OpenMash Volunteer Ride Sharing

5 June 2013

The OpenMash Volunteer Ride Sharing (VRS) application demonstrates how the OpenMash integration bus extends the reach of VistA by leveraging SaaS offerings such as the Google Application Engine (GAE) and Google Apps.

Temporal and geographic data are two domains that occur throughout applications in general. The VRS application demonstrates how both of these domains can leverage VistA specific data with other SaaS resources in the Cloud to provide increased value that leverages non-traditional assets such as the Volunteer community. The extended collaboration of public, private, and community is enabled and scaled by secure SAAS infrastructure while maintaining necessary security and privacy concerns.

The VRS use case is meant to be illustrative of how an open source, open architecture framework enables the immense value of existing SaaS offerings to be integrated seamlessly with VistA. By providing simple, secure, reliable, and flexible integration for VistA, OpenMash makes the benefits of Google Apps available to system architects seeking to integrate VistA.

# Volunteer Ride Share Functionality

The VRS is a web application running in the Google Application Enginer (GAE) cloud. Volunteers register their user profile including available times and geographic location. Similarly, patients register their geographic location. VistA events mediated by the OpenMash integration bus trigger RESTful VRS services that match the patient with a volunteer who is available in their geographic area with availability during the time window of the scheduled meeting. The volunteer receives a notification via email and clicks a link to confirm or reject the match. Both volunteer and patient are notified of the confirmed meeting. VistA is updated via a callback to the OpenMash bus.

# Benefits

Allowing VistA to leverage Cloud assets such as Google provides two categories of benefits. First, there are IT benefits provided by the Google App Engine. It provides trusted, high availability, elastic infrastructure to the community. This includes network, server, and application infrastructure that is proven and integrated for SaaS delivery.

Google Apps provides additional domain specific functionality. There are many domains addressed by Google Apps including email, collaboration, scheduling, and geographic information. See the additional resources listed below for other Google Apps benefits. Although GAE and Google Apps are great Cloud services, this document is *not* intended to be a recitation of Google Apps benefits. Rather, it is intended to demonstrate how these benefits are made available to VistA via OpenMash and their relevance to the Scheduling component in particular.

# Resources

<http://www.google.ca/enterprise/apps/business/resources/library.html>

<http://www.google.com/apps/intl/en/landing/choosegoogle/tenreasons.html>

<https://cloud.google.com/products/>

Attached PDF

Google Apps Data Sheet

Google Security Whitepaper

Google Apps and Cloud Platform Audit and Certification Summary

Total Economic Impact of Google Apps

# Google Apps Domain

OpenMash has setup the GoogleApps domain at <https://mail.google.com/a/test3.sheepdoginc.ca/>

Account credentials are listed below:

**Admin User:** [admin@test3.sheepdoginc.ca](mailto:admin@test3.sheepdoginc.ca)

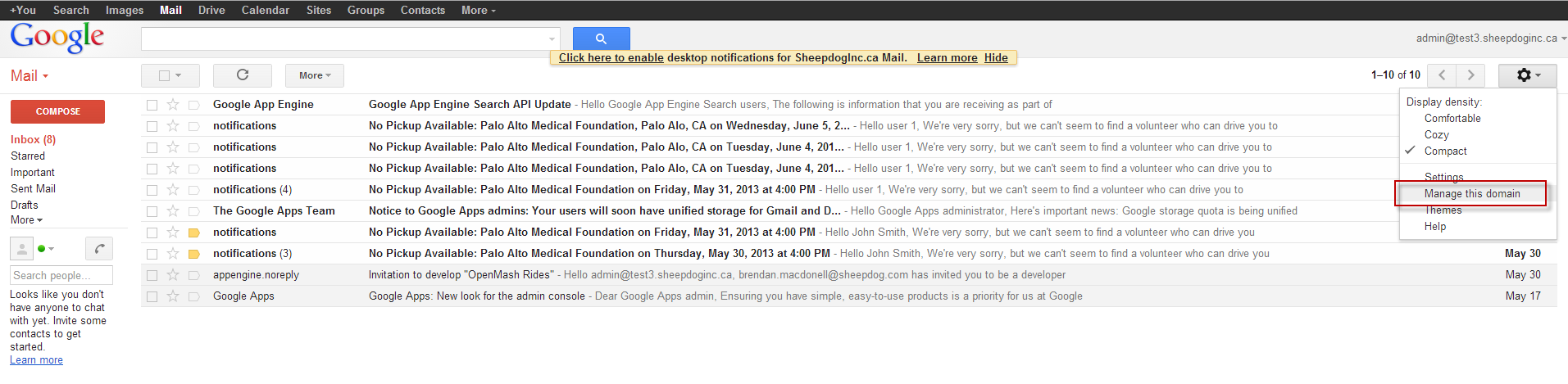
**Password:** vatalend2n1

**Normal Users:** [user1@test3.sheepdoginc.ca](mailto:user1@test3.sheepdoginc.ca) to [user50@test3.sheepdoginc.ca](mailto:user50@test3.sheepdoginc.ca)

**Password:**vatalend2n1

You can log in to any of these accounts by visiting https://www.gmail.com and entering the full email address as the username, or alternately by visiting <https://mail.google.com/a/test3.sheepdoginc.ca/> and entering a username and password. As a normal user, the user interface should be pretty much indistinguishable from a standard gmail account.

As an admin user, you can access the domain control panel by clicking the gear button in the top right-hand corner, and selecting "Manage this Domain". The control panel should be fairly self-explanatory - for example, to manage calendar resources, you would click "Settings" in the top bar, select "Calendar" from the left-hand navigation bar, and then select the "Resources" tab.



This information is provided simply to allow evaluators full access to the environment and to demonstrate that the full power of Google Apps can be made available to VistA users. No additional administration is necessary for purposes of the demo.

# Volunteer Ride Sharing App

The AppEngine application is running at [http://openmash-rides.appspot.com](http://openmash-rides.appspot.com/). I've set up some test accounts as follows (all passwords are **vatalend2n1**):

Patient: [**user1@test3.sheepdoginc.ca**](mailto:user1@test3.sheepdoginc.ca)

Volunteer: [**user2@test3.sheepdoginc.ca**](mailto:user2@test3.sheepdoginc.ca)

Administrator: [**admin@test3.sheepdoginc.ca**](mailto:admin@test3.sheepdoginc.ca)

External Administrator:[**openmash.superuser@gmail.com**](mailto:openmash.superuser@gmail.com)

Feel free to use any of the rest of the first 10 accounts (user1 .. [user10@test3.sheepdoginc.ca](mailto:user10@test3.sheepdoginc.ca).)

Users 3-13 are volunteers

Others are patients

Both the patient and volunteer are located in the California bay area. Though we don't have a client for the application, you can use the following debug page to trigger appointment notifications:

<http://openmash-rides.appspot.com/tasks/notification/?patientEmail=user1@test3.sheepdoginc.ca&appointmentAddress=Palo%20Alto%20Medical%20Foundation,%20Palo%20Alo,%20CA&appointmentTime=2013-06-06T05:00:00-0300>

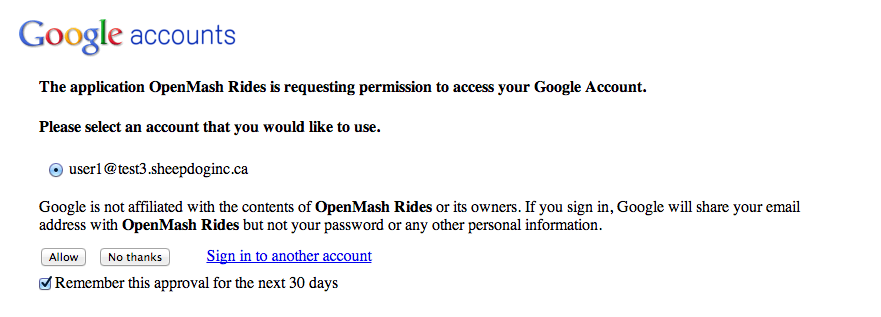
If you want to trigger a batch export of user stats Fusion Tables, visit <http://openmash-rides.appspot.com/tasks/snapshot/>

For-profit Google Apps domains are unable to plot maps in Fusion Tables. To work around this limitation, an external administrator account was set up in gmail. Non-profit entities can to have this restriction lifted. See <https://support.google.com/fusiontables/answer/1657096> for more information.

# Integration Tests

## Integration Test 1 – Create and Persist Patient Profile

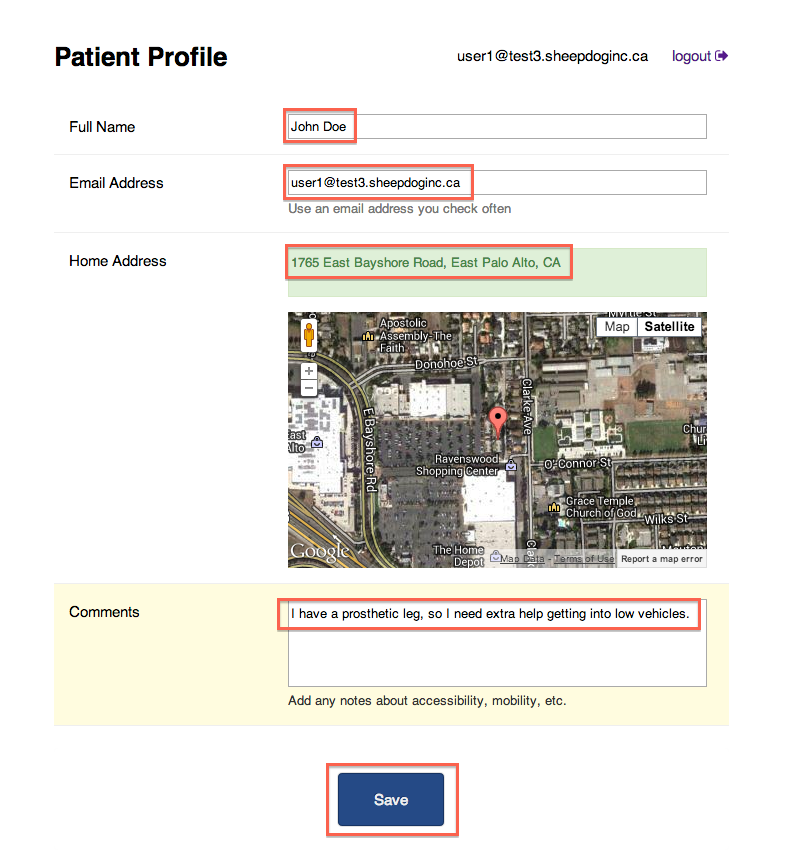
1. Create a new Google gmail account.
2. Click on [http://openmash-rides.appspot.com](http://openmash-rides.appspot.com/) for initial login



1. Select patient or volunteer. Start with a Patient.



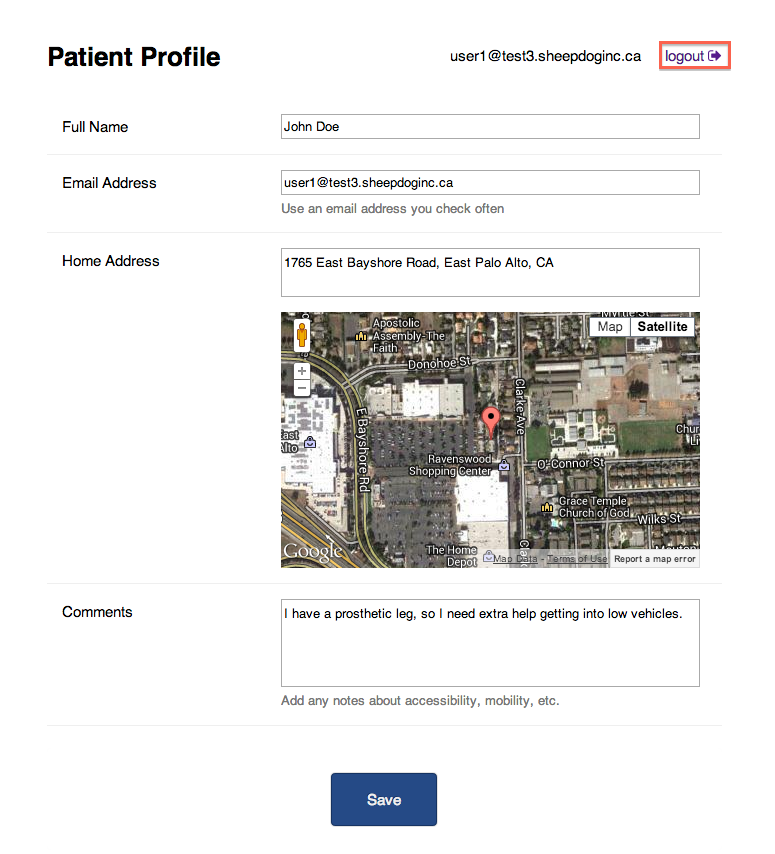
1. Enter patient profile and click Save.



1. After saving changes click logout.
2. Log back in to verify account info has been persisted.



1. Confirm information has been persisted. Then log out.



## Integration Test 2 – Create and Persist New Volunteer Profile

1. Create a Google Gmail account.
2. Click on [http://openmash-rides.appspot.com](http://openmash-rides.appspot.com/) for initial login
3. Login to the VRS application using a different Google account than the one used in Integration Test 1.



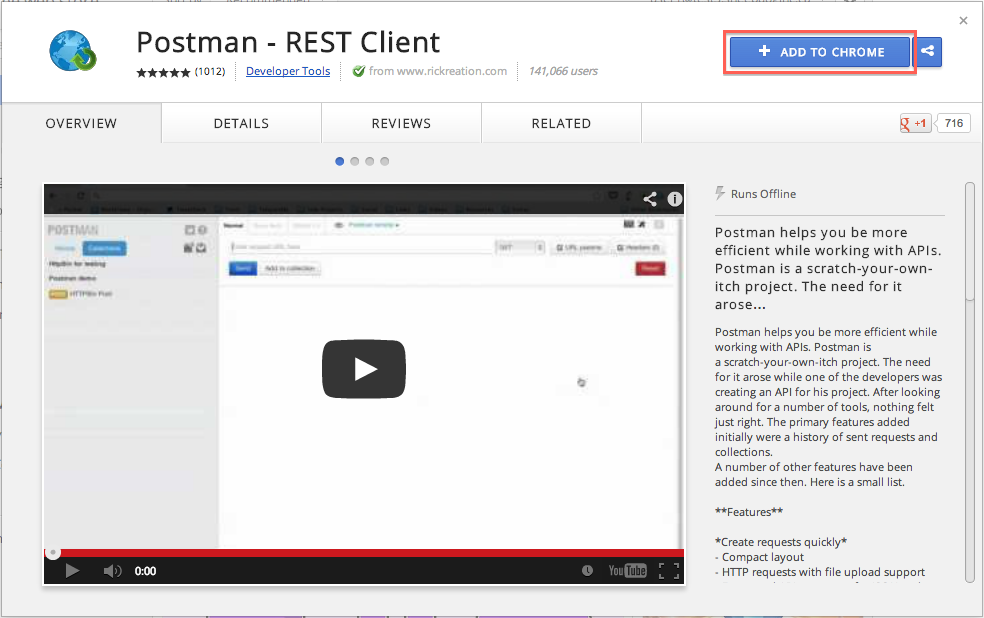
1. Select “I am a volunteer”.
2. Fill out the volunteer profile. Select a valid maximum distance and click and drag a box to indicate when the volunteer will be available. Be sure to enter an address in the San Francisco area because that is where the patients are located in the test database. Click Save.



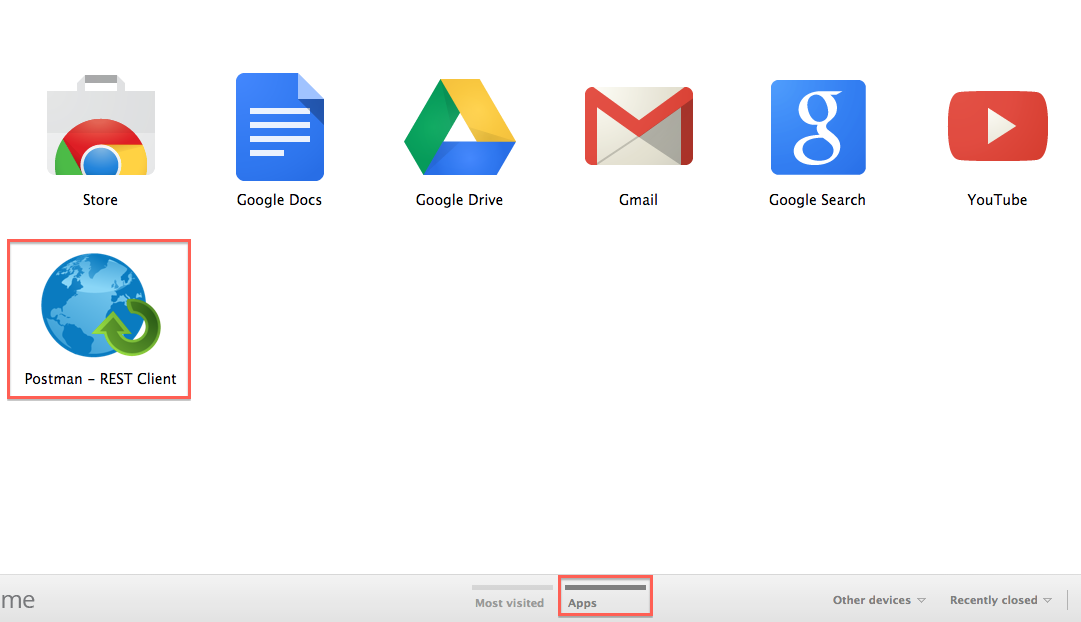
1. After saving changes click logout.
2. Log back in to verify account info has been persisted.
3. Confirm information has been persisted. Then logout.

### Integration Test 3 – No Matching Volunteer and Patient

1. Start Google Chrome.
2. Install the Postman REST client from <https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?hl=en>



1. Open a new tab in Chrome, select the “Apps” section, and click the “Postman – REST Client” icon.

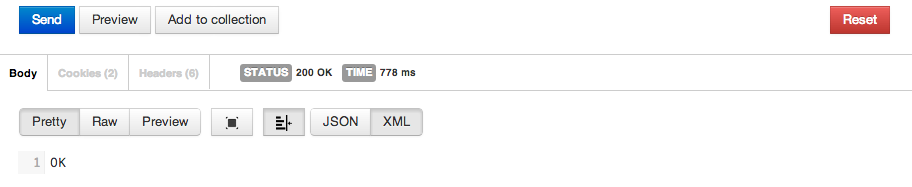


1. In Postman, select the “Normal” tab, select “POST” as the request method, and enter the URL “<http://openmash-rides.appspot.com/resources/notification/>”. Set the encoding to “x-www-form-urlencoded”, and add the following key / value pairs:
   1. *patientEmail*, with the email address of the patient created in Integration Test 1;
   2. *appointmentAddress*, paired with the address of a nearby medical clinic; and
   3. *appointmentTime*, paired with a time that was not marked as available in the volunteer's calendar in Integration Test 2.

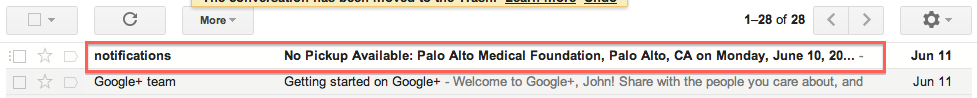
Hit “Send” to submit the request.

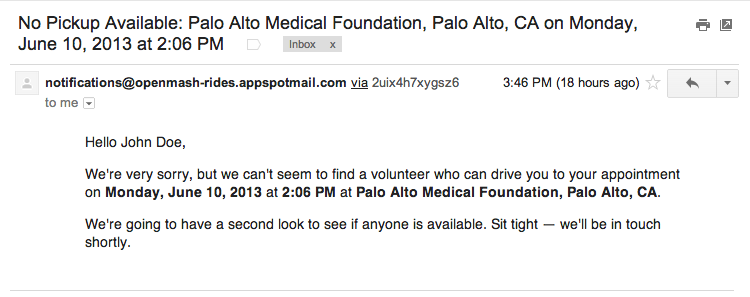


1. Verify that the server responds with the status “OK”



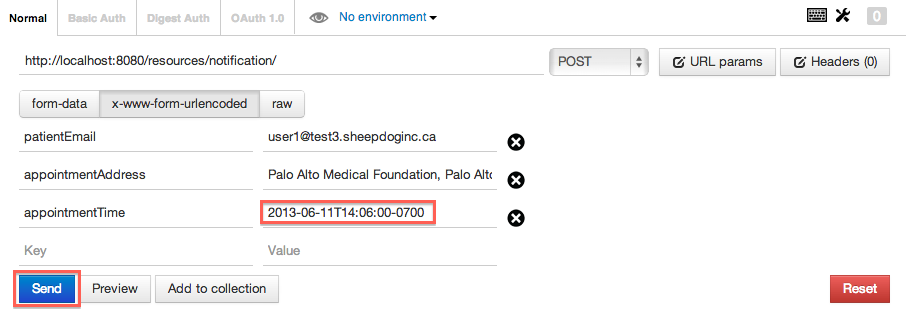
1. Log into the gmail account of the patient created in Integration Test 1.
2. Verify that the patient has received an email informing them that a volunteer is not available to pick them up.

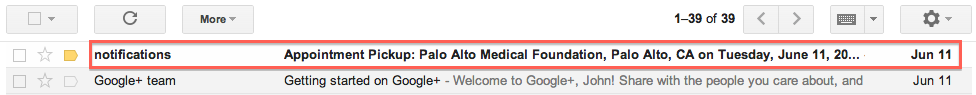


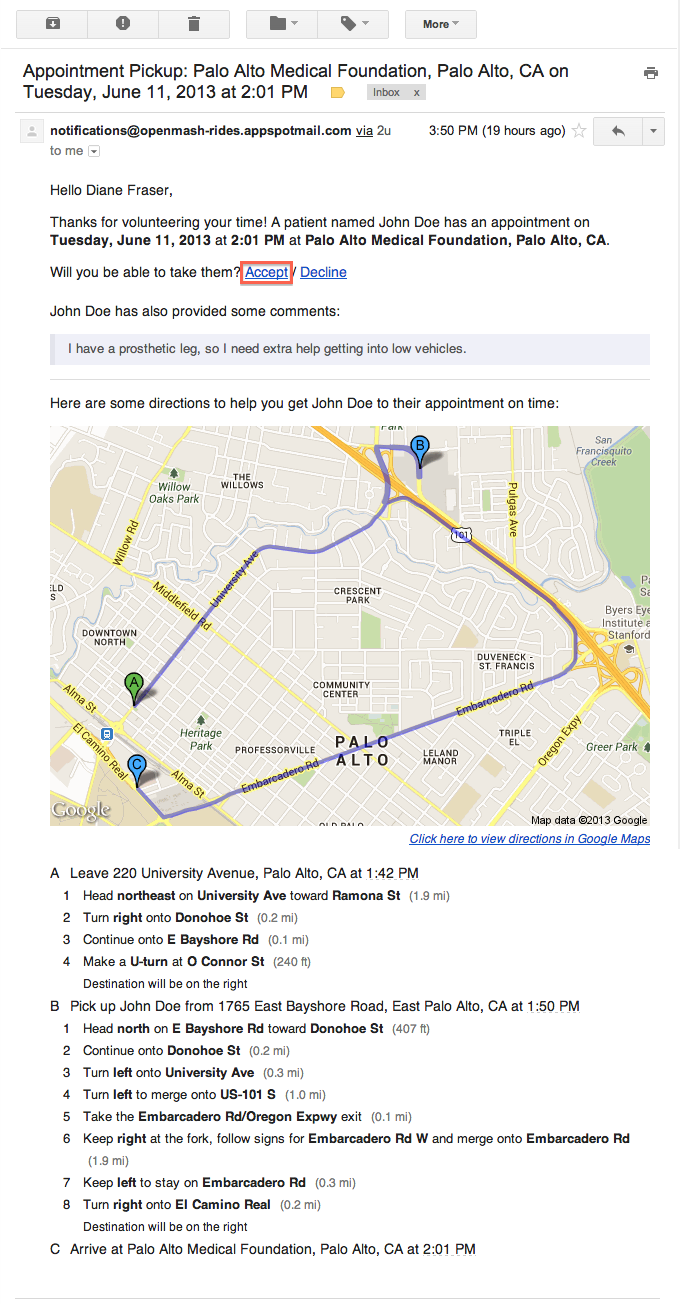
1. Log out of the patient's account.

### Integration Test 4 – Matching Volunteer and Patient

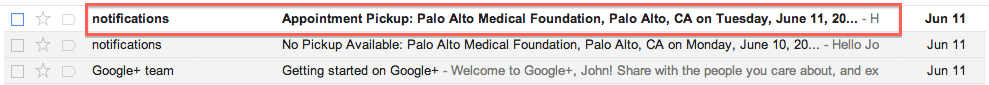
1. Open the Postman REST client.
2. Enter an appointment time during which the volunteer created in Integration Test 2 is available, and hit “Send” to send a notification to the server for a new appointment.

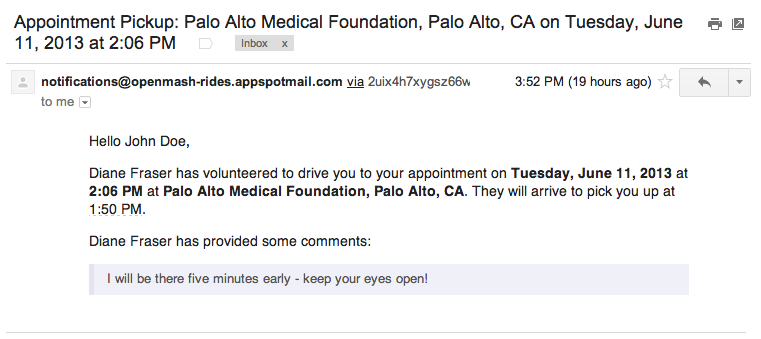


1. Verify that the server returns an “OK” response.
2. Log into the gmail account of the volunteer created in Integration Test 2.
3. Verify that the volunteer received a request to take the patient to their appointment.
4. Examine the email to verify that details such as the pickup time, date, and location are correct, and ensure that the directions given to the volunteer make sense. Click “Accept” to accept the pickup request.



1. Log out of the volunteer's account, and log into the account of the patient created in Integration Test 1.
2. Verify that the patient has received an email informing them that a volunteer will pick them up for their appointment. Check that the information in the notification email is correct.

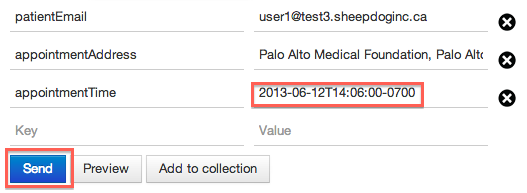




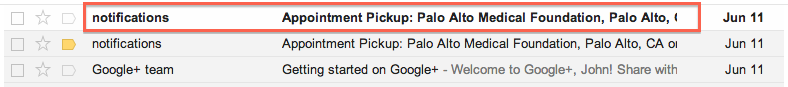
1. Log out of the patient's gmail account.

### Integration Test 5 – Matching Volunteer Rejects Opportunity

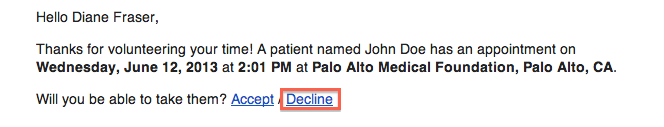
1. Open the Postman REST client.
2. Enter an appointment time during which the volunteer created in Integration Test 2 is available, and hit “Send” to send a notification to the server for a new appointment. Use a different time than you used in Integration Test 5 as the volunteer is considered busy in a time period for which they have accepted a pickup request.



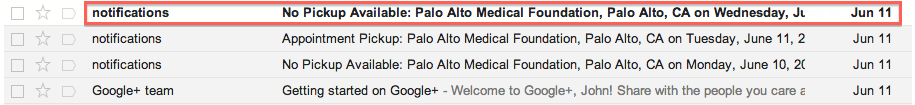
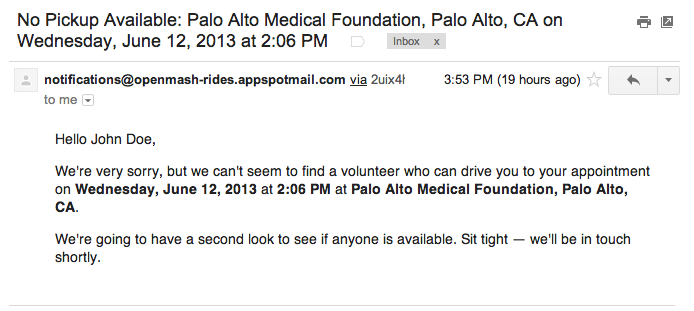
1. Verify that the server returns an “OK” response.
2. Log into the gmail account of the volunteer created in Integration Test 2.
3. Verify that the volunteer has received an email requesting that they pick up the patient in the new timeslot.



1. Open the appointment pickup request and click “Decline”.



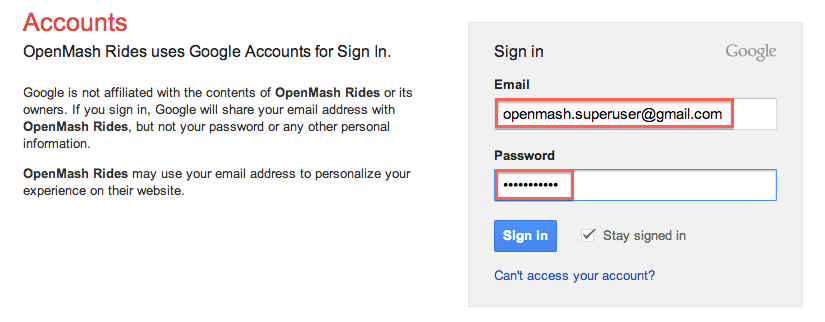
1. Log out of the volunteer's gmail account and log in as the patient created in Integration Test 1.
2. Verify that the patient received a notification that no volunteer is available to pick them up.



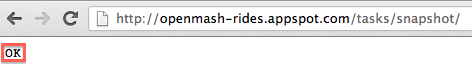
1. Log out of the patient's gmail account.

### Integration Test 6 – Data Visualization – Ride Sharing History

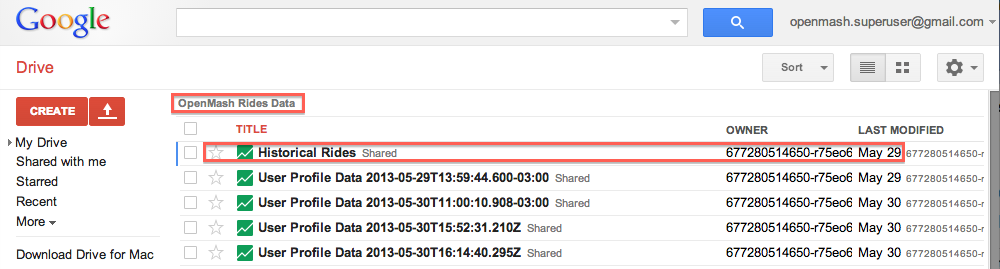
1. Visit [http://openmash-rides.appspot.com](http://openmash-rides.appspot.com/) to log in.
2. Log in as an outside administrator: use the username “[openmash.superuser@gmail.com](mailto:openmash.superuser@gmail.com)” and the password “vatalend2n1”.



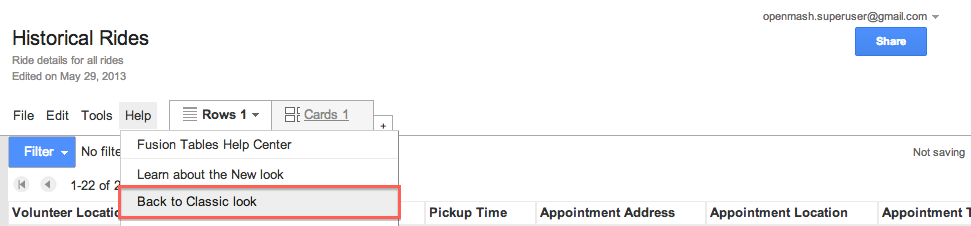
1. Visit [http://openmash-rides.appspot.com/tasks/snapshot/](http://openmash-rides.appspot.com/tasks/snapspot/) to manually trigger a batch data snapshot to Fusion Tables. Verify that the server sends an “OK” response.



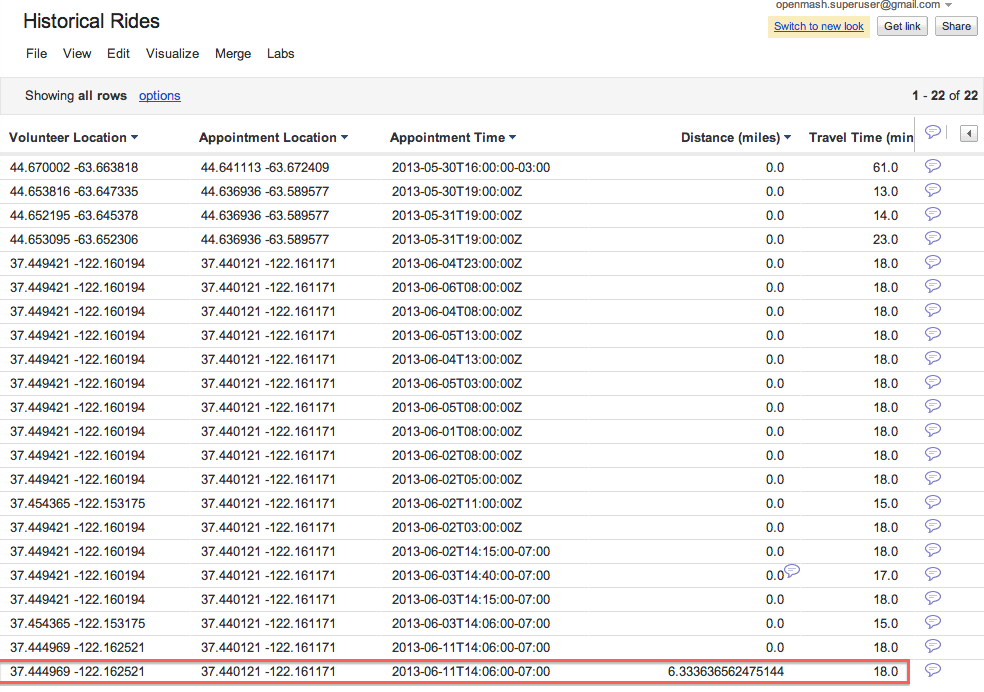
1. Return to [http://openmash-rides.appspot.com](http://openmash-rides.appspot.com/). Click on the “stats” link in the page header.
2. Ensure that a new tab was opened showing the contents of a folder called “OpenMash Rides Data”. Click on the file named “Historical Rides”.



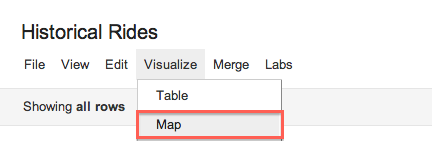
1. In order to visualize data in Fusion Tables, you will need to use the “Classic look” theme. Select the “Help” menu, and click “Back to Classic look”



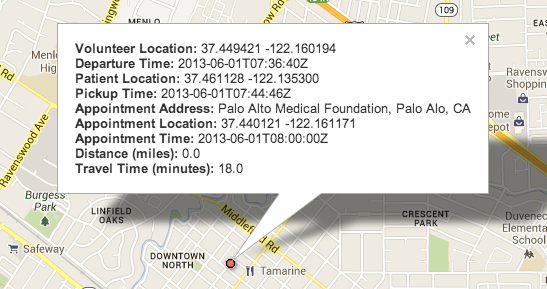
1. Verify that the ride from Integration Test 4 appears at the end of the file. The rides from tests 3 and 5 should not be present.



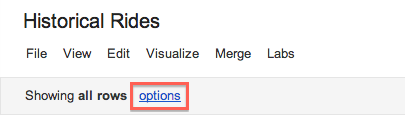
1. Plot the geolocation data on a map by selecting the “Visualize” menu and selecting “Map”.



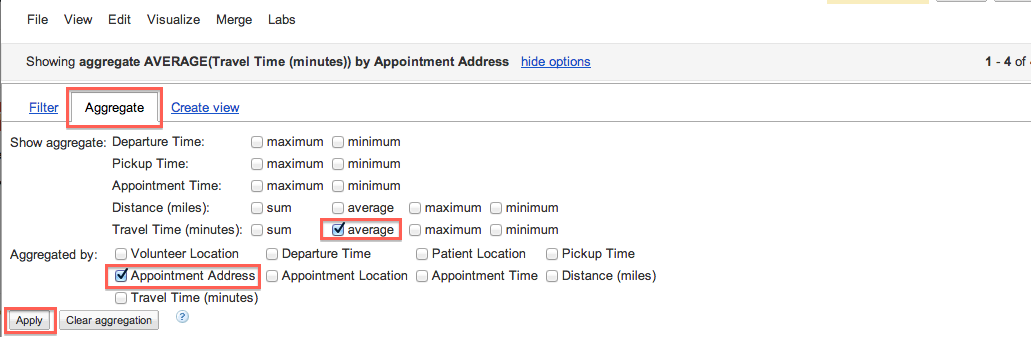
1. Verify that you can plot the volunteer, patient, and appointment location data for each ride.
2. Ensure that clicking on a point plotted on the map displays all details of the associated row.



1. Select “Visualize” from the menu, and click on “Table”
2. Click on the “options” link



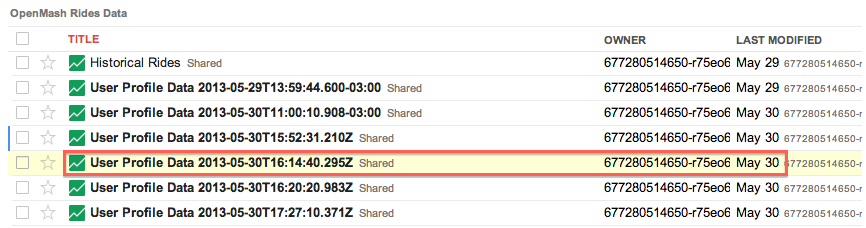
1. Click on the “Aggregate” tab. Tick the “average” checkbox from the travel time row, tick the “Appointment Address” aggregate, and click “Apply”.



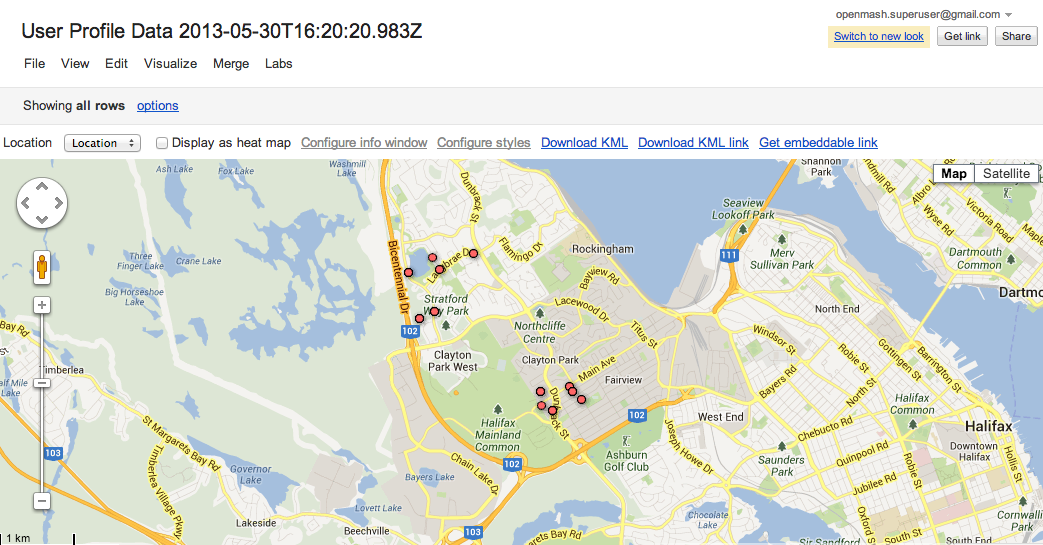
1. Verify that the average travel time to get to an appointment is displayed for all distinct appointment addresses in the system.
2. Log out of the administrator's account.

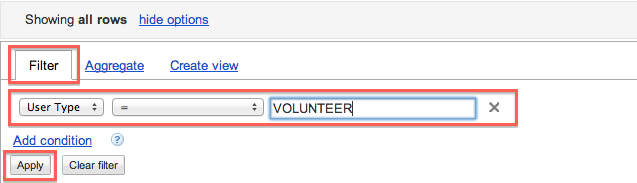
### Integration Test 7 – Data Visualization – Comparing Volunteer and Patient Distribution

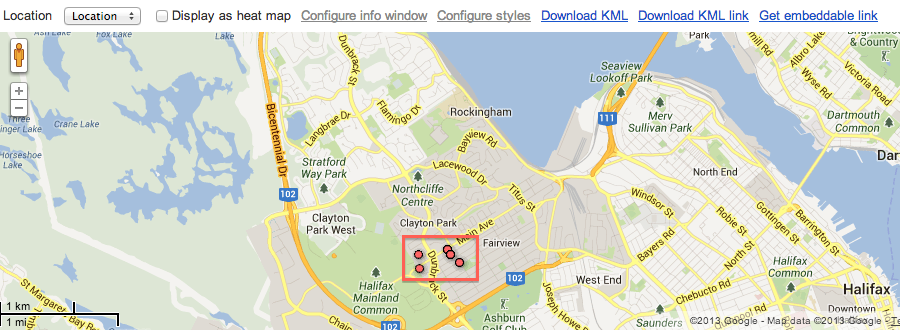
1. Visit [http://openmash-rides.appspot.com](http://openmash-rides.appspot.com/) to log in.
2. Log in as an outside administrator: use the username “[openmash.superuser@gmail.com](mailto:openmash.superuser@gmail.com)” and the password “vatalend2n1”.
3. Click on the “stats” link in the page header.
4. Click on the file named “User Profile Data 2013-05-30T16:20:20.983Z”.



1. Click “Visualize” and select “Map”. The locations of all patients and volunteers will be plotted together on the same map.
2. Zoom in on the cluster of profiles in the Halifax, Nova Scotia area.



1. Select “options” and add a filter to display only users who are volunteers. Click “Apply”.
2. Verify that all of the volunteers are located in the Clayton Park / Fairview area.



1. Change the user type filter to “PATIENT” and click “Apply”.



1. Verify that most of the patients are located in the Clayton Park West area.

