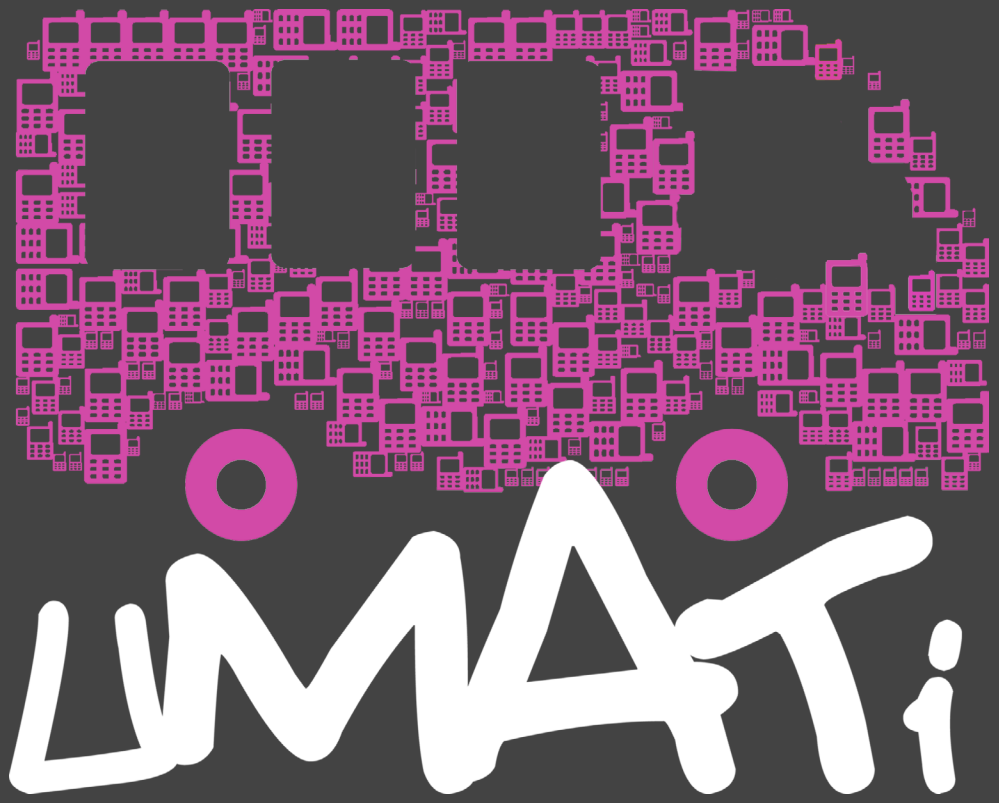


DIGITAL MATATUS CROWDSOURCING APP GUIDEBOOK

May 18, 2015

Carolina Morgan, Fei Xu
Marcel Williams, Rida Qadri

SCHOOL OF COMPUTING AND INFORMATICS
UNIVERSITY OF NAIROBI

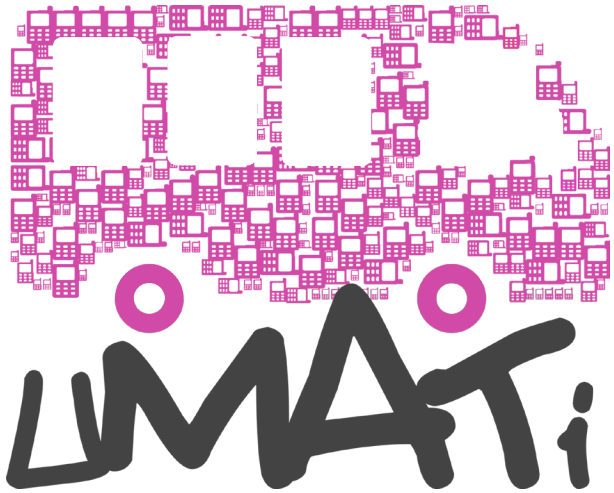
MIT CIVIC DATA DESIGN LAB

SUMMARY

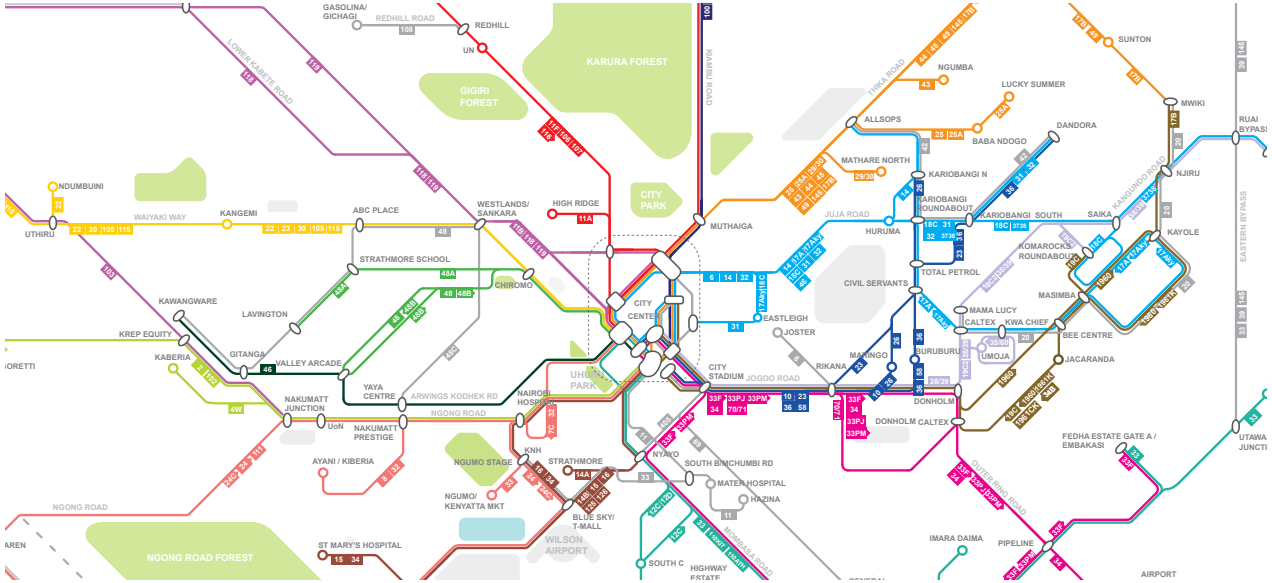
Nairobi's Matatus system is a complex one that is always changing to meet the transit needs of the city. The matatus route map needs to be equally dynamic.

Our task was to create an app that would help the Digital Matatus team streamline some of its data collection and analysis process to be able to keep the map current and accurate.

We conceptualized an app that will crowdsource route data by enticing users to track their matatus rides. Our prototype only begins to illustrate the possibilities. There is great potential to expand the app into an integral part of the matatus experience.
Introducing:



BACKGROUND



MIT's Civic Data Design Lab and the University of Nairobi's Change for Development Lab formed a partnership to develop the first ever map of Nairobi's informal public transit system: the matatus. Once published, the map received wide recognition and a grateful welcome by the city.

However, the process of creating the app took months of hard work by university students and professors. Armed first with GPS tracking units and eventually GPS-enabled phones, students rode all routes in the city. In order to ensure full coverage, they sought out route owners, bus drivers and riders and painstakingly catalogued all the routes for the first time. They then developed a static GTFS database including bus routes, stops and location data. They sifted through this data and eventually arrived at the final route paths, with their many splits and joints. The result was the stylized map of matatus routes we have today.

STRATEGY

users:
MATATUS RIDERS

device:
SMART PHONES

platform:
APP INVENTOR + FUSION TABLES

THE CHALLENGE

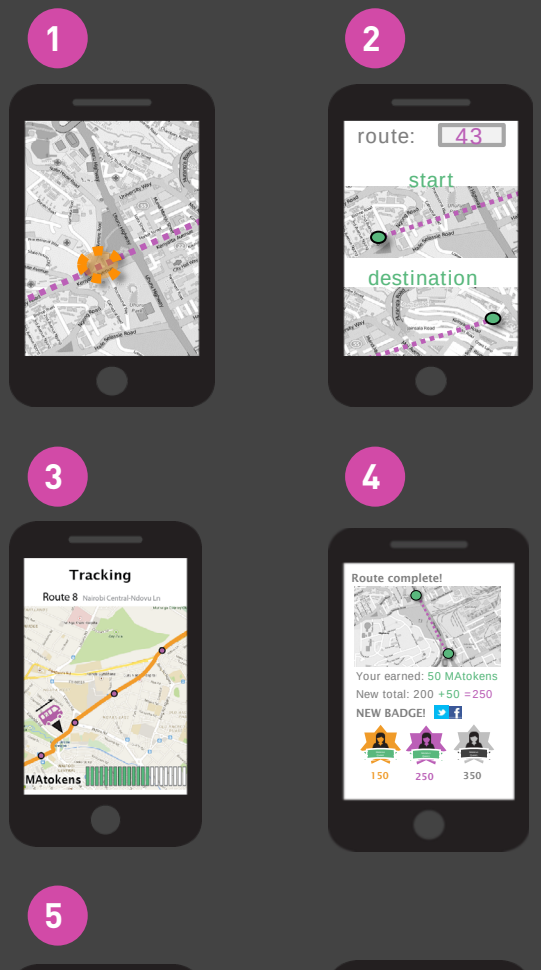
As routes changed, bus stops moved, and new lines were added, the map quickly started to become outdated. Yet repeating the labor-intensive process would be too time and funds-consuming. Our challenge was to design an app that would shorten the map's production time by crowdsourcing and automatic at least parts of the process.

Our strategy was to capitalize on Nairobi's existing assets: the large number of people who take matatus every day, and the widespread use of smartphones. By developing an app that enticed everyday riders to track themselves while they ride, we would be opening up a vast source of information about how the system works.

There are existing apps that have similar functions to what we needed. Transitwand and flocktracker are very useful apps for collecting GPS points, but they are designed for researchers, not for the general public. Ma3route is an app that has emerged in Nairobi that uses the Digital Matatus data, and it has garnered a large following for its traffic updates, but its directions are not map-based and the reports mostly focus on crashes and accidents without GPS location data. We wanted to create an app that was user-centered but still performed the required functions.

On the one hand, this presented certain technical challenges: the app needed to collect continuous GPS data rather than sporadic points, and it needed to be able to determine whether the user is riding matatus or walking and whether the route change is temporary or permanent. On the other hand, implementation challenges included minimizing data and battery usage, incentivizing app download and usage, providing useful information to the user, and ensuring full coverage of the city despite geographic socioeconomic or demographic disparities. We addressed these challenges in an app we called uMATi.

THE CONCEPT



"Umati" means crowd in Swahili. We chose this name because it reflects the main aspect of the app: its power lies in its widespread use. So developing an incentive for riders to use the app was key. The concept was as follows:

1. A user intending to ride a matatus turns on the app to navitage an interactive map with an overlay of matatus routes in order to find which route to take and where is her stop.

2. Before she gets on the bus, she enters her trip parameters. She selects her the matatus route she is taking, her starting point (her current location) and her destination (this is important in case she later forgets to turn off her tracking).

3. Once she gets on her bus, she starts the tracking. As she rides, the app collects a series of GPS points which get labeled according to the route she identified. She can see where she is on the map and the distance she has travelled.

4.When she arrives at her destination and ends her tracking, her meters travelled get converted into points we call "MATokens."

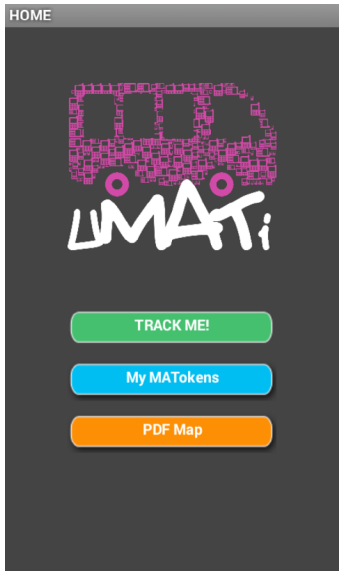
5.These points unlock coupons which she can trade for goods and services at local stores. She also earns badges as she rides more and more which she can post on her social media pages.

The points collected get added to a database. Each rider's trip becomes a fragment that is part of the larger system of routes.The data can then be aggregated either manually or automatically into an updated map. An algorithm could be written to determine a threshold for when enough users differ from an existing route to re-draw the map.

PROTOTYPE

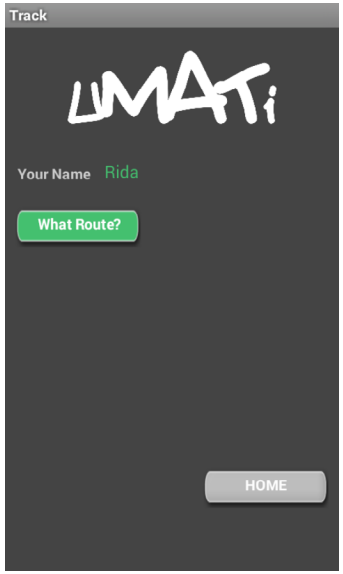
The app described above is a major undertaking, with many technical and practical aspects that need careful development beyond our capabilities. However, in order to demonstrate the concept, we developed a simple GPS tracking app that could serve as the foundation for full build-out of the app. Our prototype has a sleek design, basic tracking functionalities, and simple data collection. Other important components such as the data synthesis algorithm, point snapping to roads, and the points system are not included.

THE APP: HOW IT WORKS



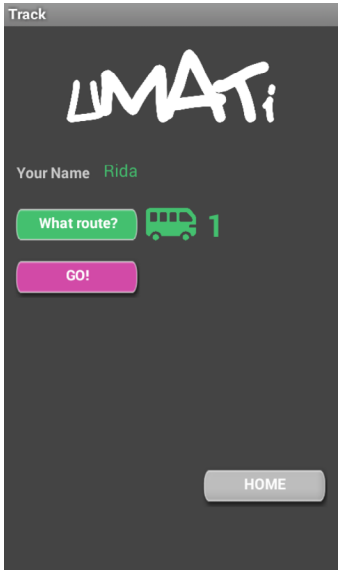
1

Press Track Me on the Home screen to go to Tracking options



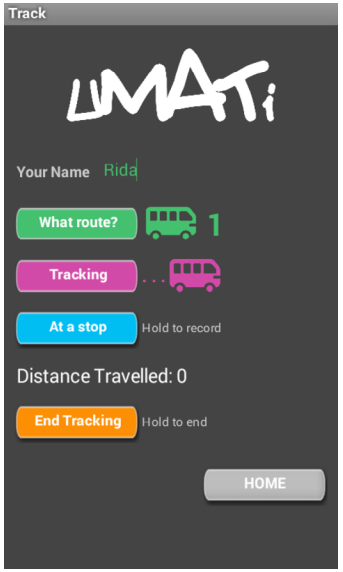
2

Give yourself a Username (this is in lieu of having a unique Device User ID)



3

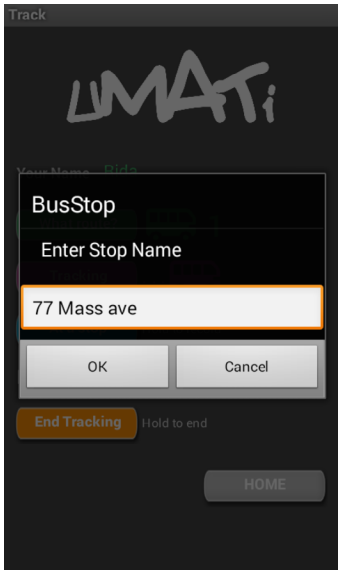
Select Route from Route options. This will identify each trip with the route you are on. Press Go to begin tracking.



4

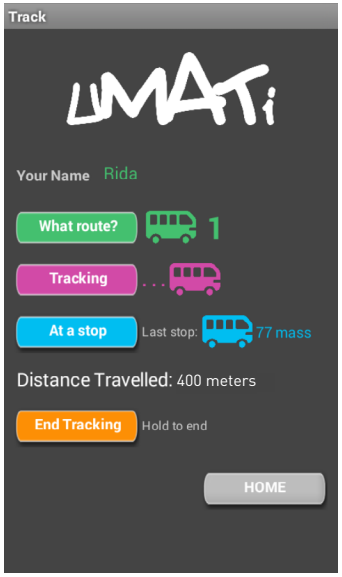
You can record Bus Stops by holding down the Bus Stop button, where you can identify the Bus Stop

THE APP: HOW IT WORKS



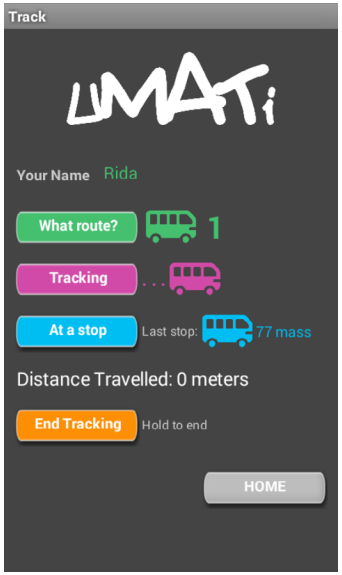
5

You can record Bus Stops by holding down the Bus Stop button, where you can identify the Bus Stop



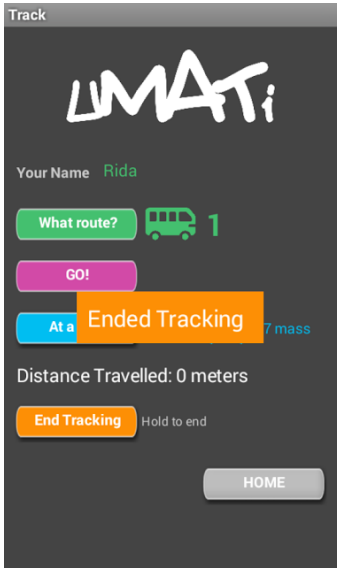
7

As your bus moves, the distance travelled gets updated in meters and displayed at the bottom of the screen.



6

The last bus stop you're at is recorded on the screen



8

When you want to End Tracking you hold that button till a message pops up.

APP INVENTOR

This app is programmed through App Inventor 2. The AIA file with the block code described is provided along with this guide.

For more information , please visit: http://appinventor.mit.edu/explore/library.html

App Inventor Resources:
http://appinventor.mit.edu/explore/ai2/tutorials.html
http://puravidaapps.com/

BACK END: FUSION TABLES

To store the data, we set up a fusion table. Fusion tables require users to have 'special permission' to write into them, which is why a service account email address was set up for the app. This ensured that our end user would have writing privileges to the table through the app, without making the fusion table publicly accessible. You have to download the KeyFile (.p12 key) and Service account email set up through Fusion Table's Developers console (https://console.developers.google.com/)

The Fusion Table:

- Date: A timestamp of every record
- UserID: An Identifier for each user. Taken from the Username for now, but should be eventually replaced with a unique device id
- TripID: An identifier for each unique "Trip" (Collection of points from one session) a User takes
- Route: The number of the Bus Route the user is on, selected from the pre-programmed list of option on the screen
- Latitude and Longitude: Location recorded every 10 seconds
- Stop: A dummy variable to signify if the current record is a BusStop. Coded as a 1 if it is a BusStop, blank if it is not.
- BusStopID: The name of the Bus Stop added by the User, after pressing the At A Stop button.

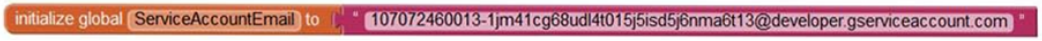
Date	UserID	TripID	Route	Latitude	Longitude	Stop	BusStopID
May 5, 2015 12:58:25 PM	Rida	1430845093081	1	42.35959	-71.09362	1	77 mass
May 5, 2015 12:58:23 PM	Rida	1430845093081	1	42.35959	-71.09362		
May 5, 2015 12:58:22 PM	Rida	1430844910039	1	42.35953	-71.09383		
May 5, 2015 12:58:16 PM	Rida	1430844984477	1	42.35948	-71.09452		
May 5, 2015 12:58:12 PM	Rida	1430844910039	1	42.35953	-71.09383		
May 5, 2015 12:58:06 PM	Rida	1430844984477	1	42.35948	-71.09452		
May 5, 2015 12:58:02 PM	Rida	1430844910039	1	42.35953	-71.09383		
May 5, 2015 12:57:56 PM	Rida	1430844984477	1	42.35948	-71.09452		
May 5, 2015 12:57:52 PM	Rida	1430844910039	1	42.35953	-71.09383		
May 5, 2015 12:57:45 PM	Rida	1430844984477	1	42.35948	-71.09452		
May 5, 2015 12:57:42 PM	Rida	1430844910039	1	42.35953	-71.09383		
May 5, 2015 12:57:35 PM	Rida	1430844984477	1	42.35948	-71.09452		
May 5, 2015 12:57:32 PM	Rida	1430844910039	1	42.35953	-71.09383		

BACK END: THE CODE

FUSION TABLES

We set up the fusion tables component of the app in AppInventor. To initialize acces, we had to create global variables for:

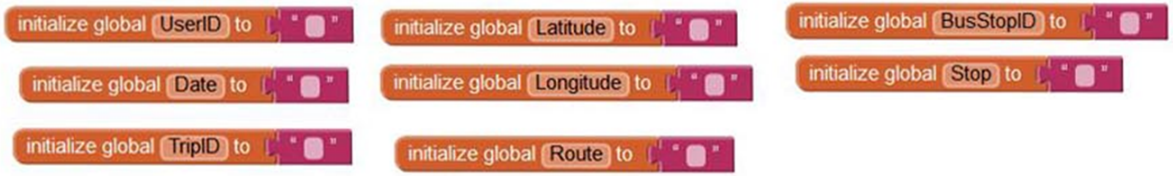
Service Account Email, with the email ID associated with the Service Account.



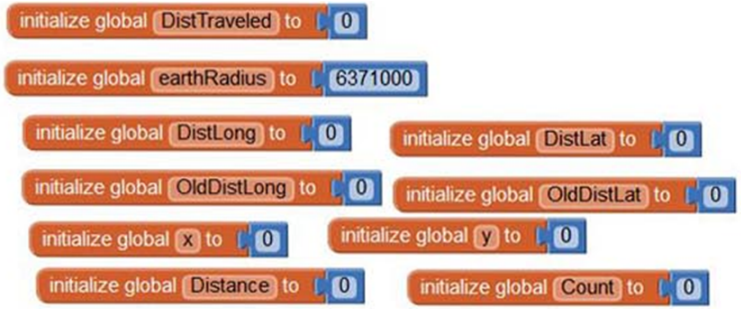
TABLE_URL which points the app to the URL of the Fusion Table

Table ID of the Fusion table, and the name of the KeyFile which allows access

We Then initialize the columns:



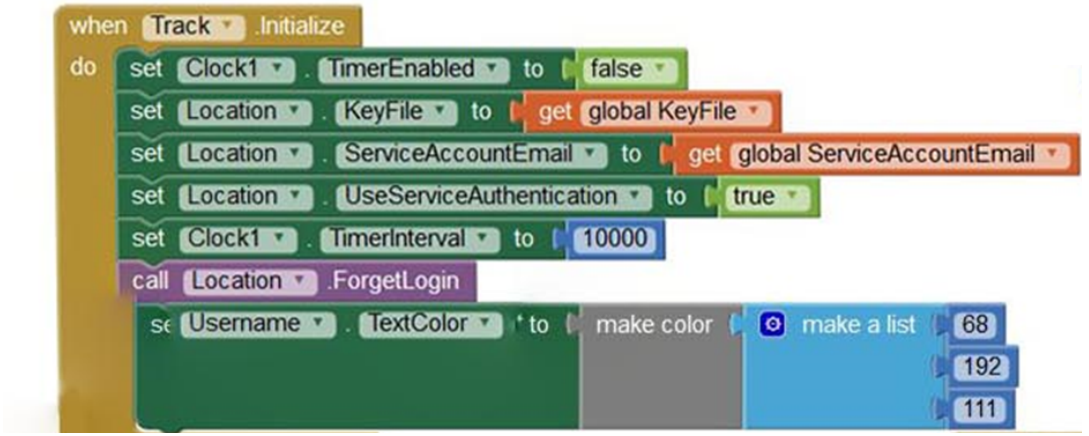
We set up the processes for calculating Distance Travel



THE CODE: FUNCTIONALITIES

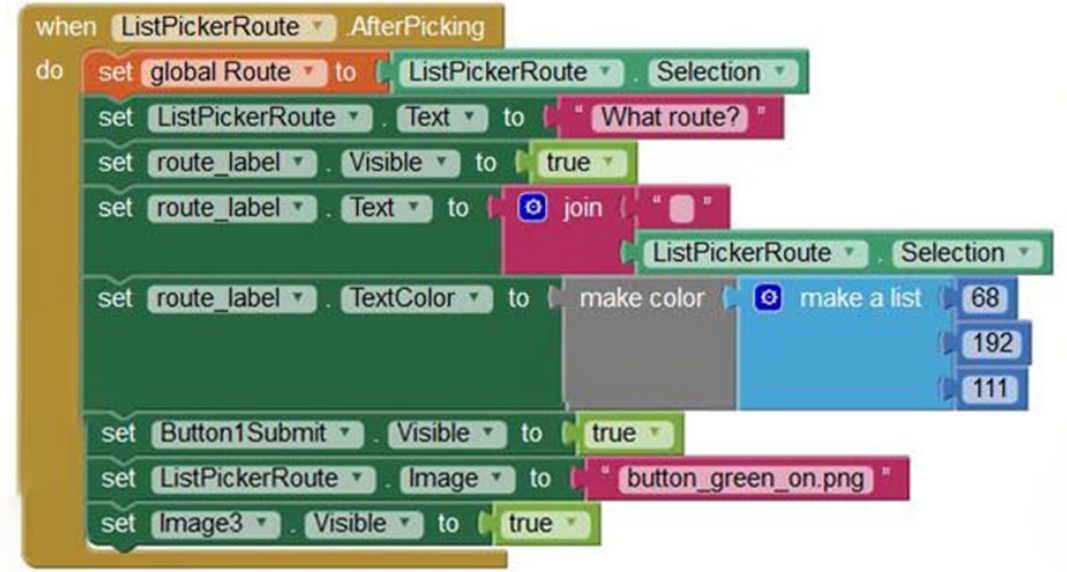
TRACKING

When Track.Initialise tells the app how to behave when the Track screen is pulled up. In this case the Keyfile and service account email to access the fusion table (called Location) is set. The App is programmed to insert route data in the table every 10seconds which means the clock timer interval is set to go off every 10seconds (in milliseconds) once it is turned on. To begin with we set the timer to False so that the timer only begins counting when we turn it on(set it to true). The login details of the user stored in the device are forgotten so that the User can access the fusion table through the service account email. A Textcolour is set matching our uMATi theme.



ROUTE PICKER

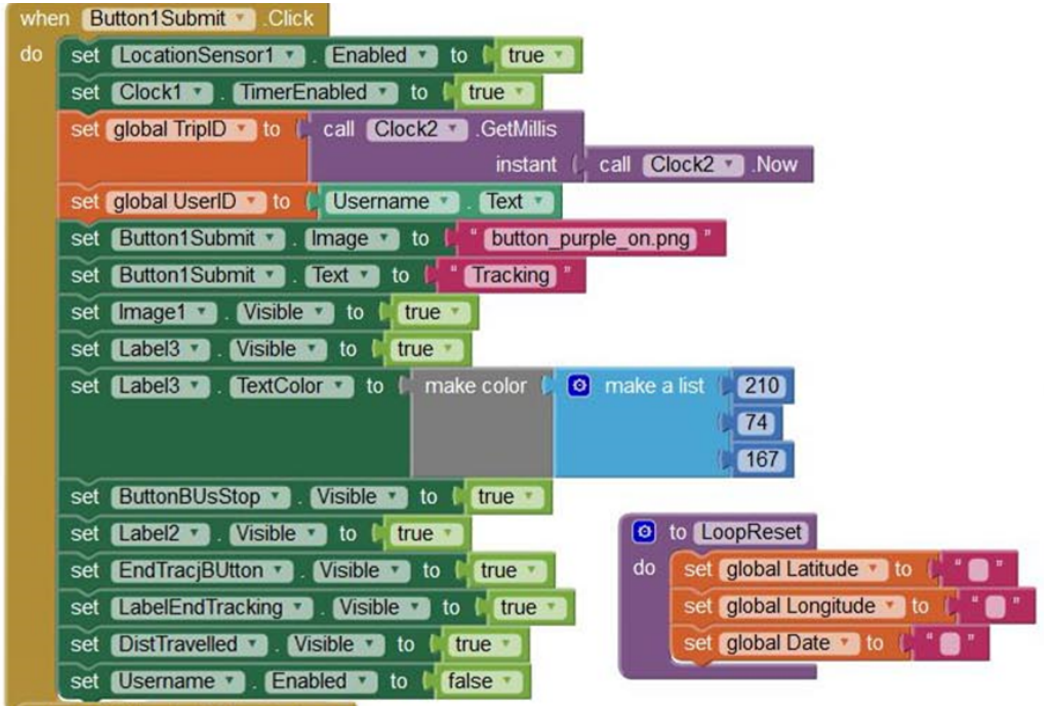
When listpickerroute.afterpicking sets up the process for how the app behaves once the route is picked from the dropdown menu. Once the Route is selected, the global route variable is set to whatever Route number was picked from the List by the user. The App then displays your route information on the screen and makes the submit button visible.



THE CODE: FUNCTIONALITIES

SUBMIT

When Button1submit.click handles the post submission process. The location sensor is turned on, the clock timers is turned on, the UserID and TripID are set for that session and other features of the app like End Tracking and BusStop button are made visible.



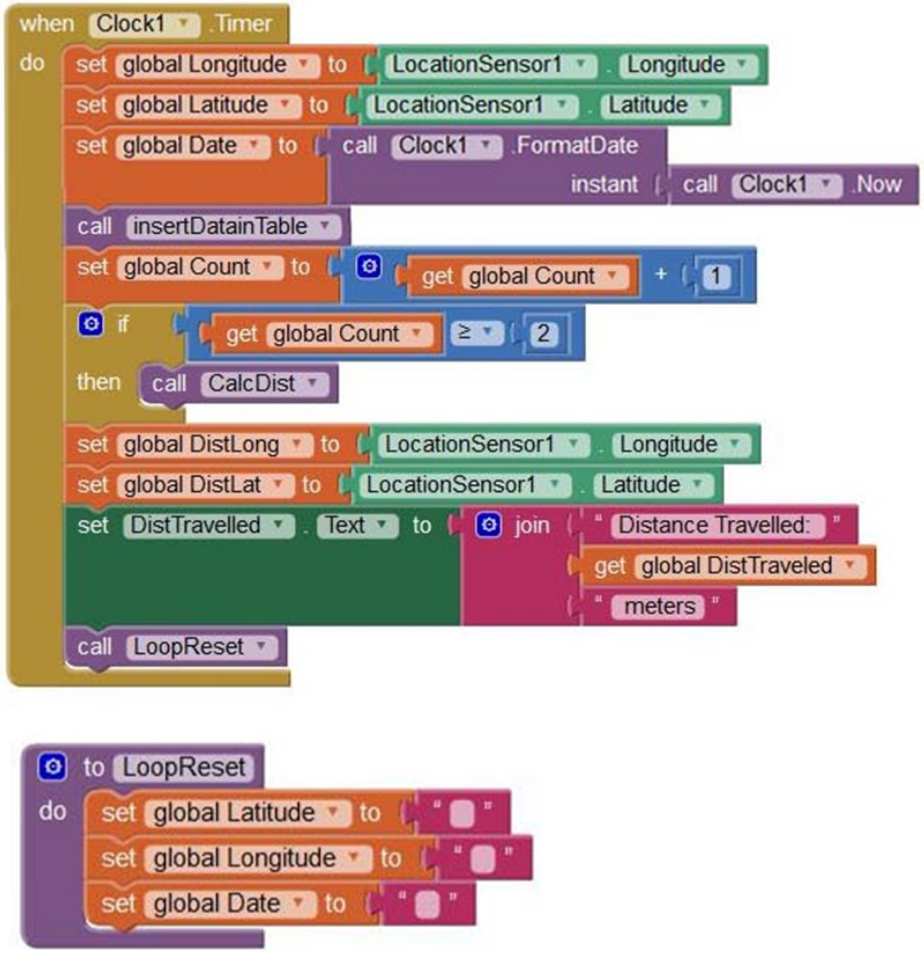
THE CODE: FUNCTIONALITIES

TIMER

We can now program what happens every time the clock timer goes off (every 10 seconds after being activated)

The latitude and longitude and Date/time stamp are set and the procedure Insert Data in Table is called, to tell the app to create the Insert Data query for Fusion Tables. The distance tracking functionality will be discussed in the next section. Then the LoopReset procedure is called.

This procedure resets the Latitude, Longitude and Date so that a new one is entered everytime the timer fires. The Trip Id User ID and Route stay the same.

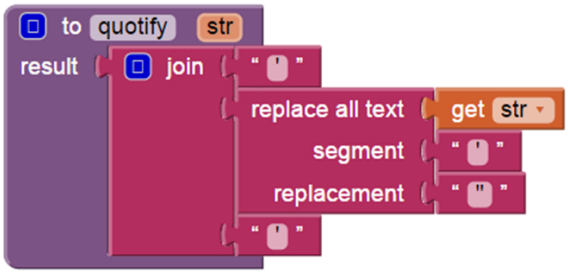


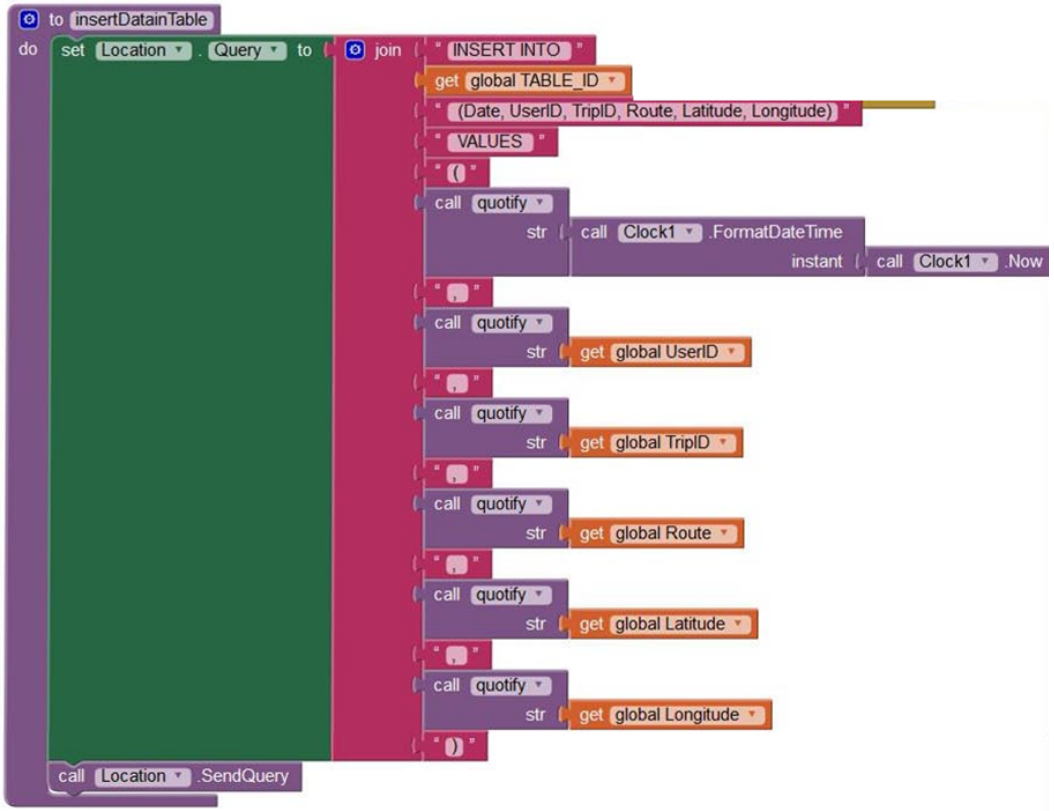
THE CODE: FUNCTIONALITIES

INSERT DATA

The InsertDatainTable procedure in essence builds an SQL query to insert rows into the Fusion Table. To first create the query we must call up the quotify procedure, which inserts quotation marks to the column values, so as to have Fusion Tables accept the output. (Only text strings must be quotified, anything defined as a number in the fusion tables should not be quotified)

The global variables are defined in the above procedures, so they don't need to be specified here. Since Fusion Tables has specific data to insertion requirements, the quotify procedure is used to place " " around the data, The call location.sendquery sends this query to fusion table.





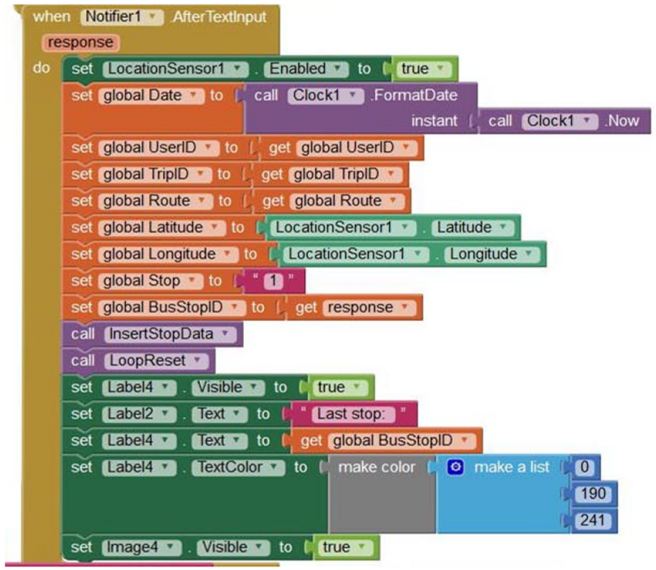
THE CODE: FUNCTIONALITIES

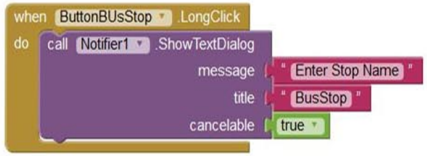
BUS STOP RECORDING

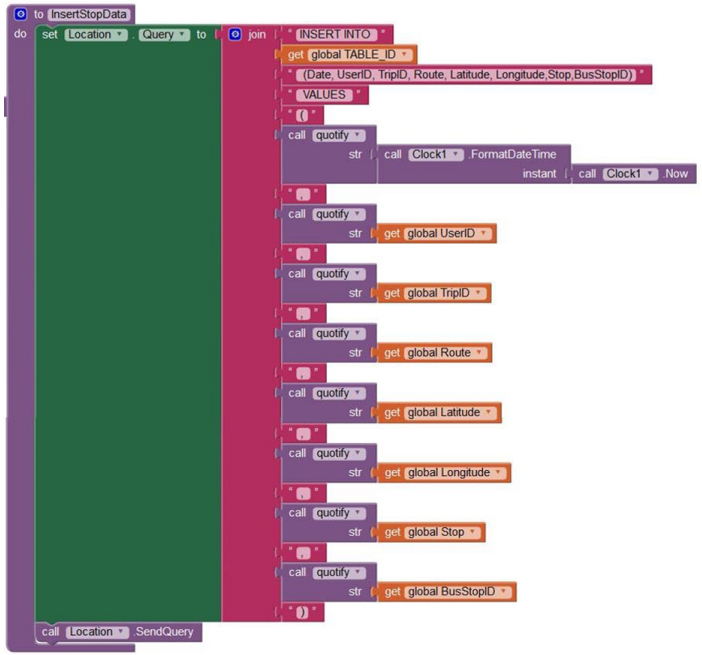
When the bus stop button is held, a popup dialogue (Notifier1) shows up allowing you to enter the name of the bus stop you are at.

When Notifier1.Aftertextinput controls how the app functions after inputting text in the pop-up. It turns on the location sensor to record the location of the bus at that point, the other global variables for all the columns are defined and another procedure InsertStopData is called up. The Global Stop variable is a new column which is set to be 1 if the location being recorded is a Bus Stop but will be empty if it is not.

InsertStopData essentially follows the same rules that the InsertDatainTable procedure followed before.



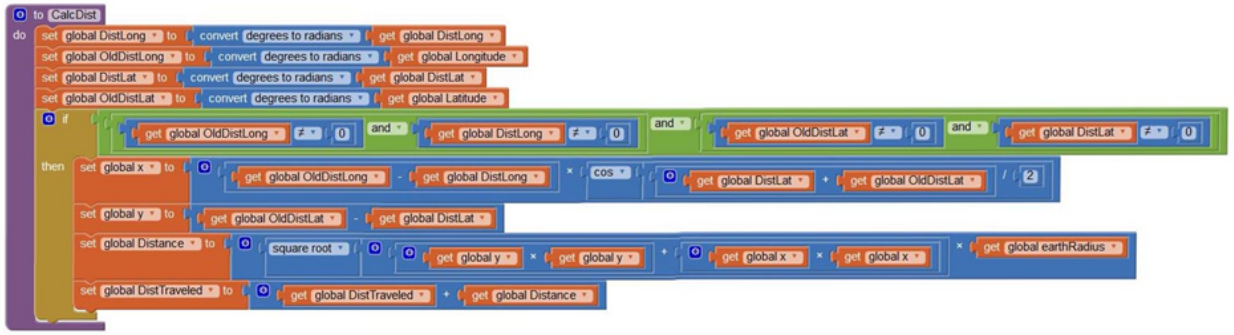




THE CODE: FUNCTIONALITIES

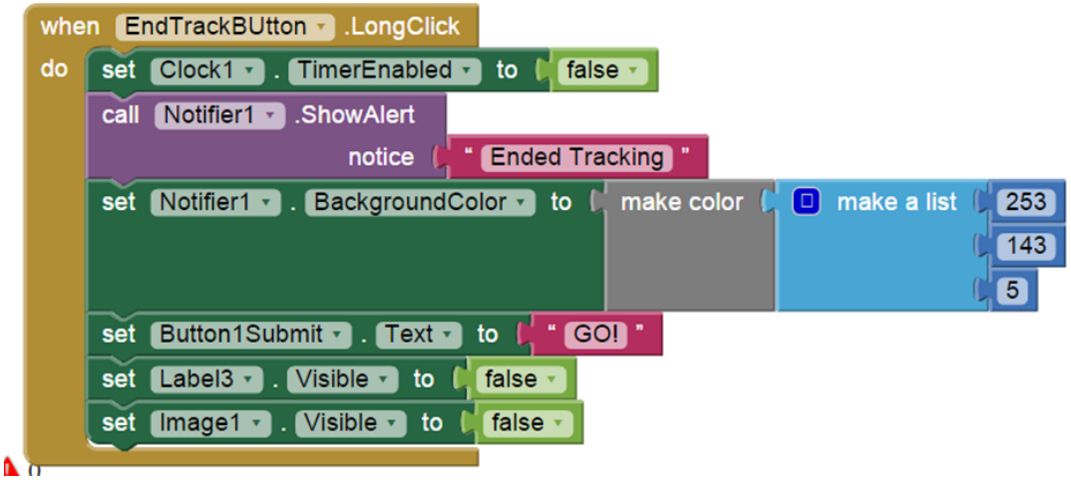
DISTANCE TRAVELLED

The app calculates your distance travelled every 10 seconds, once the timer is fired. It does so by subtracting the previous set of location variables with the current ones.



FINISH TRACKING

To stop tracking, the long-click functionality of the EndTrackButton is defined. The app turns off the clock timer which essentially turns off the functionalities of the app. Eventually this should become an automated feature, which stops tracking once the intended destination is reached.



TESTING

DOWNLOADING APP TO PHONE

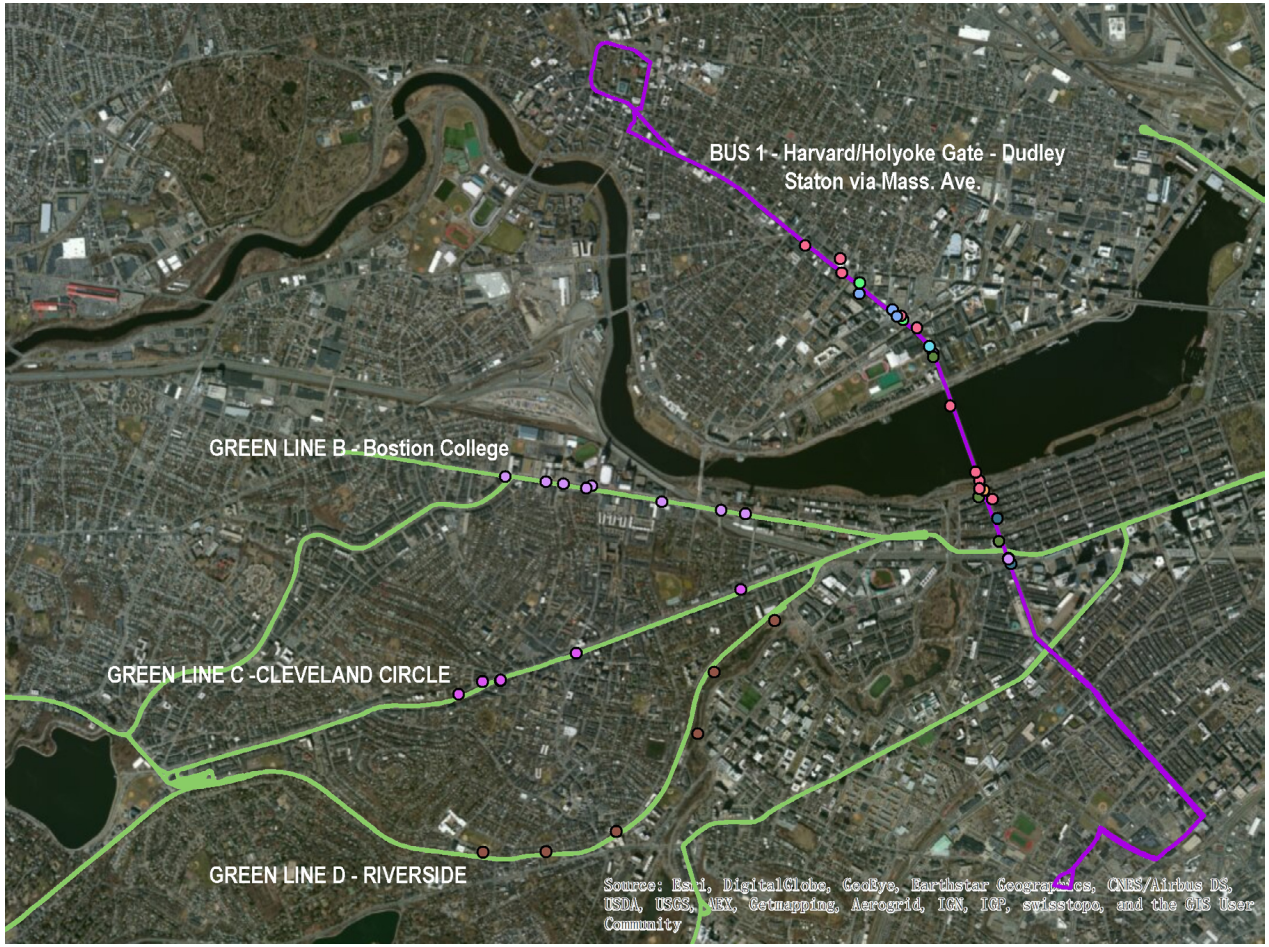
The app can be downloaded from App Inventor as an APK file.

We tested the app in Boston to ensure the following functionalities worked correctly:

- Geotagging
- Input into fusion tables
- Trip ID generation
- Bus Stop recorder
- Distance calculator

There are some remaining issues. For example, our 10 second loop might be too much, since some phones' GPS location sensors update less frequently, resulting in multiple data entries with the same GPS location.

The green and purple lines show the actual bus and train routes tested. The points along the lines are the points we collected through the app.



FUTURE TASKS

A range of tasks remain to bring the application up to a point of broader public launch. Several improvements will be necessary to move from a prototype phase to a fully functional app.

Currently the prototype enables a user to select from just three routes, whereas there are currently well over 120 in the city of Nairobi. Given this variety, a different user interface to simplify selecting a route (a search by name or number, for example) would be required. The final application would also need to support the logging of new routes, ideally with a prominent new button on the Track screen. This would also require new data structures on the back end to collect and store these unclassified points.

The prototype application accesses the Fusion Tables server every 10 seconds, which is accompanied by a screen popup. If the app were scaled up to a wide user base the number of Fusion Tables API calls might become unmanageable and/or expensive to maintain. A more finished version would eliminate this popup and report all recorded points only after the entire route had been recorded. This would both limit the number of times the server is accessed, reducing bandwidth consumption requirements, and make the application user experience more seamless.

Ultimately a dedicated database structure and server should replace the Fusion Tables API for the storage and recording of GPS points and bus stop locations. This would make the application less dependent on the performance of other software and service providers and give its creators more control over how the database is structured and managed. A list of GPS points (potentially in a simple CSV format) from the phone would be pushed to a dedicated server, where a script would process and file them away into the appropriate database.

Since many users are unlikely to remember to press a button confirming the end of their journey, especially if the app is running in the background, an automatic "stop recording" feature would be quite important in a future release. This would work by asking users to provide their destination when selecting the route they intend to travel either through a text search or ideally by dropping a pin on a map. When the bus reached a given radius of this point, the recording would stop automatically. If the user failed to reach the point then location recording would terminate automatically after a certain period.

A finished app would also require a proper user login system that assigns enables each user to select a unique ID and password. Our team did not attempt build one because it would not have contributed to the core functionality of the app and would have required significant effort and research. Once established, the user login would enable each recorded location point to be given a unique combination of user and trip ID numbers.

Providing a real time map of Nairobi with the user's current location as well as the existing matatu routes would make the app more useful and enhance the user experience. The app could then serve as a reference resource in addition to collecting location coordinates and earning the user maTokens. There is also the possibility of incorporating real-time alerts on traffic or accidents, essentially integrating the features of the popular Ma3Route platform.

In testing the app we found that the locations reported to Fusion Tables would not change with every 10 second cycle but instead every three or four cycles, corresponding to every 30-40 seconds. This may be a result of the location sensor component of AppInventor updating on a different cycle or waiting to update until the location has changed by a relatively large margin. Since it is very difficult to uncover the interior workings of AppInventor components we may never know for sure. In any case, the points reported are still sufficient to track most bus routes.

Applications created through AppInventor have the limitation of not being able to run in the background, that is, while other apps are running or while the home screen of the app is displayed. As a consequence, the app will also cease recording points when the phone's display is turned off. Since requiring potential users to both keep their screens on and to refrain from using their phones for other things is unreasonably burdensome, a different development platform would be required.

AppInventor proved to be an excellent development platform considering the need to develop a prototype rapidly and with relatively little team programming expertise. While easy to use, however, it also places significant limitations on how the app functions, its level of customizability and even the ability to identify bugs. Any app released to the public at large in Nairobi should be developed in a more traditional programming environment with a team of dedicated coders.

BACKEND AND MANAGEMENT

A future research team would be responsible for developing the algorithms and scripts necessary to process the data points and update the existing database of Matatu routes. A first step would involve filtering points based on their accuracy and removing any obvious outliers. Locations outside the city, highly uncorrelated points and points corresponding to unrealistic travel speeds would all be eliminated. A separate algorithm would then snap the points to a network of roads, similar to what is done in the directions functionality of Google Maps. Once this is established, another routine would compare the paths recorded by users to the existing database of Matatu routes. If paths from enough users deviate significantly from an existing route, its official path would be edited to reflect the deviation. The thresholds and criteria for updating and maintaining the map would probably have to be established through experience, with people reviewing proposed changes for a period until there is enough confidence to automate the process. Similarly, if enough users report paths along a new route, the route would be added to the database and reflected in updates to the map.

Bus stops points would be reported in a like manner, filtered for accuracy, and then processed a range of algorithms. If a number of reported stop points cluster around a given location, the algorithm would update this location on the official map. The shifting location of bus stops in response to local conditions may complicate the process, requiring a pickup zone to be specified instead of a specific stop location.

Many other data mining and analytics opportunities exist, including tracing the speed and reliability of the routes, how frequently routes change, and the collection behavior of its users. Assuming the data can be sufficiently anonymized, records from the app could become a rich travel record dataset for transportation research, a resource that companies and governments currently pay millions of dollars to obtain.

Oversight of the effort would remain with MIT, but the University of Nairobi would take over active management and maintenance of the code. In the long term, a public policy organization like KIPPRA could manage the project.

MARKETING

The app's marketing strategy would involve a variety of different approaches distributed through a number of different channels.

The management team should aim to cultivate a partnership with the existing service Ma3Route, a popular web-based platform that crowd-sources for transport data and provides users with information on traffic, gives text-based Matatu directions and driving reports. It already has a large and engaged user and would nicely complement the functionality of Umati. In exchan ge for the marketing exposure from being featured prominently on their service, the management team could share our map data, which would enable them to dramatically improve Matatu directions.

Establishing partnerships with local retailers is critical to effectively marketing the app. We would target larger grocery chains and telecom companies that are well recognized by a broad section of the population. This could include, for example, grocery stores Nakumatt and Chandarana and the telecoms Airtel and Safaricom. In exchange for offering coupons or discounts on their products and services in exchange for MaTokens, we would give them free exposure and advertising through the app's platform. People earning MaTokens would have a strong incentive to patronize these businesses.

The management team could also appeal on the grounds of helping a cause that universally benefits the citizens of Nairobi.

A grassroots-style marketing campaign could also be launched on social media, with efforts targeted first at the tech community. It could emphasize both the social good of improving our knowledge of Matatu routes in addition to the incentive of collecting MaTokens, which can be spent in a variety of ways. Student groups based out of the University of Nairobi could also organize spontaneous get-togethers in popular Nairobi hangouts, wearing shirts with the app's QR code and handing out promotional materials.

THE TEAM

From left:

Marcel Williams
marcelw@mit.edu

Carolina Morgan
caromor@mit.edu

Rida Qadri
rqadri@mit.edu

Fei Xu
fei_xu@mit.edu