. Springer, 2011.

[16] J. Krumm and D. Rouhana. Placer: semantic place labels from diary data. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 163–172. ACM, 2013.

[17] M. Kunzmann and V. Daigler. 2010-2012 California Household Travel Survey Final Report. Technical report, California Department of Transportation, June 2013.

[18] Mick P. Couper, Don A. Dillman, Laura E. Erhard, Paul J. Lavrakas, Steven Polzin, Guy Rousseau, and Clyde Tucker. Expert panel review of the 2016 national household travel survey. Technical report, Federal Highway Administration, Apr. 2015.

[19] Pew Research Center. Mobile Fact Sheet, Jan. 2017.

[20] Philip Winters, Sean Barbeau, and Nevine Georggi. Testing the Impact of Personalized Feedback on Household Travel Behavior (TRAC-IT Phase 2). Technical Report FDOT BD 549 WO 24, National Center for Transit Research, Tampa, Florida, Mar. 2008.

[21] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2):1–27, Feb. 2010.

[22] P. Stopher, Y. Zhang, and Q. Jiang. Tour-based analysis of multi-day GPS data. In *12th World Congress on Transport Research*, 2010.

[23] P. R. Stopher, Q. Jiang, and C. FitzGerald. Processing GPS data from travel surveys. *2nd international colloqium on the behavioural foundations of integrated land-use and transportation models: frameworks, models and applications, Toronto*, 2005.

[24] J. Wolf. Using GPS data loggers to replace travel diaries in the collection of travel data. 2000.

[25] J. Wolf, W. Bachman, M. S. Oliveira, J. Auld, A. K. Mohammadian, P. Vovsha, National Cooperative Highway Research Program, Transportation Research Board, and National Academies of Sciences, Engineering, and Medicine. *Applying GPS Data to Understand Travel Behavior, Volume I: Background, Methods, and Tests*, volume 1. Transportation Research Board, Washington, D.C., June 2014. DOI: 10.17226/22370.

[26] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding transportation modes based on GPS data for web applications. *ACM Transactions on the Web*, 4(1):1–36, Jan. 2010.

[27] M. Zhong, J. Wen, P. Hu, and J. Indulska. Advancing Android activity recognition service with Markov smoother. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 38–43. IEEE, 2015.