# em-arch: A system architecture for reproducible and extensible collection of human mobility data

*Kalyanaraman Shankari*
*Pavan Yedavalli*
*Ipsita Banerjee*
*Taha Rashidi*
*Randy H. Katz*
*Paul Waddell*
*David E. Culler*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-88

http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-88.html

May 20, 2019

Acknowledgement

In addition to NSF CISE Expeditions Award CCF-1730628, this research is supported in part by gifts from Alibaba, Amazon Web Services, Ant Financial, Arm, CapitalOne, Ericsson, Facebook, Google, Huawei, Intel, Microsoft, Scotiabank, Splunk and VMware. This work was also supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

# em-arch: A system architecture for reproducible and extensible collection of human mobility data

January 22, 2019


## Abstract

Smartphones have revolutionized transportation for travelers by providing mapping services that tell users how to get to a specific destination as well as ride-hailing services that help them get there. However, the data collected from these services are limited in temporal, spatial, or categorical scope. For a variety of solutions in urban planning, transportation, or healthcare, collecting rich and granular data of human mobility is critical. Yet, there are few end-to-end, open-source platforms that allow the development of *human mobility systems* (HMS) to collect, access, and leverage these data in a seamless and customized fashion.

We present a novel platform for HMS studies and outline an architecture for such platforms generally. The open-source, extensible data collection platform can be customized to address a wide variety of disciplines. It is validated by usage patterns from three use cases from applied projects. The platform architecture defines the structure of the platform, identifies the key modules and classifies them as core or extensible.

Our use cases used an average of 64% of the features of the platform, with approximately 3-4 months of part-time CS undergraduate time for each new case. Every use case contributed at least one extension, primarily client-related, back to the platform. We hope that the reusability of the platform, combined with the rigor of the architecture will propel the field of human mobility systems (HMS).



**keywords:** system architecture, human centered, mobility, extensibility, usability


## 1 Introduction

It is human nature to inquire about the whereabouts of others, asking, “Where did you go?”, “Which route did you take?”, or “How long did it take you?”. The origins and destinations of travelers, the time it takes to travel, the cost and purpose, and other trip characteristics have historically been collected through travel surveys, which form the practice standard for mobility data [Jean Wolf et al., 2014]. Few entities, even in the era of big data, have the ability to track all of these parameters in detail - Uber includes only ride-hailed trips, Waze focuses on personal car trips, and Facebook gathers only geotags from posts. City planners, transport engineers, healthcare advocates, and gaming gurus, among many others, desire to develop applications based on individual travel diaries, but they currently do not have the capability to do so. There exists neither a comprehensive platform to collect data nor transparent access to the data once collected.

This lack of completeness and transparency in human travel diary collection, despite the bevy of potential applications, has become a major hindrance for comprehensive mobility solutions given the rapidly changing nature of transportation. We believe that this oversight is attributable to the *builder-deployer gap* in this domain. *Deployers*(e.g. mobility researchers) use these systems as tools in their work, focusing on the application, while *builders*(e.g. computing experts) focus on building the systems themselves. We propose an interdisciplinary approach that combines system-building rigor with the concerns of deployers, under the field of *human mobility systems* (HMS).

This paper includes two main contributions.

- It describes a **platform** generalized from three canonical, real-world use cases. The platform includes novel design features to encourage extensibility and reuse. To our knowledge, this is the first such HMS platform in which the applications were developed by groups other than the primary platform builder, and installed by end users on their personal devices. It is also the first such platform evaluated using quantitative metrics.

- It outlines a **architecture** for this class of platforms. The platform architecture is complete, detailed, and end-to-end. It identifies the tiers that typically constitute such platforms, breaks them up into individual modules, determines the design tradeoffs for each module, and classifies the modules as core and extensible.

Why is it so important to include a platform architecture? It is essentially a theoretical description of the range of structures underlying particular platforms for HMS. An architecture shifts focus from the superiority of specific implementations to the general concepts that underlie a class of systems, and the design tradeoffs associated with each (Figure 1). This generalization can be useful to both *deployers* and *builders*. *Deployers* can now have a shared vocabulary to compare different systems in this class, and determine the one that is most appropriate for their needs. Similarly, *builders* can now have a set of small, well-defined modules that they can focus on developing or improving, and a skeleton to put new modules that they develop in context.

The novelty of this architecture lies in its completeness, and provenance. Since the architecture needs to engage both deployers and builders, the particular tiers and modules that comprise the architecture are intentionally **not** novel. The goal is to use concepts that are so conventional that deployers can use Internet resources aimed at a lay audience to build familiarity. The field of HMS can leverage this platform architecture for future platforms.

We recognize that the architecture for HMS presented here may not be the final word, as it is generalized from a small, but diverse, set of use cases. Our main goal is to use our interdisciplinary background to start a discussion around generalizing and evaluating human mobility systems.

In Section 2, we describe the taxonomy of software complexity and generalization. In Section 3, we present some suggestions for continuing and deepening the framework discussion. In Section 4, we examine prior work from both the deployer and builder communities, with a particular focus on their published architectures. In Sections 5, 6 and 7, we outline the platform architecture of the client, server and analysis tiers, respectively. Finally, in Section 8 we evaluate the platform and framework against the

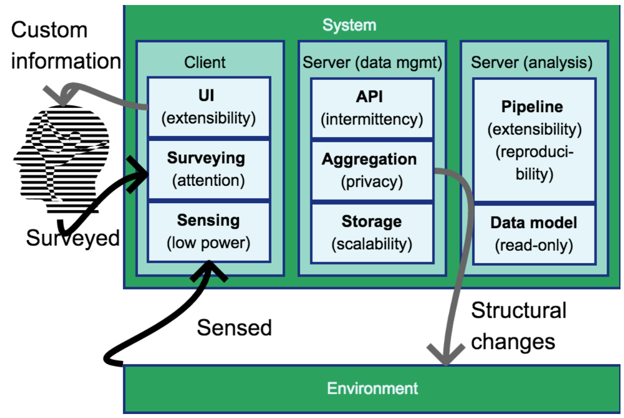


Figure 1: High-level components of the system and their primary challenges. Such systems receive *inputs*(black arrows) from sensors (e.g. travel trajectories) and from end-users (e.g. how they felt during the trip). They can also provides *outputs*(gray arrows) of personalized information to individual users and of aggregate metrics to the public. The aggregate metrics can be used for both short-term (traffic signal control; congestion pricing) or long-term (new roads; new transit line) modifications to the environment.

three canonical use cases - (i) a classic travel study, (ii) a crowdsourcing initiative for accessibility metrics, and (iii) a behavioral study on incentivizing sustainable transportation, before concluding with Section 9.

## 2 Software generalization

After we published our previous paper on ANONYMIE[1] and its usage[2], we noticed that practitioners who were interested in using it would invariably refer to it as "the app" - e.g. *I think that your app is nice but would really benefit from a better user interface.* Since anonymie is actually a platform, we thought that it would be useful to delve deeper into understanding the distinction. In this section, we briefly explore the taxonomy of software complexity (Fig. 2) and generality. The purpose of this breakdown is to understand the differences among existing human mobility data collection solutions and the necessity to position novel concepts into higher layers of the platform.

[1]name changed for double blind review
[2]citation redacted for double blind review

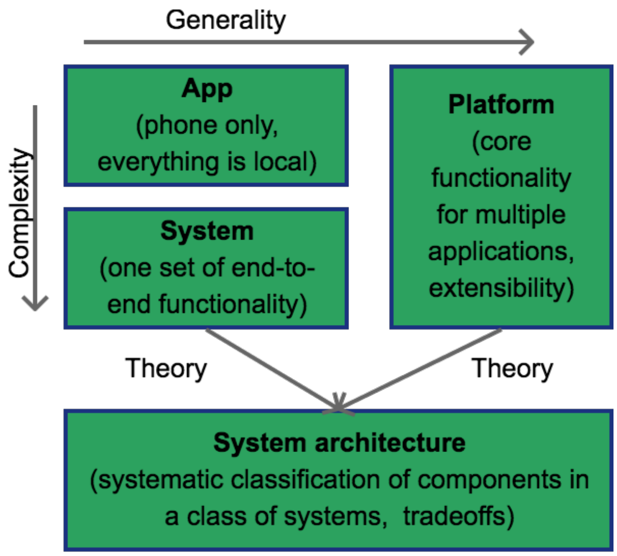


Figure 2: Taxonomy of software solutions wrt complexity and generality

## 2.1 app

An app-only solution involves software that runs locally on the client phone. The app must store data and perform all computation locally. Benefits of an app-only approach include: (i) simplicity, (ii) privacy, and (iii) no/low data usage.

However, the app-only approach is also very limited *because* the data are only local, which restricts accessibility and aggregate analysis. Data accessibility is so critical that even"GPS loggers" developed for personal use (e.g. GPS Logger for android [GPS, 2018], myTracks [myT, 2018]) support emailing the data or uploading to iCloud/Google Drive.

## 2.2 system

An end-to-end system includes both client and server components, in which data are collected locally by a client device but stored and processed remotely on a server. As a result, a system provides a comprehensive solution for one particular application. An example would be a one-off or customized solution for conducting a travel survey [Carrel et al., 2016] or motivating behavior change [Jariyasunant et al., 2015]. While most phone applications are commonly referred to as *apps*, they are actually *systems* with data transmitted and stored on a shared server. The lack of reuse, however, can potentially lead to wasted work. If there are multiple applications that use the same core functionality, it is useful to generalize them into a *platform*.

## 2.3 platform

A platform contains core functionality that is abstracted out from multiple systems and used to deliver a range of systems. End-users of the systems are typically not aware that they are built on top of a platform. Building a platform requires identifying core and extensible components that can be composed into systems. Indeed, *builders* focus on refining and maintaining the core functionality of the platform, while *deployers* extend it to create multiple distinct systems. Platforms are typically motivated by a broad set of use cases. An example platform is ANONYMIE, which is a generalization of systems that combine background sensed and surveyed data to instrument human mobility data.

## 2.4 platform architecture

A platform architecture is a structural representation of the modules of the platform. It identifies the core components of a class of platforms, describes how they are linked, and determines the various possible design choices for each component. Having a well-defined architecture allows platforms to be better understood and extended.

This paper encompasses the platform and architecture levels of HMS. It first outlines a high-level platform architecture for platforms that combine sensing and surveys for collecting human data. Then, it outlines the design decisions made by the ANONYMIE platform for each of the modules in the system architecture, with a particular focus on novel design aspects that aid extensibility and reproducibility.

# 3 Discussion

This paper focuses on a platform for human mobility data, and a platform architecture that describes it. Introducing a platform architecture in addition to a platform shifts the focus from the features associated with one particular platform, to the general concepts that underlie this class of platforms.

However, this is an early version of the architecture, not the last. The platform and architecture are derived from a diverse set of use cases, but the initial set size is small. There are surely other modules that will need to be introduced as other applications are considered. The classification of modules into core and extensible may shift as we consider other platforms with different design choices. This paper seeks to open an interdisciplinary *discussion* with the eventual goal of reusing, understanding, and evaluating HMSes. In this section, we provide some concrete examples of how such a discussion might continue.

### 3.1 Using the platform

The platform architecture and platform were driven by three canonical HMS applications. Other potential applications include:

**Health:** Long been studied by academics [Doherty and Oh, 2012], monitoring health has reached the market of smartphones and watches. Such health monitoring platforms can link the type of activities, contextual factors, if autonomously sensed, to health indicators [Dobkin, 2013].

**Logistics:** The area of supply chain and logistics has requirements similar to HMS. Researchers have explored the use of location/activity sensing platforms to observe the operation of their freight company employees, vehicles [Wang et al., 2014], and/or packages [Morgul et al., 2013]. Customizable geolocation monitoring systems can be used improve the efficiency of freight routing algorithms.

**Offline decision making:** Tracking human mobility provides physical analogues to online tracking. Economists and marketers can observe the real world browsing and selection behavior of shoppers, similar to Amazon. Urban planners can observe the mobility choices of citizens, similar to social media likes. Systems can use personalized options to direct users toward optimized alternatives for various metrics such as price and sustainability.

### 3.2 Integrations

ANONYMIE is an open platform for collecting automatically sensed and surveyed human mobility data. It can also be combined with other platforms to provide services to end-users. For example, it can be integrated with (i) an open source gamification platform (e.g. habitica [hab, 2018]) to use sophisticated motivation techniques, (ii) open source trip planners (e.g. Open Trip Planner [otp, 2018], Digi-Transit [dig, 2018]) in order to provide personalized directions, and (iii) social networks (e.g. Twitter, Facebook, Yelp) in order to share human mobility information effectively. It can also infer activities by consuming information from existing platforms that collect incidental geotagged information (e.g. tweets, posts), although this may require significant text processing [Rashidi et al., 2017].

### 3.3 Comparing platforms

The community can use its architecture and evaluation metrics to compare multiple existing platforms to one another. One potential process by which this might happen is:

1. The community nominates HMS platforms for consideration.

2. The platform maintainers evaluate their individual platforms against the architecture.

3. Maintainers submit a self-evaluation (e.g. Table 3). Each entry could include a simple ✓/×, or a short note on the design decision. If open source, it could also include a link to the relevant code repository, which would allow peer review of the evaluation.

4. In addition, if maintainers identify missing modules, they submit them for inclusion into the architecture.

5. The community collectively determines which modules to include in the architecture.

6. Survey papers are published that includes both peer reviewed and self-reported evaluations for existing platforms, and an updated architecture.

7. Future platform papers would evaluate themselves against the architecture at the time of publication. Perhaps a central document is updated with new platforms as the related papers are published.

Similarly, as new applications (Section 3.1) use platforms, in addition to citing the platform that they used, they can report the development time for their customization and the changes needed. This reporting can help with a more comprehensive evaluation of the utility metric.

### 3.4 App-specific metrics

Geolocation data monitoring survey instruments typically have low response rates, as fine-grained continuous tracking raises privacy concerns while draining battery. It is common practice in the literature to provide > $15 to participate in a data collection project for a week or so, when an unbiased sample representing the population is targeted. To reduce the cash incentives offered, the User Interface (UI) may provide other benefits such as trip planning, health monitoring and market vouchers. Such user interfaces involve multiple possible design choices, and it is not clear which will be most engaging. Research communities in various application domains can compare design choices through targeted user studies, and establish best practices specific to their application.

## 4 Related Work

Human mobility systems form the foundation for multiple different applications. Thus, there is an abundance of systems that are relevant to human mobility. A full listing of these systems is outside the scope of this paper. Instead, we identify axes that define the HMS platform design space and select a sample of the related work that spans it. The related work includes examples of applications corresponding to the individual use cases for our platform (Table 1) and examples of ones that deal with survey data or sensed data or both (Table 2). In the rest of this section, we extract some patterns from these examples, delve deeper into differences from the most closely related platforms, and discuss the choice of projects and features for comparison.

### 4.1 Related work selection

The methodology used to select projects and comparison features for the related work is chosen to find a small, but representative spanning set.

#### 4.1.1 Project choice

Our use cases span popular application domains, so the related work is large. We picked a set of curated papers providing a flavor of the space, using the criteria of openness and novelty.

1. We chose systems from academia since they are more likely to be open. This necessarily excluded proprietary projects such as rMove [Flake et al., 2017, Greene et al., 2016], Google Location History [goo, 2018] or Strava [str, 2018].
2. We chose systems that were novel and varied from other systems in the same group in at least one feature. This avoided overwhelming the analysis with almost identical entries.

#### 4.1.2 Comparison features

In order to quickly compare the projects in the related work to one other, we extracted very simple features that are relevant to the the construction of systems and architectures for HMS. These features are:

**sense:** Indicates whether the project supports background sensing

**survey:** Indicates whether the project supports human-reported information using surveys

**creator:** Indicates whether the project was created by **B**uilders or **D**eployers,

**architecture:** Indicates the level of detail at which the architecture is described. At the highest level, it only shows the relationship between **T**iers, but it can also show the details for the **C**lient, **S**erver, **A**nalysis tiers,

**OS:** Indicates the phone OSes supported; **a**ndroid-only or **i**OS-only or **B**oth

**open source:** Indicates whether the project is open source and the code is actually accessible

### 4.2 Observations

While looking at prior systems (Table 1), it is clear that deployers have constraints that builders are often able to ignore. One obvious example is the set of mobile OSes supported - deployers almost always support both android and iOS because they care about the coverage and representativeness. The one deployer project (SFTQS) that was android-only included an explicit argument that the bias in its data was small. But this constraint also restricts deployer effort to a fairly small set of use cases - most deployer effort is concentrated in travel diary creation. Builders have the luxury of experimenting with new and innovative use cases, but typically stop with a proof of concept on either android or iOS. Further, most systems, even by builders, are one-off projects

| Project | sense | survey | creator (B/D) | arch. (T/C/S/A) | OS (a/i/B) | open source | notes |
|---|---|---|---|---|---|---|---|
| **Classic travel diaries** | | | | | | | |
| FMS [Cottrill et al., 2013] | ✓ | ✓ | D | T | B | × | Data must be uploaded manually. Survey on website |
| SFTQS [Carrel et al., 2016] | ✓ | ✓ | D | × | a | × | Fairly complex app-based surveys for travel satisfaction. Requires surveys on 5 days of 6 week study. Based on ODK |
| DataMobile [Patterson and Fitzsimmons, 2016] | ✓ | × | D | T | B | ✓ | Only pre and post-study surveys are listed. System is open source, but only one application is described |
| **Crowdsourcing applications** | | | | | | | |
| Biketastic [reddy et al., 2010] | ✓ | * | B | T | a | × | sensed data used to derive traffic, and roughness. routes could be tagged with media. System was deployed but only for 12 users, so will categorize as created by builders |
| CycleTracks [Hood et al., 2011] | ✓ | * | D | × | B | ✓ | open source single mode travel. Manual start/stop of trip. open source, extended by other MTAs, e.g. Atlanta to record infrastructure issues. Unclear if this is done through surveys [Poznanski, 2013] |
| Tiramisu [Zimmerman et al., 2011] | ✓ | ✓ | B | T | i | × | single mode collection. manual start/stop of trip. provides a service (real-time bus and fullness information) to users. |
| **Behavior change** | | | | | | | |
| Matkahupi [Jylhä et al., 2013] | ✓ | × | B | × | a | × | allows users to set their own goals, and presents challenges based on travel patterns. |
| PEACOX [Bothos et al., 2014, Schrammel et al., ] | ✓ | × | B | A | a | × | clear choice architecture with multiple theory-based approaches for persuasive change. provides service (trip planner). |
| QT [Jariyasunant et al., 2015] | ✓ | * | D | T | B | × | reports travel along cost, $CO_2$, time. Correction of automatically sensed mode by logging in to a website. No other ongoing survey information. |

Table 1: Related applications, grouped along multiple axes. All the applications are published as standalone systems. Explanations: (i) * in a column implies that the answer is not clear, details are in the notes, (ii) column descriptions, including the abbreviations, are at Section 4.1.2

and are not open source. The CycleTracks app suggests that deployers do reuse open source projects if they meet a significant need.

Most prior platforms (Table 2) were developed by builders, as expected. However, few appear to address large-scale deployment concerns. In particular, except for ohmage, they only support either android or iOS - mostly android - which severely limits their use in deployer applications. Platforms tend to be open source much more often than applications, which is expected, since writing extensible software without making it open source requires significantly greater engineering design. However, the details are complicated - sometimes, part of the platform is not open source (ParticiPACT), or the code is not linked anywhere (Vita).

## 4.3 Most closely related

The most closely related platform is ohmage + lifestreams [Tangmunarunkit et al., 2015, Hsieh et al., 2013], and to a lesser extent, ParticiPACT + MSF [Cardone et al., 2014, Cardone et al., 2013]. While there are some key limitations as outlined below, it would be interesting to have both of them included in any future comparison of platforms (Section 3.3).

**No true multi-method** Although it supports both sensing and surveys, ohmage is primarily survey focused. Two of their studies (Mobilize, PREEMPT) are purely survey-based. It also does not appear to support combined passive sensing and self-reporting - the third study (moms) involved applicants using two separate apps, for self-reporting (survey) and mobility (sensing). In contrast, MSF, participACT's sensing architecture paper [Cardone et al., 2013] is focused on passive sensing, with a clear event architecture for combining various sensors, and for sensor based survey triggers. Unfortunately, it works only on android and does not address the limitations on background processing in iOS (**Location State Machine** in Section 5.1)

**android+iOS support?** While ohmage self-

| Project | sense | survey | creator (B/D) | arch. (T/C/ S/A) | OS (a/i/B) | open source | notes |
|---|---|---|---|---|---|---|---|
| | | | Survey-only **or** sensing-only platforms | | | | |
| Sensr [Kim et al., 2013] | × | ✓ | B | T | i | × | allows authoring of web-based survey tools; each survey response can be tagged with locations/photo/text, but no background sensing |
| ODK [Hartung et al., 2010, Brunette et al., 2013] | × | ✓ | B + D | T* | a | ✓ | survey responses can include sensed data; both single locations and tracks, manually triggered; SFTQS app extended it for background tracking. ODK 2.0 architecture is much more complete |
| DIMMER [Krylovskiy et al., 2015] | ✓ | × | B | S | * | * | platform proposes micro-services architecture for the server, unlike SOA of prior work; no details on “mobile applications”; funded by EU SMARTCITIES project, but no claim to open source |
| BOSS [Dawson-Haggerty et al., 2013] | ✓ | × | B | C + S | * | ✓ | services for smart building applications. only mobile component is web interface that supports personalized climate control. analysis (model training) is assumed to be part of the application |
| | | | Survey + Sensing platforms | | | | |
| AndWellness [Hicks et al., 2010] | ✓ | ✓ | B | C + S | a | * | similar to ODK but incorporates continuous location and activity sensing; shared server; architecture does not include analysis; extensibility is in future work; oriented toward ESM; visualization is standard, gamification would be hard; deployment options unclear |
| ohmage [Tangmunarunkit et al., 2015, Hsieh et al., 2013] | ✓ | ✓ | B | C + S + A | B | ✓ | from the same group at UCLA as AndWellness; follow-up project?; client architecture is scattered; most projects are survey-based; app was extended for PREEMPT, but unsure how much effort needed; analysis module was used for one project and hasn't been updated since 2014 |
| ParticipACT [Cardone et al., 2014, Cardone et al., 2013] | ✓ | ✓ | B | C + S | a | * | Client is open source; server is not. client architecture is extremely detailed but android-specific; deployment only reported on pre-installed phones; extension by other groups unclear |
| Vita [Hu et al., 2013b, Hu et al., 2013a] | ✓ | ✓ | B | C + S | a | * | Extremely detailed SOA for both client and server components for mobile crowdsourcing; architecture split across two papers; Smart City applications developed by research team; unsure if ever deployed; “open source mobile CPS”, code location unknown |

Table 2: Related platforms, grouped along multiple axes. Explanations: (i) * in a column implies that the answer is not clear, details are in the notes, (ii) column descriptions, including the abbreviations, are at Section 4.1.2

reporting apps are available for both android and iOS, it is not clear that the passive sensing ones are. Passive sensing frequencies are listed at 1 minute or 5 minutes ( [Tangmunarunkit et al., 2015], Section 3.2.2); iOS does not allow time-based configuration of sensor frequencies. MSF works only on android.

**Unclear analysis architecture** In participACT, the server and analysis architecture is unclear. How is the data analyzed? Is there a pipeline? How can others reproduce the analysis? How can they extend it? This obscurity extends to the actual source code. Although participACT is open source, their server code is only available “upon request,”. The ohmage analysis architecture is remarkably similar to ANONYMIE's (Section 7) which provides additional validation for our design choices. It is more feature rich in terms of change detection and prediction, but it is unclear whether the design supports reproducibility. In particular, it does not appear that any of the ohmage studies collected data on their own server instances or ran the analysis on their own data.

**App installation** For ohmage, in two of their three studies, participants were provided with phones with the app pre-installed. In the third (Mobilize, 2013), all participants were also developers, so it is unclear how representative the deployment process was. For participACT, all participants were provided with smartphones, presumably with the app pre-installed.

Finally, DataMobile [Patterson and Fitzsimmons, 2016] is a system, but the authors have solicited partners to use it for data collection. There is no user interaction beyond an initial survey, and it is not clear that the app is extensible enough to be modified in other ways. It also only connects to servers at the author's lab. However, it does represent a re-usable system, and it would be interesting to include it in the platform comparison (Section 3.3). It appears to be under active development, and it may still evolve into a full platform in the future.

# 5 Client architecture

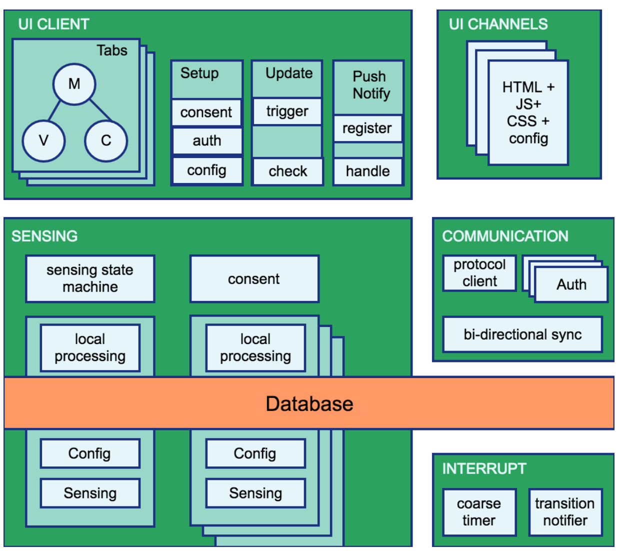


Figure 3: Client architecture, including modules for configurable sensing, robust communication and customizable UI

As we have seen from Section 4, most prior HMS projects have focused on the smartphone app and its ability to sense location and accelerometer data. However, most focus purely on the sensing and ignore the human interaction component. We develop a more complex architecture outlined in Figure 3 addressing this gap.

The core modules include configurable sensing, robust communication, and context-sensitive prompts. The novel component primarily involves the user interface and customization.

The three canonical use cases that we consider primarily modified the UI. They not only changed the user visible screens, but also configured the local end trip detection module notifier to display different prompts, and configured the communication to send data to their own server instance.

## 5.1 Sensing

The sensing module is conceptually simple - it reads and stores sensor values, automatically, in the background. However, power and latency considerations are important while choosing a particular point in the design space.

**Local buffering** The primary storage tradeoff relates to the frequency at which the data is uploaded to the server. While it may seem intuitive to use the server directly as storage by uploading the data as it is read, the radio draws significant power when turned on, so data should be buffered locally as much as possible. In the case of primarily passive data collection, such as for HMS, it is sufficient to upload data after a trip is complete.

Buffering also reduces data loss due to poor connectivity, and decreases the latency of computations on locally sensed data. However, it increases the latency of aggregate operations computed on the server, such as traffic speeds or counts for particular segments.

**Local processing** The primary processing tradeoff involves latency versus flexibility and complexity. Local processing on buffered data has the lowest latency but the least flexibility, since it has to be implemented in native code for each mobile OS (e.g. android, iOS, ...) that the platform supports. Local processing is also useful if the data volume of the sensor overwhelms the buffer.

ANONYMIE uses local processing for: (i) *low latency,* basic filtering of location points for use in the location state machine, and (ii) *large data volume,* accelerometer-based gesture detection for shakes or bumps.

**Location state machine** iOS supports a limited set of *background modes* [3], restricting the sensors (i.e. sound, location and bluetooth) that can be accessed in the background. The sensor must be relevant to

[3] `https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html`

the published app functionality (e.g. the VoIP background mode can only be used by VoIP apps, not mapping apps), the user must permit the app to access these sensors, and then explicitly permit them to be accessed in the background. This means that all other sensors (e.g. accelerometer) have to piggyback on one of the supported sensors for their operation. Further, the sensor APIs disallow periodic sampling, probably to prevent them from being used as periodic timers - e.g. the location sensor has a distance filter instead of a time filter.

These requirements imply that the primary tradeoff is between continuous sensing, which increases power consumption, and turning off tracking while at rest, which loses the first few minutes of a trip.

If we do turn off tracking while at rest, we need to have a state machine for location sensing that automatically transitions between `WAITING_FOR_TRIP_START` and `ONGOING_TRIP`.

**Consent** Most mobile OSes already require explicit consent for access to privacy-sensitive sensors such as location. However, additional regulations such as the European General Data Protection Regulations (GDPR) or academic Institutional Review Boards (IRB)s may require deployers to obtain more explicit consent that covers not just which data is collected, but also how it is processed and stored. All sensing should be stopped until explicit consent is received, and the consent should be documented for future reference.

## 5.2 Communication

The communication module deals with automatic upload of collected data and download of recently computed data for improved performance. This module needs to handle all aspects of communication, including establishing connections, authentication, and dealing with errors.

**Auth** All API calls to the server that transmit or receive personal data should be authenticated. The most basic form of authentication is to send a stored password, entered by the user, from the app to the server. While this is intuitive and easy to use, it should be combined with verification to avoid email hijacking.

For short studies with significant researcher interaction, an alternative is to pre-generate a list of random tokens and hand them out to participants. The researcher then does not need to know the users' email and can just use the unique token for all indexing.

For longer studies, the OAuth standard specifies the generation of encrypted tokens (JWT) with configurable expiry times. OAuth JWT tokens can be generated using open source auth servers such as Keystone, or by integrating with third party sign-in provides such as Google or Facebook.

**bi-directional sync** The main consideration for the bi-directional to/from data transfer is the **D**urability component of ACID transactions. Since any data transfer can be unreliable, the transfer should handle both poor connectivity and potential server errors without losing data.

One technique to accomplish this is to delete buffered data only after a `push` call fully succeeds. This may result in duplicate data from partial retransmissions but will not lose data. iOS allows apps to run in the background for no more than 30 seconds, so this code path should use parallel, async calls and rate limiting to speed up execution.

**protocol client** The HTTP REST protocol is a popular choice for client-server communication in prior HMS. However, pub/sub protocols such as MQTT, are popular for iOT systems. The resulting tradeoffs are closely related to those for 5.1. REST is better for batched intermittent connections, where connection setup and teardown do not cause significant overhead. MQTT works better for data that is continuously streamed to the server since the persistent connection reduces overhead. Again, for primarily passive data collection, REST is sufficient.

## 5.3 Interrupt handler

The interrupt handler deals with external triggers. Two current examples are:

**Coarse timer** We need to have a timer interrupt fire periodically to perform regular maintenance and recover gracefully from unexpected situations. For example, we may want to: (i) push any pending data that was retained in the buffer from previous partial retransmissions, or (ii) reset the location state machine if it is an inconsistent state.

This may appear to be trivial, since most standard OSes include a timer interrupt. However, in order to reduce power drain, many mobile OSes have limits on background operation for non-system services. If the limits are too strict (e.g. on iOS), we may need server intervention (e.g. *silent push notifications*) for reliable operation.

**Event notifier** The module also needs to deal with context-specific user notifications. While various events can be detected by the local processing module, deployers should be able to choose the message and actions in the displayed notification. Since the event is detected in the background, when the configurable UI is suspended, there needs to be a native code module to listen to the transitions and display the user-visible message.

### 5.4 User Interface (UI)

The primary tradeoff for the UI is performance versus effort. Native UIs have better performance, but more effort. This increased effort is intrinsic - native UIs will require a different implementation for each mobile OS that needs to be supported. Using a hybrid app approach, (e.g. PhoneGap, Apache Cordova), allows the core modules to be in native code while the UIs use standard web technologies (HTML+CSS+Javascript). This approach allows a single, consistent UI to be reused across multiple mobile OSes, while channel specific UIs can be dynamically downloaded on demand.

While the UI can be completely customized to meet the needs of the application, all of the three canonical use cases have used these three core components.

**Setup** An onboarding process introduces the app, acquires consent, and authenticates the user. This process can also include other initial steps, such as choosing a username or collecting demographic information.

**UI update** There needs to be a mechanism (triggered on app launch, or by the *coarse timer interrupt*, Section 5.3) that periodically checks for updates to the UI channel and applies them, potentially asking the user for confirmation.

**Notifications** The app needs to register for event notifications - both for context-specific user notifications and, due to OS restrictions (Section 5.3) for coarse timer interrupts.

### 5.5 User Interface (UI) channels

Each deployer who uses the platform should be able to configure it accordingly. Since the user interacts primarily with the UI, we expect that deployers might want to change the information displayed, the qualitative input solicited and the controls visible to the end user.

In order to provide maximum flexibility, platforms might want to support separate UI *channels* that the end-user can switch to. Each UI channel can have a completely different look and feel and can specify completely different configurations for the various modules.

Supporting dynamic UI *channels* also includes several other benefits:

**Randomized trials** It is easy to conduct randomized behavior trials by randomly directing end-users to different channels as they install the app.

**Custom server support** Since modules can be configured by the UI, installs using different channels can send their data to different servers. This allows deployers to have complete control over the collected data.

**Standardization** Particular deployer communities (e.g. travel survey groups) can develop canonical user interfaces for their particular use cases. This makes it easier to launch new examples of that use case, and also shortens the methods section of the resulting papers.

**Reproducibility** Such standardization would be difficult for behavioral studies, in which the goal is to innovate new methods of interaction. However, once the new interaction method has been embodied in a published channel, the study can be generalized or reproduced by recruiting new users and asking them to use the channel.

## 6 Server architecture

Although sensing (Section 5) is typically the focus of the related work, most deployers will also want to upload the data to a server for long-term storage, shared access, and complex analysis. The architecture of this server software is typically elided from platform descriptions. For example, a review of Experience Sampling Software( [Pejovic et al., 2015], Table 1) indicates that only ohmage includes a server component. The key modules for this tier are storage, data communication, and analysis. This section describes such an architecture in greater detail.

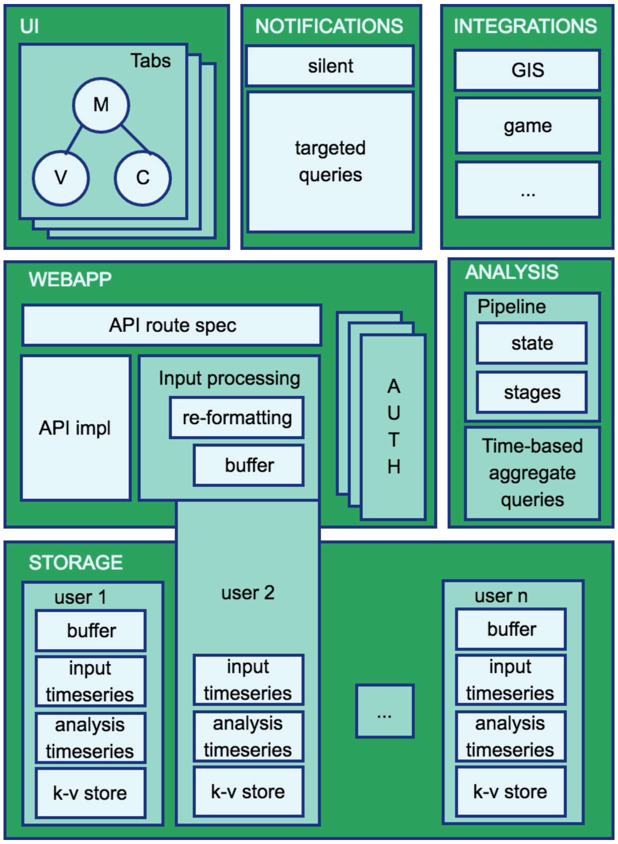


Figure 4: Server architecture, including modules for storage, communication and integration.

## 6.1 Storage

Storage is the key component of the server architecture. However, the actual storage instance or database product chosen depends on multiple factors like the number of users, the response time expected, and the resources available for the data collection. So we instead focus on the types of data collected and the broad storage category for each data type. These broad types are listed below.

**Input timeseries** The input data received from the smartphone app is stored in a timeseries database. Our data model and analysis pipeline (Section 7) treat this data as *read-only*.

The data can be conceptually viewed as separate user databases, each with multiple streams of data (e.g. `location`, `transition`). Most processing will work on one user at a time, multi-user queries will be aggregated across user databases. This formulation is compatible with privacy sensitive implementations.

Note that we actually have intermittent timeseries data because the sensors on the phone are not typically guaranteed to be periodic. And even if they were, if we use the state machine for lower power drain, there will be no data for long stretches of time. However, we still consider it to be a timeseries, since the primary querying method will be for a time range, and the primary index should be the timestamp associated with each data point.

**Analysis timeseries** Analysis results generated after processing the input data are stored in a separate timeseries database. While the volume of this data is not likely to be as high as the raw data, it is also time-indexed, and using the familiar timeseries interface allows us to stack analysis results (Section 7) in a consistent fashion.

**K-V store** Modifiable objects (e.g. `profile`, `config`) are conceptually modifiable objects associated with a particular user by a key. If the deployers would like versioning, and don't want to install two separate database packages, this data can also be stored in the timeseries database - the entry with the most recent timestamp is valid. However, these data will be looked up by key and not by time range, unless somebody requests an audit. So it does not need to be indexed on the timestamp.

**Incoming buffer** Since the background operation on iOS is time-bound (Section 5.2) we want the data received from the phone to be stored as quickly as possible. As server and database loads grow, directly storing incoming data into a potentially distributed timeseries database could introduce high latency. Instead, we can dump the incoming data into a separate, potentially local buffer, and move it into the timeseries before processing. This additional step also allows us to run pre-processing steps that unify the data model before inserting into the timeseries.

## 6.2 Other components

The analysis component of the server architecture is fairly complex, and is described in detail in Section 7. The other components of the server architecture are fairly straightforward and their novel features are described in brief in this section.

**Webapp** The webapp layer defines the API routes used by all clients, including the smartphone app, and any browser-based UIs. The webapp layer also authenticates all user-specific API calls, and needs to support the same set of authentication methods as the smartphone app (Section 5.2).

**Push Notifications** This module integrates with push notification services to send both *targeted surveys*, which send a link to a survey based on user mobility patterns, and *silent notifications*, which are used as coarse timer interrupts on the smartphone (Section 5.3).

**Integrations** This module handles external integrations. Current examples include `OpenStreetMap`, for GIS lookups, and Habitica, for gamification.

# 7 Analysis architecture

HMSes typically include aggregate analysis (e.g. modeling or dashboards) on the collected data. The raw sensor data needs to be cleaned and post-processed to provide the inferred data that is aggregated. These inference algorithms need to be transparent and reproducible so that they can be understood and improved by the research community. We meet these goals by defining a data model and algorithm structure for reproducible analysis. In the travel diary use, the data was collected on a server without any analysis. The analyst was then able to download the raw data and run the analysis pipeline on her laptop.

## 7.1 Pipeline

The analysis component is structured as a *progressive system* in which the only permanent state is the input data received from the smartphone app. The algorithm is structured as a pipeline with a set of *stages*, and the input to each stage is the output from the previous stage. Since each stage only modifies its output, the stages are idempotent. This implies that, given the same inputs and the same algorithm stages, the results will be the same.

This has several important implications.

**Reproducibility** Analysts can reproduce results from any version of the algorithm simply by running the code on a fresh set of inputs. If the code is versioned properly in a source control system (e.g. `github`), then reproducing results at a previous time $t$ is as simple as: (i) downloading the raw data, (ii) checking out the version $v$ of the source code at time $t$, (iii) running $v$ on the raw data. This allows analysts to reproduce prior results even as the code-base has evolved beyond the time that the data was collected.

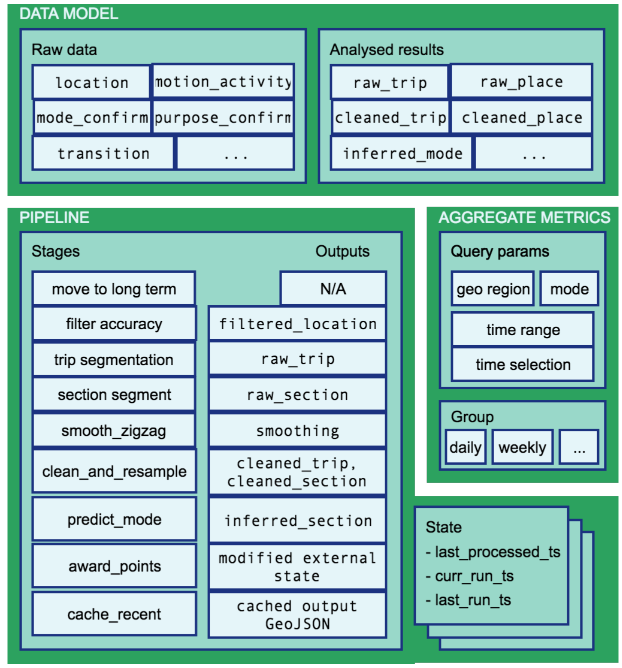


Figure 5: Analysis architecture, including modules for processing the data in idempotent stages, a data model that supports such an algorithm, and aggregate queries.

**Extensibility** If a researcher develops a new algorithm for a particular stage, she can run both the current state of the art and the new algorithm against the same input data and compare the results. If she chooses to publish the algorithm implementation, other researchers can reproduce her results by running the published algorithm against the raw data.

## 7.2 Data model

This algorithm design needs a data model that does not require modifying any fields. This can be accomplished in at least two possible ways:

**Carry forward** The data from the previous step is carried forward to the next step. For example, every trip can store the location points associated with it. Unfortunately, as the number of inferred objects increases, this can get increasingly unwieldy. For example:

1. since each multi-modal trip can be split into multiple *sections*, should every section also store the location points associated with it? This will cause duplication of location points between trips and sections.
2. How can we store ground truth for a trip as it ends, potentially before the pipeline has run and generated a trip object?

**Time association** The newly created objects for each step are associated with start and end time information. We can then associate raw or processed inputs with any object by querying for entries within that time range. This addresses most of the concerns with the *Carry forward* method. For example:

1. each section and each trip will have start and end timestamps. The set of points associated with a particular section or trip is then just the set of points between the start and end timestamps. This general query structure makes it easier for researchers to experiment with alternate algorithm implementations - each algorithm segments the raw data differently but does not duplicate it.
2. Since ground truth is a user input, it should not contain a reference to inferred data. Instead, the ground truth object also contains the time range that the user has *confirmed* (e.g with `mode` or `purpose`). The confirmed value for an inferred trip is represented by the the `confirm` object that overlaps with the inferred time range. This approach can be used for both trips and sections, depending on how much editing power the deployer wishes to provide to the end-user.

### 7.3 Aggregate metrics

The user provides a time range and a grouping, and gets back mode-specific aggregates. Some non-obvious details to note are:

**Geo queries** Analysts may want to restrict data retrieval to a particular region. This implies that all aggregate queries should support an (optional) geo-region, and the underlying timeseries should support geo-queries as well.

**Time selections** Most timeseries will support range queries where the range is specified in UTC. However, deployers may actually want to query by time slices instead - e.g. studying commute time travel patterns might involve accessing data from 2pm - 5pm for the month of April. Storing expanded times in the local time can support such disjoint time ranges.

## 8 Evaluation

In this section, we evaluate our architecture under the context of bridging the builder-deployer gap. Our evaluation is based on its use in three separate use cases (or "apps") from deployer projects. We show that although the use cases initially appear different, they re-use several common modules without modification, and are able to extend other modules to meet their needs. We also show that the development time for the projects is much shorter than building one-off apps from scratch. Finally, these projects show the ability to overcome the social challenges associated with inter-disciplinary platform building. However, in the absence of a rigorous user study, we have no knowledge of negative cases (e.g. deployers who are unconvinced by platforms). Further, although the platform enhancements have reduced as the platform matures, requests for documentation, particularly from non-developer deployers, are increasing with adoption. Therefore, the long-term viability of such platforms is still an open question.

### 8.1 Metrics

When presenting the idea of a platform to deployers, there was skepticism about the benefits of a platform. Some of the questions that have been raised, and the metrics that we use to answer them, are:

**Q:** Is there enough common functionality that it can be abstracted out?

**extensibility:** We examine the platform components identified by the architecture, and see how they are used by the systems instantiated from it. For example, does the architecture ensure that that common functionality is reused and all customization is restricted to customizable modules?

**Q:** What is the difference between an app and a platform? What is wrong with a one-off project? How much time will using a platform actually save?

**utility:** We compare the time required to create a one-off app from scratch with the time required to customize a platform.

**Q:** Will non-developer communities embrace open source platforms? Why not continue to use consultants instead?

**adoption:** We measure external contributions to both core modules and customizations, especially if the customizations were re-used by other projects.

**Q:** What about application-specific metrics such as survey responses or app launches?

**application specific metrics:** We do not include these because the platform does not control application-specific settings, and the metrics suitable for one application may not be suitable for others.

## 8.2 Use cases

Abstracting a set of specific systems into a platform involves striking a delicate balance between breadth and compactness. The platform should be *broadly applicable* to a wide variety of applications, or otherwise deployers will continue to build one-off systems. However, if the platform is too broad, it loses the clear structure that makes it useful. Striking that balance is more an art than a science, and the balance can shift over time. Most platform builders use a small ($n \approx 3$) set of canonical use cases to define the platform. The ANONYMIE platform uses three such canonical use cases - a classic travel diary, an infrastructure crowdsourcing project, and a behavior change study (Figure 6).

All of the projects provisioned their own server and collected their own data. All of them used a UI channel on top of the ANONYMIE base app. The use cases typically used 16 out of 25 features (64%), although the exact set varied according to the use case. In all three cases, the actual customization was done by undergraduates with computer science (CS) backgrounds; the undergraduates were from three different universities. The undergraduate who worked on the `cci-berkeley` project had worked with ANONYMIE the prior summer; the others had no prior direct experience.

**classic travel survey,** `cci-berkeley` The Center for Community Innovation (CCI) is instrumenting mobility patterns of low-income households in order to study the effects of gentrification on overall Vehicle Miles Travelled (VMT).

They use a classic travel survey with a stripped down UI that only includes the travel diary. They also removed several of the controls from the profile, notably, the option for "Medium accuracy", and the entire Developer Zone. Since they had graduate students recruit participants in person, they chose to hand out a unique, randomly generated token for authentication instead of having users sign in with an email ID.

They added the ability for users to specify mode and purpose ground truth from the diary screen, including a rich set of modes such as carpool, shared ride, etc.

They initially used the event notifier to pop-up a survey at the end of every trip, but turned it off after negative feedback during the pilot. They also added a survey that would link the user token to the user UUID, but ended up not using it when they switched to token-based authentication.

They did not run the analysis pipeline on the data collection server. Instead, the data analysts pull subsections of the data onto their own laptops and run the analysis on an ad-hoc basis.

**crowdsourced infrastructure,** `opentoall` The Taskar Center for Accessible Technology (TCAT) is documenting barriers to accessibility - bumpy or non-existent sidewalks, blocked routes, etc.

While they include the classic trip diary, they prompt the user at the end of every trip for their experience of the trip, including any barriers that they encountered that are not already in the `opentoall` dataset. They use OpenID connect, linked to their own keystone server for authentication. This allows them to associate trips taken by any user with trips recommended by the `opentoall` trip planner.

They are interested in gamification to prompt crowdsourcing of barriers, as well as adding local processing for bumpy sidewalk detection using the accelerometer.

**behavior change,** `tripaware` A group of undergraduates participating in a research apprentice program studied the difference between emotion (moody polar bear) and information (suggestions for alternate modes) in motivating sustainable behavior.

They conducted a Randomized Controlled Trial (RCT); participants were randomly assigned to the emotion, information or control channels, and automatically downloaded the appropriate UI for their group.

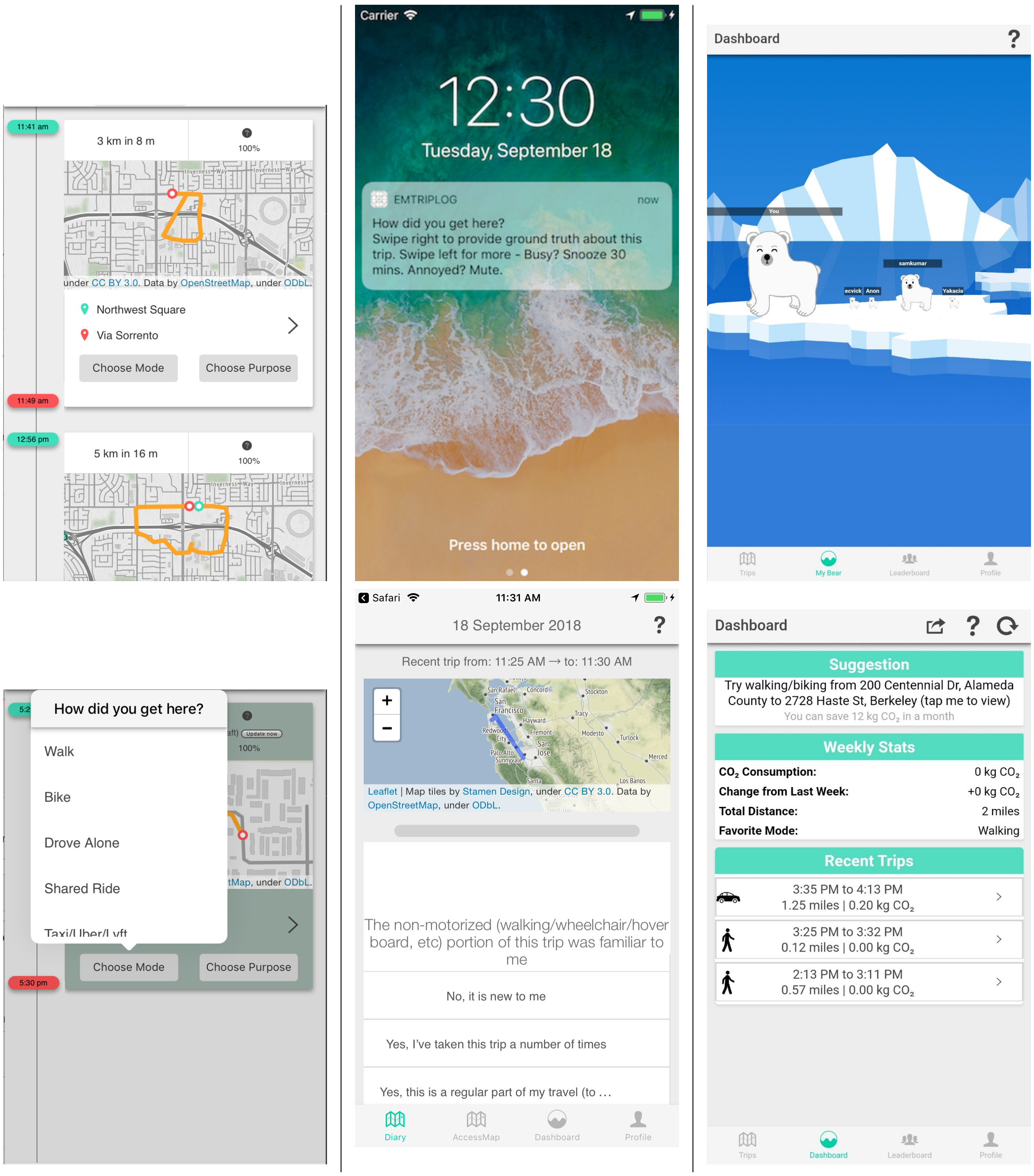


Figure 6: Screenshots of the three different use cases (L-R: `cci-berkeley`, `opentoall`, `tripaware`)

| | Feature | `cci-berkeley` | `opentoall` | `tripaware` |
|---|---|---|---|---|
| Client | Local buffering | ✓ | ✓ | ✓ |
| | Local processing | ✓ | ✓ | ✓ |
| | Location state machine | ✓ | ✓ | ✓ |
| | Consent | ✓ | ✓ | ✓ |
| | Auth | Pre-created token ↑ | OpenID connect ↑ | Google auth |
| | bi-directional sync | ✓ | ✓ | ✓ |
| | protocol client | ✓ | ✓ | ✓ |
| | Coarse timer | × | × | ✓ |
| | Event notifier | Removed after pilot | ✓ | × |
| | Setup | ✓ | ✓ | ✓ |
| | UI update | ✓ | ?? | ✓ |
| | Push notify | × | × | ✓ |
| | UI channel | ✓ | ✓ | ✓ |
| Server | Input timeseries | ✓ | ✓ | ✓ |
| | Analysis timeseries | Offline, on laptop | ✓ | ✓ |
| | K-V store | ✓ | ✓ | Added leaderboard tier position ↑ |
| | Incoming buffer | ✓ | ✓ | ✓ |
| | Webapp | ✓ | ✓ | New API endpoint for suggestions ↑ |
| | Push notify | × | × | ✓ |
| | Integrations | GIS for mode | GIS for mode, `opentoall` trip planner | GIS for mode |
| Analysis | Pipeline usage | Analyst runs offline ✓ | ✓ | New stage for *tiers, happiness* |
| | Reproducibility | Multiple analysts work with subsets of data ✓ | × | Investigate errors in mode inference ✓ |
| | Algorithm Extensions | × | × | × |
| | New data model objects | `mode_confirm` ↑ | survey result ↑ | × |
| | Aggregate metrics | × | × | × |

Table 3: Three projects and their usage of various components of the architecture. Usage key - ✓: used without modification, × not used, ↑ enhancement contributed by this project

They retained the classic trip diary for the control group. For other groups, they added a leaderboard, and modified the summary dashboard based on intervention. For the information group, they provided summary statistics and a set of suggestions for alternatives. For the emotion group, they showed a polar bear that grew or shrank, and was compared to the others in your leaderboard tier.

## 8.3 Extensibility + adoption

The community involvement metrics (Figure 7) indicate strong interest and a significant contributor base. Digging deeper, the usage matrix (Table 3) indicates that most of the components were used without modification in a majority of the projects. Most changes were to customizable modules. The only external enhancement to the core modules was the addition of a new auth by the `opentoall` project. Further, many of the contributions to customizable modules can reused by other projects. For example, the enhancement that allowed users to specify mode and purpose, introduced as part of the `cci-berkeley` project was adapted for use in the `opentoall` project.

Notable exceptions to these general results include:

**Auth** Every project used a different authentication mechanism. Having a configurable authentication mechanism allows deployers to easily switch between mechanisms, as well as allowing projects to contribute auth plugins that they needed for later re-use.

**Coarse timer/Push notify** 2 out of 3 projects did not turn on the silent push notification based coarse timer on iOS. Since the data can also be uploaded at trip end, the data collection still worked since both projects were based in the United States, which has reasonable connectivity. They also did not use targeted push notifications.

**Algorithm extensions** No group has yet contributed algorithm extensions. The CCI group is actively analysing their collected data and might contribute improvements if they develop any. Since the architecture and data model are now clearly documented, we hope that researchers who work on inference algorithms in the future will contribute them to the platform.

**Aggregate metrics** Since all the projects so far have been focused on small-scale data collection, they have not explored the aggregate analyses possible. The `opentoall` crowdsourced dataset could be an instance of such analysis once the study is complete.

## 8.4 Utility

The utility metric is difficult to assess because one-off deployer projects that did not publish source code do not publish their development time either. The commercial rMove app [Greene et al., 2016] took five months to develop, but the development team size is unknown. The one-off Quantified Traveler project [Jariyasunant et al., 2015] involved a development team of five in addition to the authors, but the details of contribution and time taken are unclear. DataMobile [Patterson and Fitzsimmons, 2016] is open source, but it only recently (June 2018) created github repositories through code bulk upload, so we are unable to see the commit history.

As shown below, all of the ANONYMIE changes so far have taken $< 3$ months with CS undergraduates working part-time. Less ambitious changes are possible with one undergraduate, RCTs with multiple UIs need a larger team.

**cci-berkeley** $\approx$ 6 weeks of full-time work by one CS undergraduate with prior ANONYMIE experience + $\approx$ 2 weeks of part-time work by another CS undergraduate to change text and colors.

**opentoall** $\approx$ 1 month of full-time work by one CS undergraduate for extending auth + $\approx$ 3 months of extremely part-time effort for UI changes + integration.

**tripaware** $\approx$ 3 months of 6-10 hrs/wk by 6 CS undergraduates to design 3 custom UIs for RCT + server changes for leaderboard and polar bear

An advanced UI is planned to be developed for Sydney area to be completed in one month time by a professional programmer and a research student, to collect the travel diary of Sydney residents for two weeks. Further, the platform is being used for collecting information about the route choice behavior of pedestrians in dense urban area of Sydney CBD.

Note that this estimate only accounts for deployer, not builder effort. While the pace of platform enhancements has slowed as it has matured, requests for clarification and documentation are increasing. These requests are particularly numerous when non-CS deployers are involved - for example, although `cci-berkeley` UI changes took only $\approx$ 2 months, it took another month and a half for the CCI group

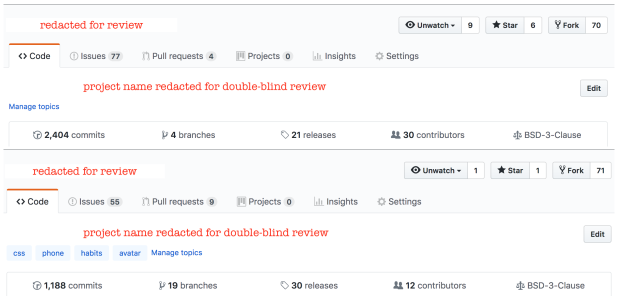


Figure 7: Basic community involvement statistics from github for the server (top) and phone (bottom) repositories

to perform routine non-platform-specific system administrative tasks. Integrating with ongoing *software carpentry*[4] efforts may result in documentation that meets the needs of non-CS audiences without overwhelming platform builders, especially in resource-constrained research environments.

### 8.5 Application specific metrics

The evaluation does not include application specific metrics. This is mainly because:

**no control:** The metrics are influenced by a mix of factors, few of which the platform controls. For example, app retention is likely to be influenced by monetary incentives, app functionality and power drain. The platform does not control either of the first two options, and the third is configurable by the app. So it is possible for applications with similar functionality, built on the same platform, to have drastically different retention rates. Thus, retention rate is not an appropriate metric for evaluating the platform.

**application dependent concerns:** Other metrics might be heavily application-dependent. For example, metrics for HMSes could include the number of questions for each trip, and the response rate. These metrics are not meaningful for gamification, where users are typically not surveyed about trips, and might even be prevented from providing additional information in order to avoid cheating. Likely gamification metrics are number of app opens and the length of time on each screen.

Of course, applications could establish their own metrics, compare implementations and establish best practices(Section 3.4)

## 9 Conclusion

Human Mobility Systems (HMS) can form the basis for applications in domains ranging from travel behavior studies to crowdsourcing initiatives to identify structural barriers to transportation. We generalize an open-source platform, ANONYMIE[5] from use cases in these domains. *Deployers* can use this platform to instantiate customized systems for their own domains. In order to bridge the gap between *deployers* and *builders* of such systems, we also outline a clear platform architecture that describes the client, server and architecture tiers, and the components in each tier.

Our evaluation shows that the architecture is: (i) *extensible*, since all customization was to non-core modules; all module extensions could be performed without rewriting other modules, and (ii) *useful*, since the time taken to create a custom "app" for a new project was $<3$ months of part-time undergraduate time.

However, the evaluation also reveals several open questions. (i) All the extensions thus far have been to the UI, and none of the projects has contributed incremental improvements to the core algorithms. (ii) In the absence of a comprehensive user study, it is unclear whether the deployer community will embrace extensible platforms, and contribute meaningfully to them. (iii) All deployments thus far have required substantial builder assistance to create documentation and clarify concepts. It is unclear how best to balance builder and non-CS deployer effort to improve usability, especially in resource-constrained research environments.

We anticipate gaining more clarity around these questions as the ANONYMIE platform usage expands, and other similar platforms for human sensing (e.g. [Patterson and Fitzsimmons, 2016, Tangmunarunkit et al., 2015]) are developed and used.

## References

[dig, 2018] (2018). Digitransit. `https://digitransit.fi/en/`. Accessed: 2018-10-25.

[4] `https://software-carpentry.org/`

[5] name changed for double blind review

[goo, 2018] (2018). Google timeline. https://support.google.com/accounts/answer/3118687. Accessed: 2018-11-11.

[GPS, 2018] (2018). Gps logger for android. https://play.google.com/store/apps/details?id=com.mendhak.gpslogger. Accessed: 2018-10-24.

[hab, 2018] (2018). Habitica. https://habitica.com/. Accessed: 2018-10-25.

[myT, 2018] (2018). mytracks for android and ios. https://play.google.com/store/apps/details?id=com.zihua.android.mytracks, https://itunes.apple.com/us/app/mytracks-the-gps-logger/id358697908. Accessed: 2018-10-24.

[otp, 2018] (2018). Open trip planner. http://www.opentripplanner.org/. Accessed: 2018-10-25.

[str, 2018] (2018). Strava. https://strava.com. Accessed: 2018-10-25.

[Bothos et al., 2014] Bothos, E., Prost, S., Schrammel, J., Röderer, K., and Mentzas, G. (2014). Watch your Emissions: Persuasive Strategies and Choice Architecture for Sustainable Decisions in Urban Mobility. *PsychNology Journal*, 12(3):107–126.

[Brunette et al., 2013] Brunette, W., Sundt, M., Dell, N., Chaudhri, R., Breit, N., and Borriello, G. (2013). Open data kit 2.0: expanding and refining information services for developing regions. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 10. ACM.

[Cardone et al., 2014] Cardone, G., Cirri, A., Corradi, A., and Foschini, L. (2014). The participact mobile crowd sensing living lab: The testbed for smart cities. *IEEE Communications Magazine*, 52(10):78–85.

[Cardone et al., 2013] Cardone, G., Cirri, A., Corradi, A., Foschini, L., and Maio, D. (2013). Msf: An efficient mobile phone sensing framework. *International Journal of Distributed Sensor Networks*, 9(3):538937.

[Carrel et al., 2016] Carrel, A., Sengupta, R., and Walker, J. L. (2016). The San Francisco Travel Quality Study: tracking trials and tribulations of a transit taker. *Transportation*, pages 1–37.

[Cottrill et al., 2013] Cottrill, C. D., Pereira, F. C., Zhao, F., Dias, I. F., Lim, H. B., Ben-Akiva, M., and Zegras, P. C. (2013). 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.

[Dawson-Haggerty et al., 2013] Dawson-Haggerty, S., Krioukov, A., Taneja, J., Karandikar, S., Fierro, G., Kitaev, N., and Culler, D. (2013). Boss: Building operating system services. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 443–458, Berkeley, CA, USA. USENIX Association.

[Dobkin, 2013] Dobkin, B. H. (2013). Wearable motion sensors to continuously measure real-world physical activities. *Current Opinion in Neurology*, 26(6):602–608.

[Doherty and Oh, 2012] Doherty, S. T. and Oh, P. (2012). A multi-sensor monitoring system of human physiology and daily activities. *Telemedicine and e-Health*, 18(3):185–192. PMID: 22480300.

[Flake et al., 2017] Flake, L., Lee, M., Hathaway, K., and Greene, E. (2017). Use of smartphone panels for viable and cost-effective gps data collection for small and medium planning agencies. *Transportation Research Record: Journal of the Transportation Research Board*, 2643:160–165.

[Greene et al., 2016] Greene, E., Flake, L., Hathaway, K., and Geilich, M. (2016). A Seven-day smartphone-based GPS household travel survey in Indiana. Washington, D.C. Transportation Research Board.

[Hartung et al., 2010] Hartung, C., Lerer, A., Anokwa, Y., Tseng, C., Brunette, W., and Borriello, G. (2010). Open data kit: tools to build information services for developing regions. In *Proceedings of the 4th ACM/IEEE International Conference on Information and Communication Technologies and Development - ICTD '10*, pages 1–12, London, United Kingdom. ACM Press.

[Hicks et al., 2010] Hicks, J., Ramanathan, N., Kim, D., Monibi, M., Selsky, J., Hansen, M., and Estrin, D. (2010). AndWellness: an open mobile system for activity and experience sampling. In *Wireless Health 2010 on - WH '10*, page 34, San Diego, California. ACM Press.

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

[Hsieh et al., 2013] Hsieh, C.-K., Tangmunarunkit, H., Alquaddoomi, F., Jenkins, J., Kang, J., Ketcham, C., Longstaff, B., Selsky, J., Dawson, B., Swendeman, D., Estrin, D., and Ramanathan, N. (2013). Lifestreams: A modular sense-making toolset for identifying important patterns from everyday life. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 5:1–5:13, New York, NY, USA. ACM.

[Hu et al., 2013a] Hu, X., Chu, T. H. S., Chan, H. C. B., and Leung, V. C. M. (2013a). Vita: A crowdsensing-oriented mobile cyber-physical system. *IEEE Transactions on Emerging Topics in Computing*, 1(1):148–165.

[Hu et al., 2013b] Hu, X., Leung, V. C. M., Du, W., Seet, B., and Nasiopoulos, P. (2013b). A service-oriented mobile social networking platform for disaster situations. In *2013 46th Hawaii International Conference on System Sciences*, pages 136–145.

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

[Jean Wolf et al., 2014] Jean Wolf, William Bachman, Marcelo Olivera, Joshua Auld, Abolfazl (Kouros) Mohammadian, Peter Vovsha, and Johanna Zmud (2014). *Applying GPS data to understand travel behavior, Volume II: Guidelines*, volume 2 of *NCHRP national cooperative highway research program report*. Transportation Research Board of the National Academies, Washington, DC.

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

[Kim et al., 2013] Kim, S., Mankoff, J., and Paulos, E. (2013). Sensr: evaluating a flexible framework for authoring mobile data-collection tools for citizen science. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 1453–1462. ACM.

[Krylovskiy et al., 2015] Krylovskiy, A., Jahn, M., and Patti, E. (2015). Designing a Smart City Internet of Things Platform with Microservice Architecture. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pages 25–30, Rome, Italy. IEEE.

[Morgul et al., 2013] Morgul, E. F., Ozbay, K., Iyer, S., and Holguin-Veras, J. (2013). Commercial vehicle travel time estimation in urban networks using gps data from multiple sources. Washington, D.C. Transportation Research Board.

[Patterson and Fitzsimmons, 2016] Patterson, Z. and Fitzsimmons, K. (2016). Datamobile. *Transportation Research Record: Journal of the Transportation Research Board*, 2594:35–43.

[Pejovic et al., 2015] Pejovic, V., Lathia, N., Mascolo, C., and Musolesi, M. (2015). Mobile-Based Experience Sampling for Behaviour Research. *arXiv preprint arXiv:1508.03725*.

[Poznanski, 2013] Poznanski, A. (2013). Analysing Demographic and Geographic Characteristics of "Cycle Atlanta" Smartphone Application Users. Master's thesis, Georgia Institute of Technology.

[Rashidi et al., 2017] Rashidi, T. H., Abbasi, A., Maghrebi, M., Hasan, S., and Waller, T. S. (2017). Exploring the capacity of social media data for modelling travel behaviour: Opportunities and challenges. *Transportation Research Part C: Emerging Technologies*, 75:197 – 211.

[reddy et al., 2010] reddy, s., shilton, k., denisov, g., cenizal, c., estrin, d., and srivastava, m. (2010). biketastic: sensing and mapping for better biking. In *proceedings of the sigchi conference on human factors in computing systems*, pages 1817–1820. acm.

[Schrammel et al., ] Schrammel, J., Busch, M., and Tscheligi, M. Peacox – Persuasive Advisor for CO2-Reducing Cross-modal Trip Planning. page 4.

[Tangmunarunkit et al., 2015] Tangmunarunkit, H., Kang, J., Khalapyan, Z., Ooms, J., Ramanathan, N., Estrin, D., Hsieh, C. K., Longstaff, B., Nolen, S., Jenkins, J., Ketcham, C., Selsky, J., Alquaddoomi, F., and George, D. (2015). Ohmage: A

General and Extensible End-to-End Participatory Sensing Platform. *ACM Transactions on Intelligent Systems and Technology*, 6(3):1–21.

[Wang et al., 2014] Wang, X. C., Zhou, Y., Goevaers, R., Holguin-Veras, J., Wojtowicz, J., Campbell, S., Miguel, J., and Webber, R. (2014). Feasibility of installing noise reduction technologies on commercial vehicles to support off-hour deliveries. Technical Report C-11-13, New York State Department of Transportation.

[Zimmerman et al., 2011] Zimmerman, J., Tomasic, A., Garrod, C., Yoo, D., Hiruncharoenvate, C., Aziz, R., Thiruvengadam, N. R., Huang, Y., and Steinfeld, A. (2011). Field trial of Tiramisu: crowd-sourcing bus arrival times to spur co-design. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, page 1677, Vancouver, BC, Canada. ACM Press.