OTP SUM: OTP Integration of Transit with Shared-Use Mobility Real-Time and Data Enhancements

Mobility on Demand Sandbox Program
Quarterly Report Q2 2017

July 31, 2017

TriMet.org/MOD

For more info and up-to-date progress, please go to www.trimet.org/mod. This dashboard was created by TriMet to provide a snapshot of the MOD Sandbox project's progress.

Challenges Addressed by Project

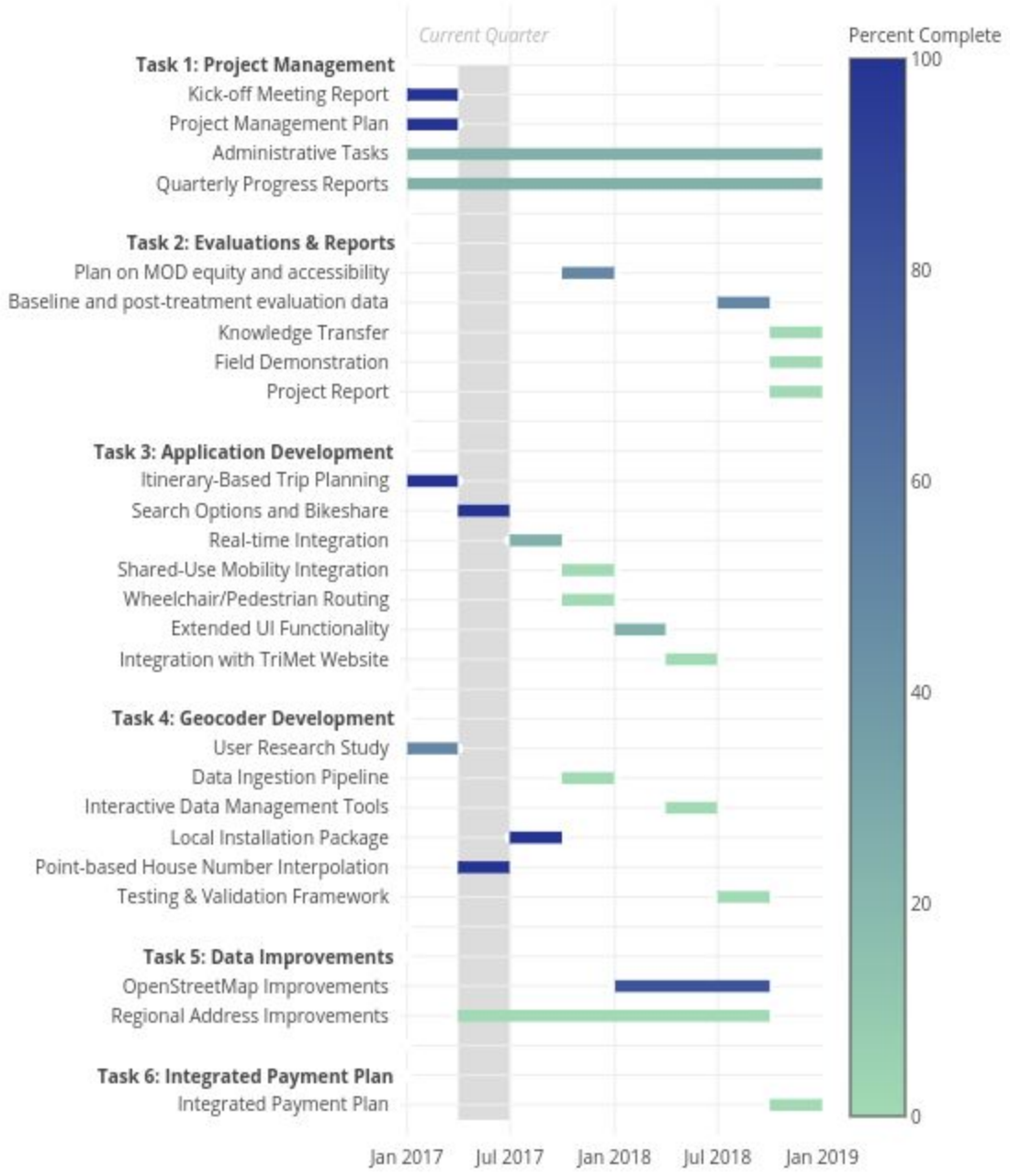
- OpenTripPlanner (OTP) does not currently incorporate shared-use modes.
- Address location for trip origins and destinations are a main requirement for trip planning, however, existing options are inadequate or cost prohibitive for government.
- Accessible trips are a challenge due to the lack of data available on the accessibility of pedestrian infrastructure and the absence of these features in a trip planner.

Anticipated Outcomes, Benefits, Impacts

- Extend the OpenTripPlanner code base to support the integration of transit trip planning with shared-use mobility modes, such as bike share and transportation network companies (TNCs), as well as updated real-time transit information.
- Implement a fully functional and comprehensive open geocoder built off the existing Mapzen Pelias geocoder. A non-proprietary and non-restrictive option for address locating would substantially lower the barrier to entry for many transit systems to offer trip planning and can achieve significant cost savings for transit agencies, government agencies, and the public.
- TriMet, in collaboration with the OpenStreetMap community, established best practices for representing accessibility information and will build out this accessibility information in the OSM network and provide a model for replicating this work in other regions.

TABLE OF CONTENTS

Project Scope and Budget Status



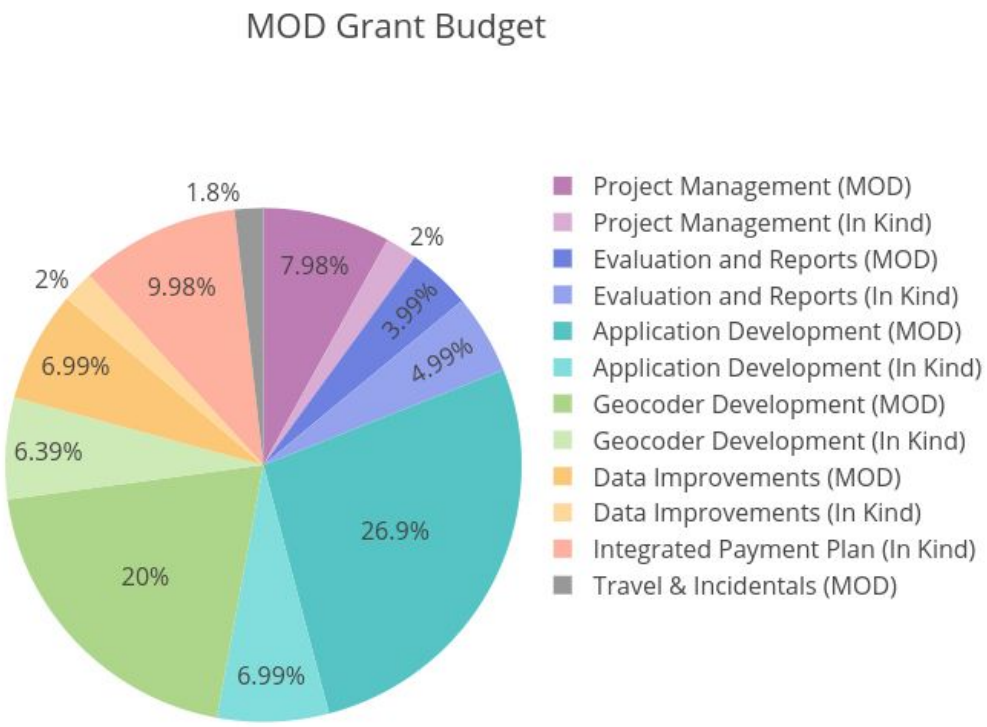
The above gantt chart illustrates the tasks and status of deliverables.

TriMet's funding allocation from the FTA of $678,000 is matched with 32% of in-kind contributions, totaling over $1 million.

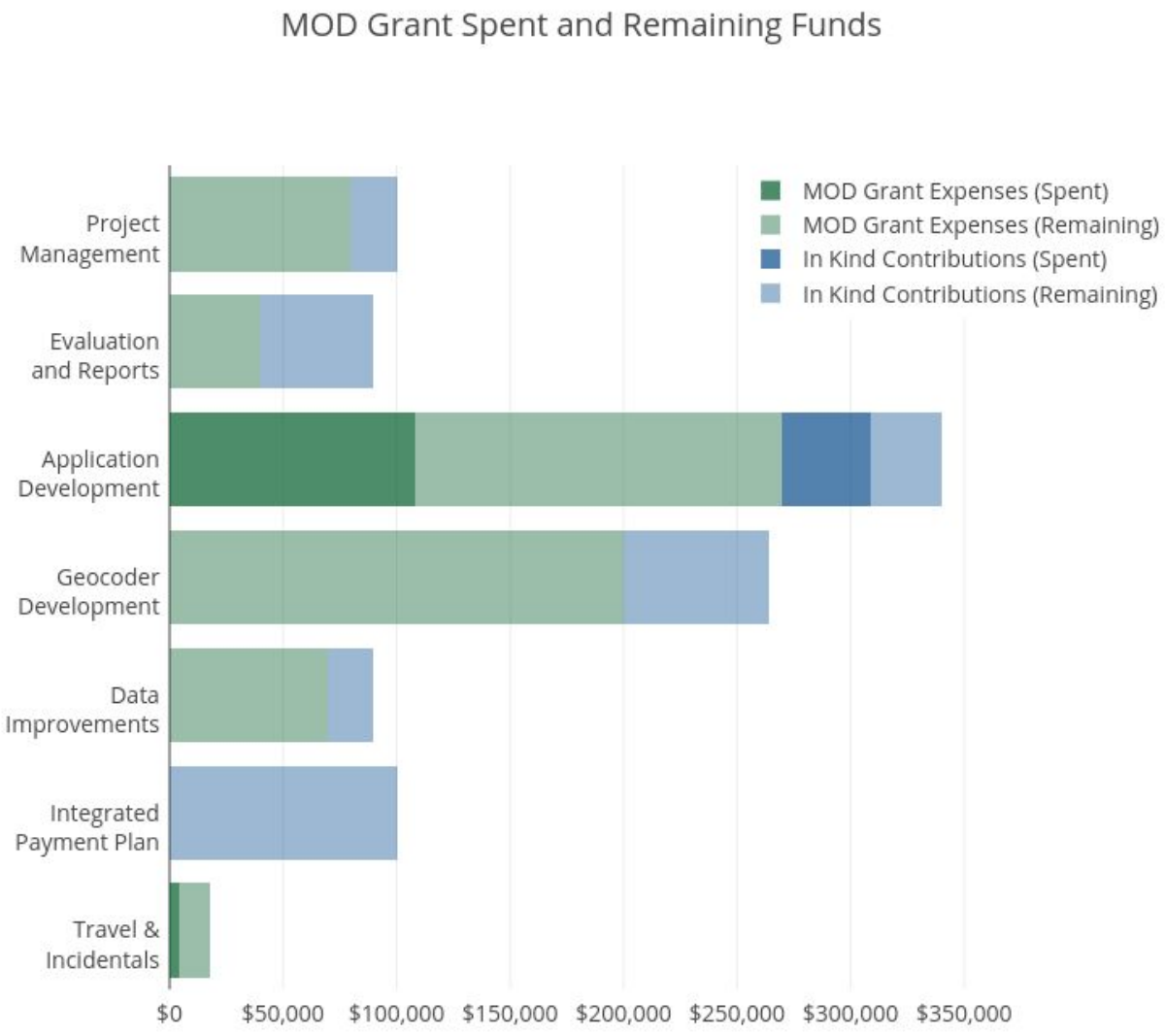
Of the $678,000 that TriMet received, $58,620 (8.6% of allocated grant funds) has been spent thus far.

The expenditures through Q1 2017 are as follows:

- $1,122 (2% of allocated in-kind funds) of in-kind contribution spent toward Evaluation and Reports;
- $54,000 (20% of allocated grant funds) spent toward Application Development;
- $4,620 (26% of allocated grant funds) spent toward Travel & Incidentals.



The above pie chart illustrates the amount and percentage of the budget allocated to each of the main tasks, divided into MOD grant expenses and in-kind contributions.



The above bar chart shows the current amount spent for each of the tasks.

Task 1: Project Management

TriMet's OTP Integration of Transit with Shared-Use Mobility Real-Time and Data Enhancements have been underway since January. All milestones and deliverables have been met and we are on schedule.

Quarterly Deliverables

Deliverables for this quarter are in the form of ongoing tasks that include scheduled weekly meetings and administrative tasks.

Quarterly Progress

Task progress includes:

- weekly scheduled meetings (slack or webinars) to ensure continued communications;
- use of Trello for project management;
- a dedicated and open TriMet MOD Project Google drive for project management;
- use of InVision for application interface development and review;
- and the continued update of the online project dashboard available to the public at TriMet.org/MOD to ensure transparency.

Task 2: Evaluations and Reports

The FTA requires the following project evaluations and reports: Evaluation Plan and Report, Equity and Accessibility Plan, Knowledge Transfer, Field Demonstration, Final Project Report.

Quarterly Deliverables

- There were no scheduled deliverables for this task during this quarter.

Quarterly Progress

- TriMet has worked with Booz Allen on finalizing the MOD Evaluation Logic Model located on the TriMet MOD Project Google Drive: https://docs.google.com/spreadsheets/d/1YIhKyHAyLr_f9ttwgSlnR_uw57npR-lC00EzKEKxMgs/edit#gid=1886309523

- TriMet's Evaluation Plan and Report is located on the TriMet MOD Project Google Drive: https://drive.google.com/open?id=17Ok54d4-IqYNdY0dw96Soy1Lc05u_jjpi0G-yOvhukQ

  TriMet's Evaluation Plan is focused on the following:
  - Trip Planner - time & cost comparisons, increased feasibility of routes (evaluation will begin upon release of beta application)
  - Pelias Geocoder - match rate and accuracy improvements **(Appendix A - Task 2 Geocoder Evaluation Quarter 2 Report)**
  - User Satisfaction - application interface and travel options (evaluation will begin upon release of beta application)

Task 3: Application Development Status

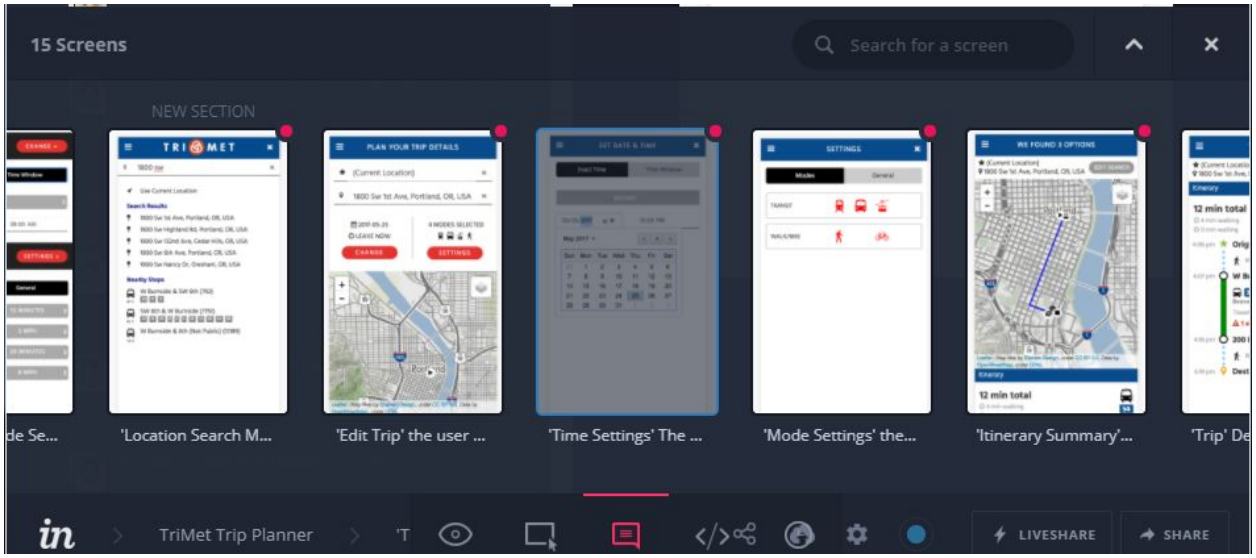
Significant progress has been made toward integrating shared-use mobility modes into the existing OpenTripPlanner application. Conveyal has designed prototypes of the new mobile-first app in InVision, with iterative improvements based on feedback from TriMet design staff.

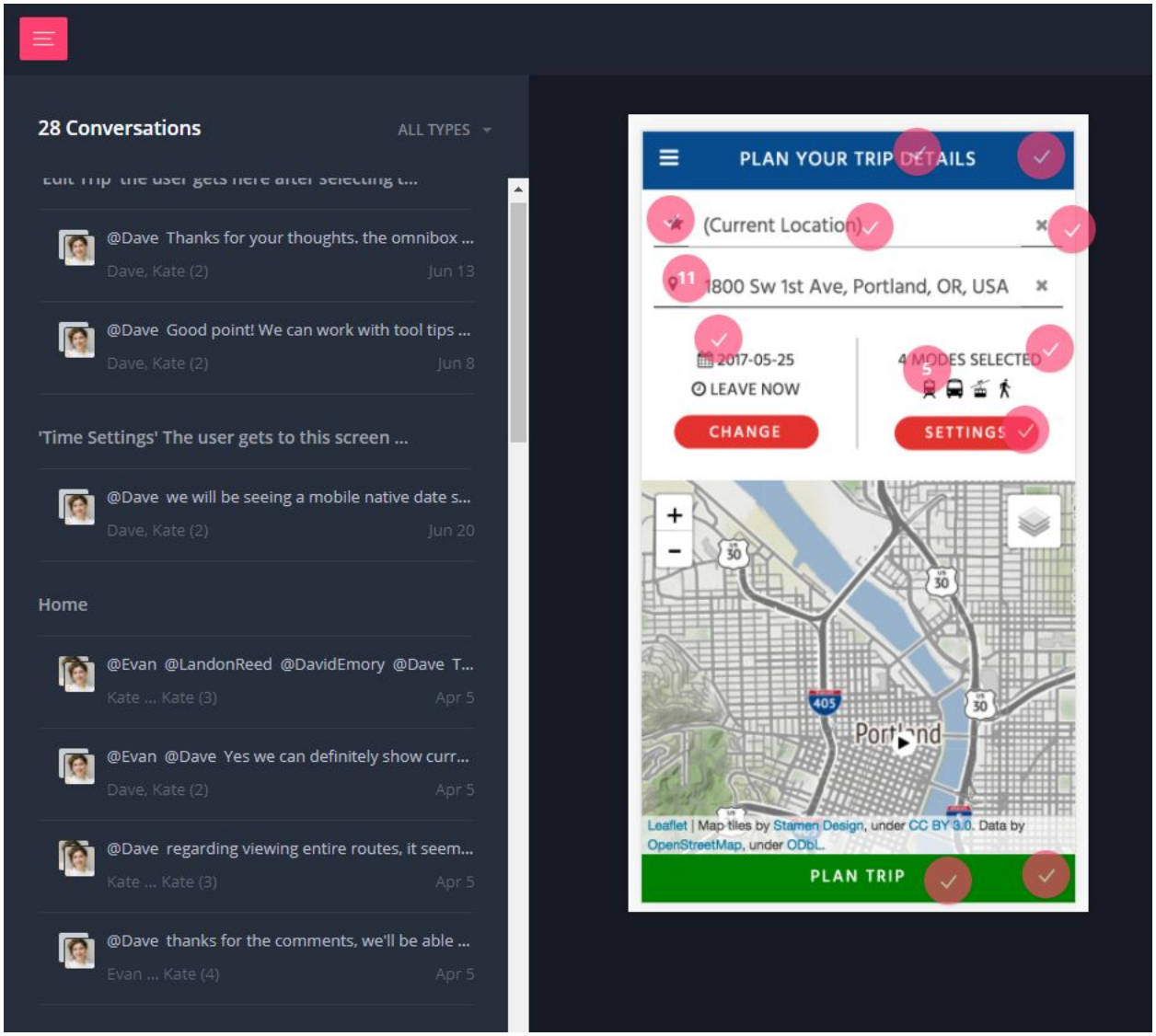
Quarterly Deliverables

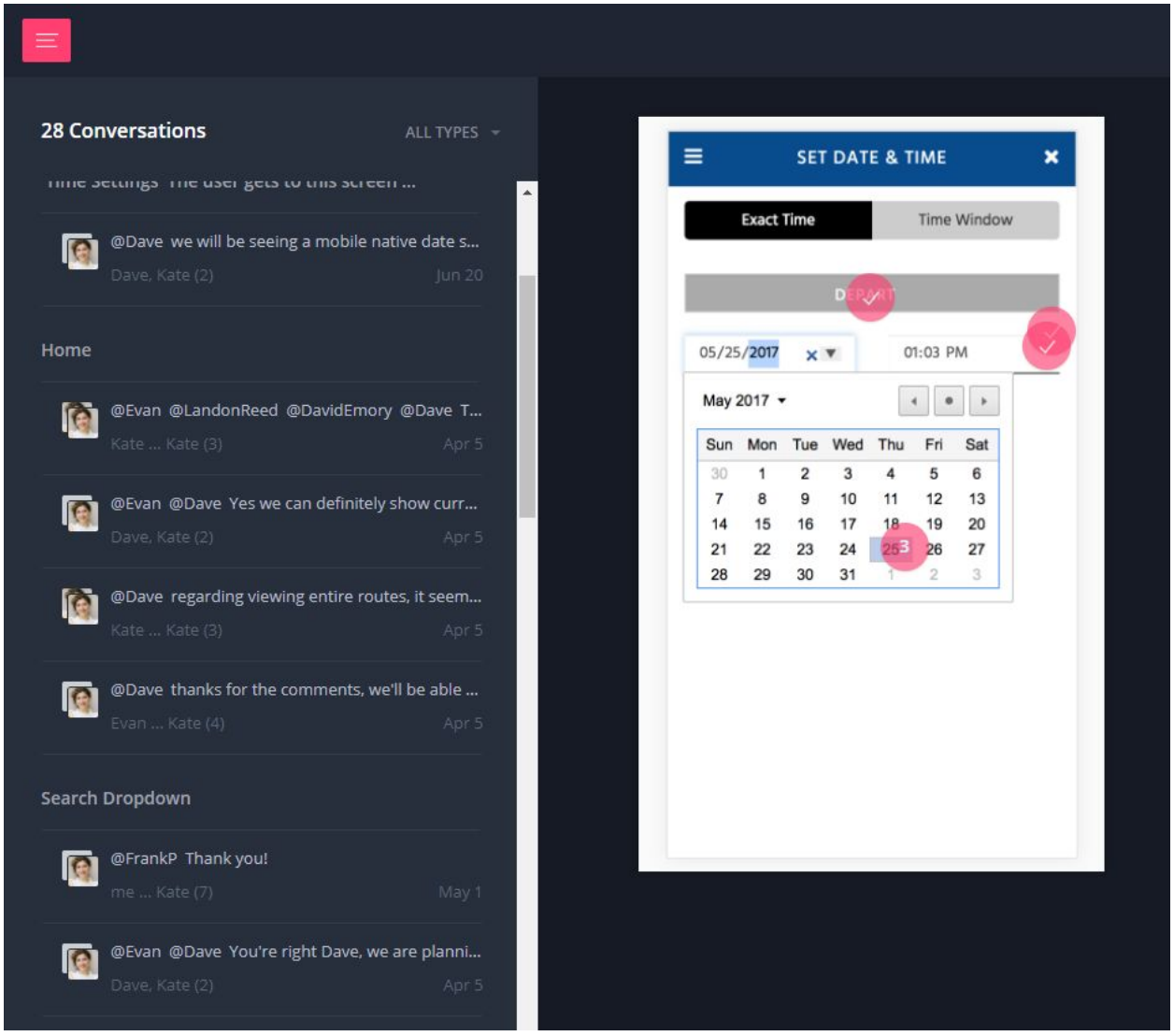
- Search Options and Bikeshare (**Appendix B - Task 3 Milestone 2 Documentation**). It was delivered and signed off on Thursday, June 22, 2017. The code for this deliverable is available on a private GitHub site until production.

Quarterly Progress

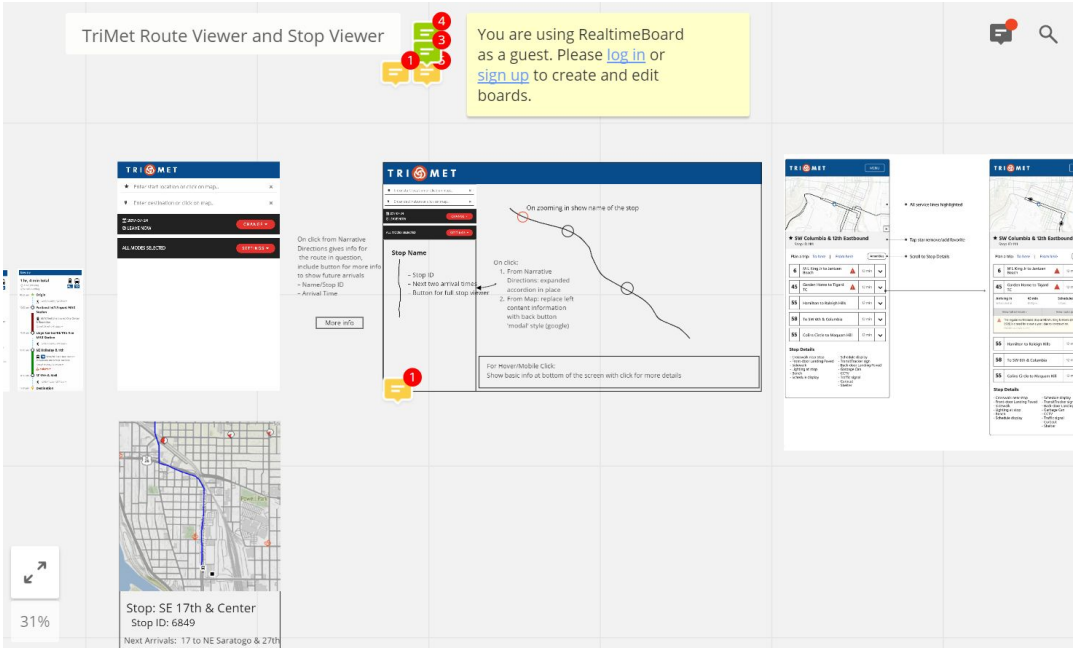
- In addition to the completed milestone, the user interface design continues to be refined in InVision:


15 Screens
Search for a screen
NEW SECTION
'Location Search M...
'Edit Trip' the user ...
'Time Settings' The ...
'Mode Settings' the...
'Itinerary Summary'...
TriMet Trip Planner
LIVESHARE
SHARE


28 Conversations
ALL TYPES
@Dave Thanks for your thoughts. the omnibox ...
Dave, Kate (2) Jun 13
@Dave Good point! We can work with tool tips ...
Dave, Kate (2) Jun 8
'Time Settings' The user gets to this screen ...
@Dave we will be seeing a mobile native date s...
Dave, Kate (2) Jun 20
Home
@Evan @LandonReed @DavidEmory @Dave T...
Kate ... Kate (3) Apr 5
@Evan @Dave Yes we can definitely show curr...
Dave, Kate (2) Apr 5
@Dave regarding viewing entire routes, it seem...
Kate ... Kate (3) Apr 5
@Dave thanks for the comments, we'll be able ...
Evan ... Kate (4) Apr 5
PLAN YOUR TRIP DETAILS
(Current Location)
1800 Sw 1st Ave, Portland, OR, USA
2017-05-25
LEAVE NOW
4 MODES SELECTED
CHANGE
SETTINGS
Portland
Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
PLAN TRIP


28 Conversations
ALL TYPES
@Dave we will be seeing a mobile native date s...
Dave, Kate (2)
Jun 20
Home
@Evan @LandonReed @DavidEmory @Dave T...
Kate ... Kate (3)
Apr 5
@Evan @Dave Yes we can definitely show curr...
Dave, Kate (2)
Apr 5
@Dave regarding viewing entire routes, it seem...
Kate ... Kate (3)
Apr 5
@Dave thanks for the comments, we'll be able ...
Evan ... Kate (4)
Apr 5
Search Dropdown
@FrankP Thank you!
me ... Kate (7)
May 1
@Evan @Dave You're right Dave, we are planni...
Dave, Kate (2)
Apr 5
SET DATE & TIME
Exact Time
Time Window
05/25/2017
01:03 PM
May 2017
Sun Mon Tue Wed Thu Fri Sat
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31 1 2 3

- We are also using RealTime Board for live, remote whiteboarding sessions:



Task 4: Geocoder Development

Pelias is a non-proprietary and non-restrictive option for address locating that is an important requirement for trip planning. This task includes the implementation of a reference framework for government agencies to auto-feed their authoritative address data into a publicly accessible geocoding service.

Quarterly Deliverables

- Local Installation Package (**Appendix C - Task 4 Milestone 4 Documentation**).

Quarterly Progress

Local Installation Package Description:
Implement a simple setup system for agencies wanting to install a local instance of the search engine using either all or a subset of the OpenAddresses/OpenStreetMap/Who's on First data. This can allow for easy testing of the specified data sources. It would also provide a solution for those needing higher rate limits than the public Mapzen Search API can support. Must at minimum support the operating systems identified as critical by user research: Windows/Ubuntu/MacOS.

Task 5: Data Improvements

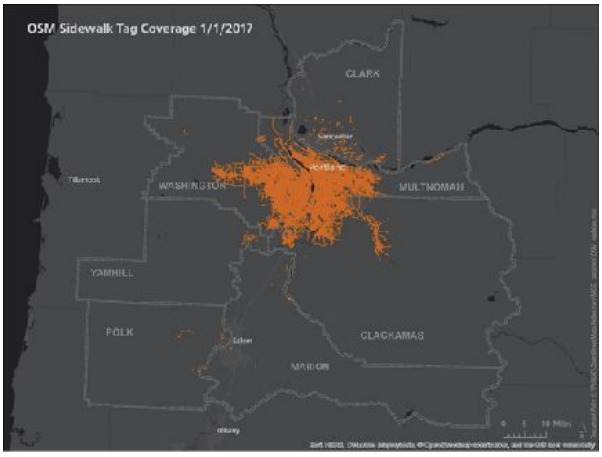
Improve OpenAddresses and OpenStreetMap (OSM) in support of comprehensive trip planning and geocoding (address matching).

Quarterly Deliverables

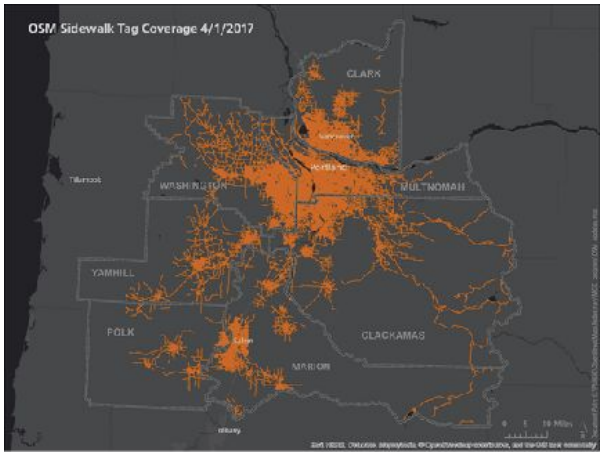
- There were no scheduled deliverables for this task during this quarter.

Quarterly Progress

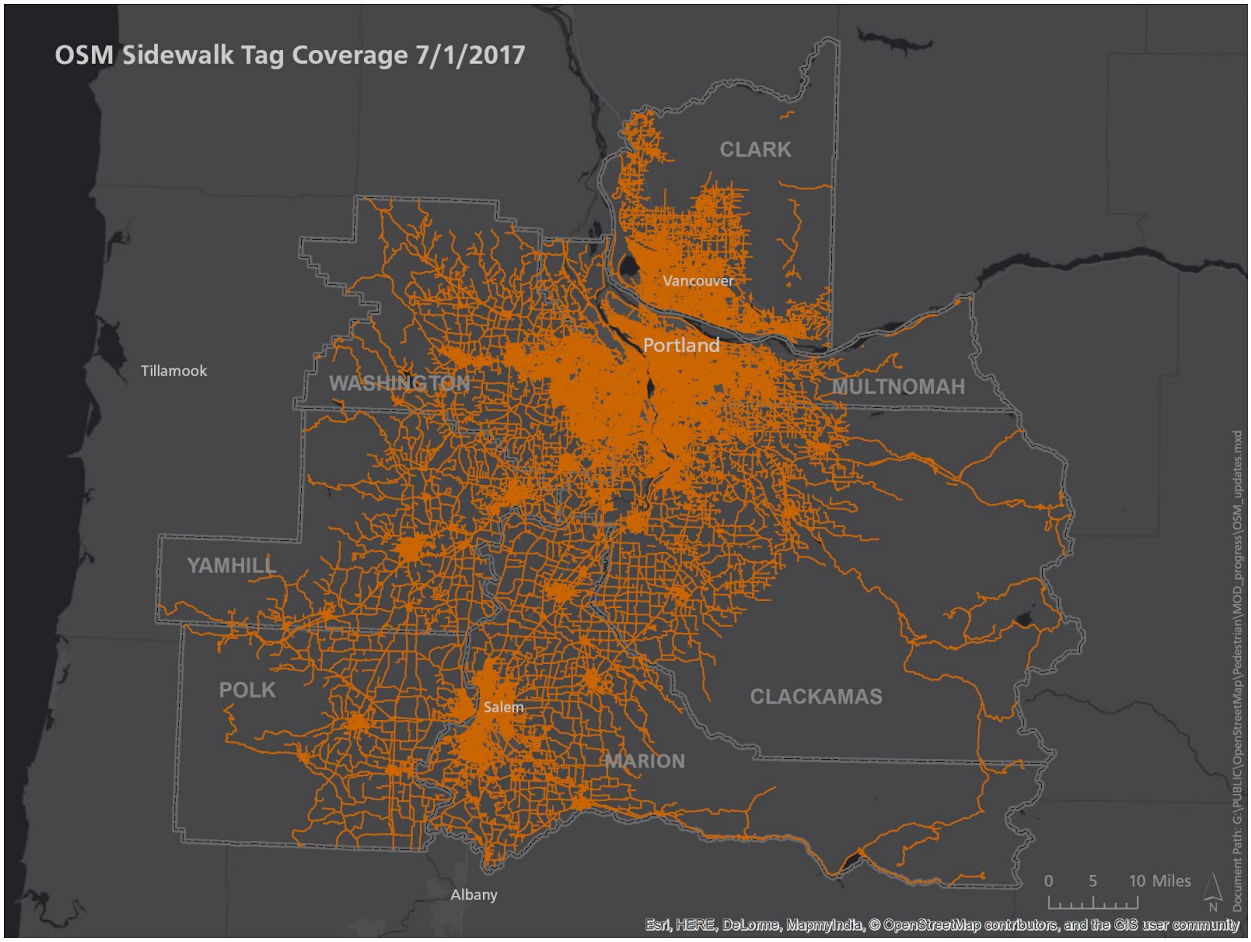
- A substantial amount of work was performed on OpenStreetMap (OSM) data improvement.
- Work will continue throughout the length of the project.
- Improvements to the OpenStreetMap sidewalk data have been made: 2,909.4 additional road miles have been tagged in this quarter.
- The percentage of appropriate streets tagged with sidewalks has increased from 72.2% to 85.7% during this quarter.



Start of Project, 1/1/17: 35.7% complete



End of 2017 Q1, 4/1/17: 72.2% complete



End of 2017 Q2, 7/1/17: 85.7% complete

Task 6: Integrated Payment Plan

As a partner on this project, moovel will facilitate compatibility with their planned booking and payment features so customers can plan and pay for their trips in one app.

Quarterly Deliverables

- There were no scheduled deliverables for this task during this quarter.

TriMet's current mobile ticketing app, TriMet Tickets

Meetings and Events

To date, TriMet has organized and/or participated in the following conferences, workshops or meetings:

- January 18 – 19, Project Kickoff Workshop; Portland, OR
- February 1, NIST Global City Teams Challenge Super Action Cluster Summit, Presentation; Portland, OR
- April 5, TransITech Conference, Presentation; San Antonio, TX
- April 12, Shared-Use Mobility Center, Webinar Presentation
- April 20, Metro RLIS Stakeholders Meeting, Presentation; Portland, OR
- April 20, Mobility on Demand (MOD) Community of Practice Workshop; Washington, D.C.

TriMet conducts weekly project meetings on the following rotating Slack channels every Thursday at 10am PST. This quarter, they occurred on the following days:

- Geocoder Meetings (https://trimet-mod-sandbox.slack.com/messages/geocoding/)
- Application Development Meetings (https://trimet-mod-sandbox.slack.com/messages/general/)

Upcoming Highlights

- TriMet is presenting *Integrating Transit with Shared-Use Mobility Options - MOD Sandbox Grant* at the Association for Commuter Transportation (ACT) conference, which will take place in New Orleans, LA July 30 - August 2, 2017, (http://www.actconf.org/full_schedule.cfm).
- TriMet has been selected to present *Solving the last mile problem with OpenTripPlanner (OTP), Mapzen Pelias, and open data* at the annual FOSS4G conference, which will take place in Boston, MA August 14 –19, 2017, (http://2017.foss4g.org/accepted-presentations/#government).

APPENDICES

Appendix A - Task 2 Geocoder Evaluation Quarter 2 Report

Appendix B - Task 3 Milestone 2 Documentation

Appendix C - Task 4 Milestone 4 Documentation

Appendix D - Additional Task 4 Milestone 4 Documentation

Appendix A - Task 2 Geocoder Evaluation Quarter 2 Report

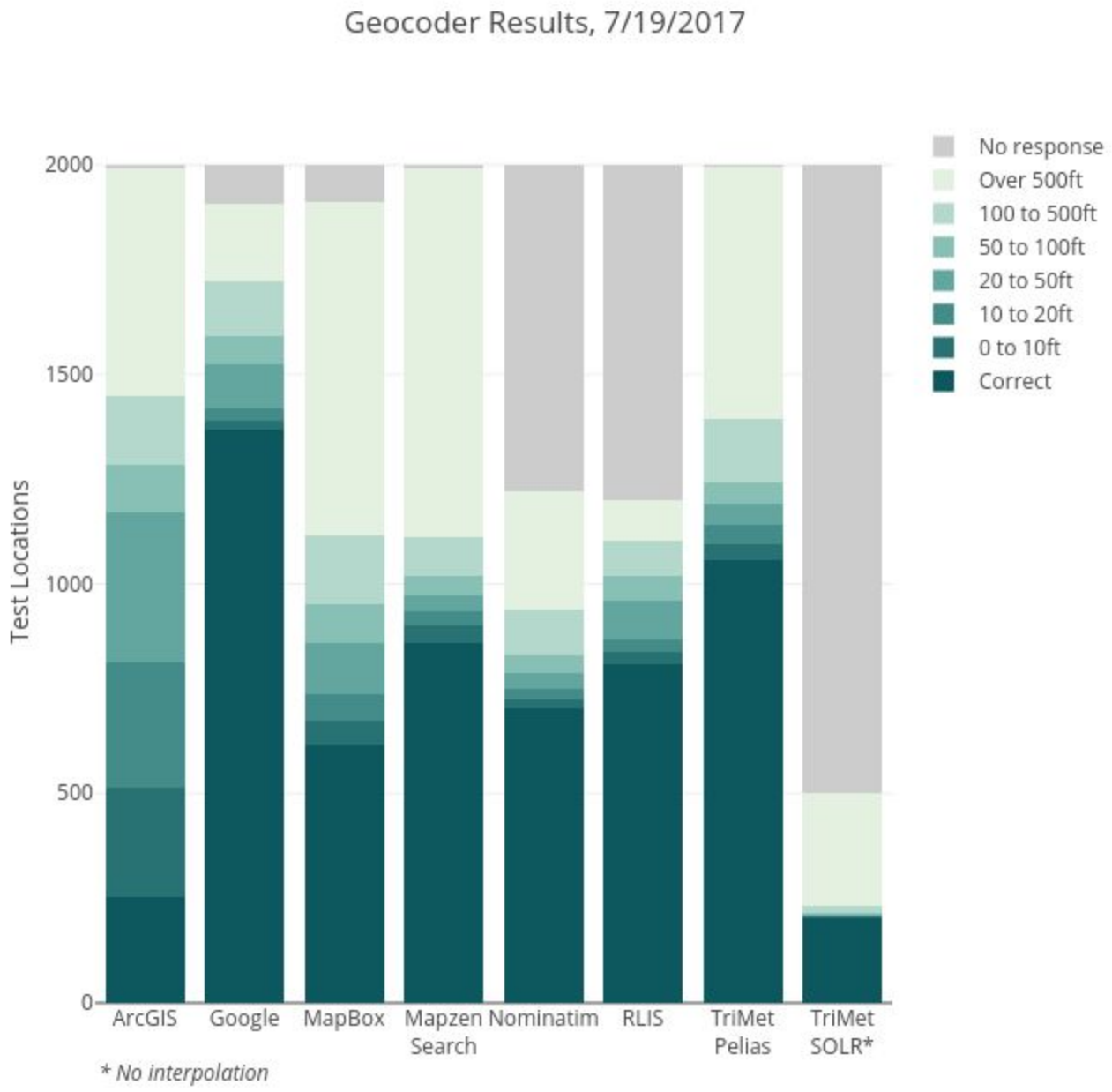
(4 pages total)

Part of the improvements to the OpenTripPlanner include an improved and open-source geocoder. This appendix describes the evaluation process involved in measuring and evaluating the geocoder progress and improvement.

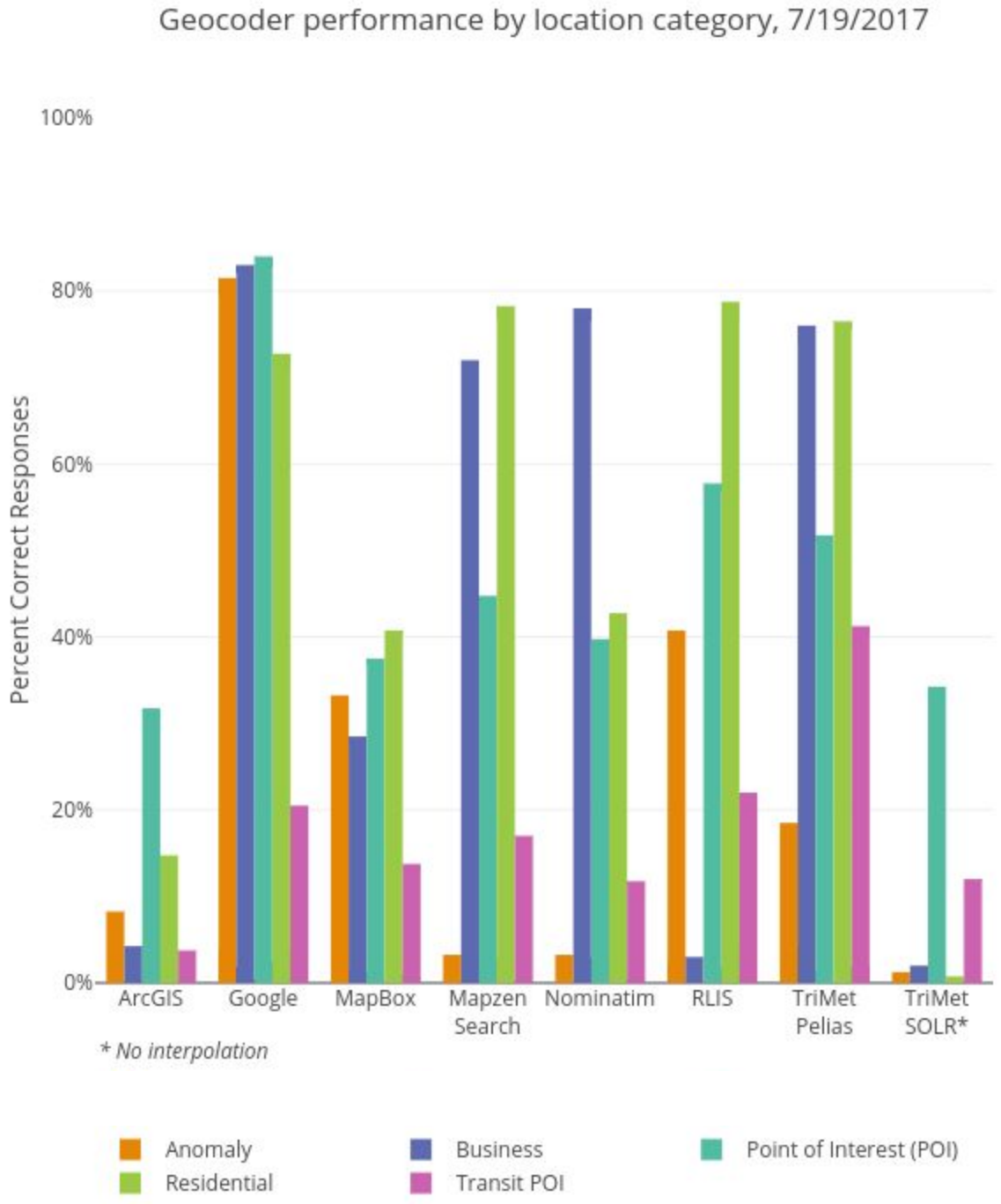
In order to compare the geocoders, TriMet developed a test suite of 2,000 refined and validated locations. These consisted of the following categories: Top User Submissions, Intersections, Commonly Misspelled Locations, Multifamily Residential, Landmarks, Theoretical Addresses, eFare Outlets, Leading Zero Addresses, Bus Stop IDs, Locations with Aliases, Venues, Proportional to Population, Misspelled Street, Misspelled City, Wrong Suffix and Transposed Street. More details about this test suite can be found here: https://docs.google.com/spreadsheets/d/1b0zxcb_5w0M6ydStkVlL9ceIAs5P_gJhdQNLfhd0pyA/edit?usp=sharing.

Based on our initial evaluation of the many geocoders available -- ArcGIS, Google, Mapbox, Mapzen, OpenStreetMap, Oregon Metro's and TriMet's SOLR -- we realized that a polygon-based evaluation method (over a point-based one) was necessary for the following reasons:

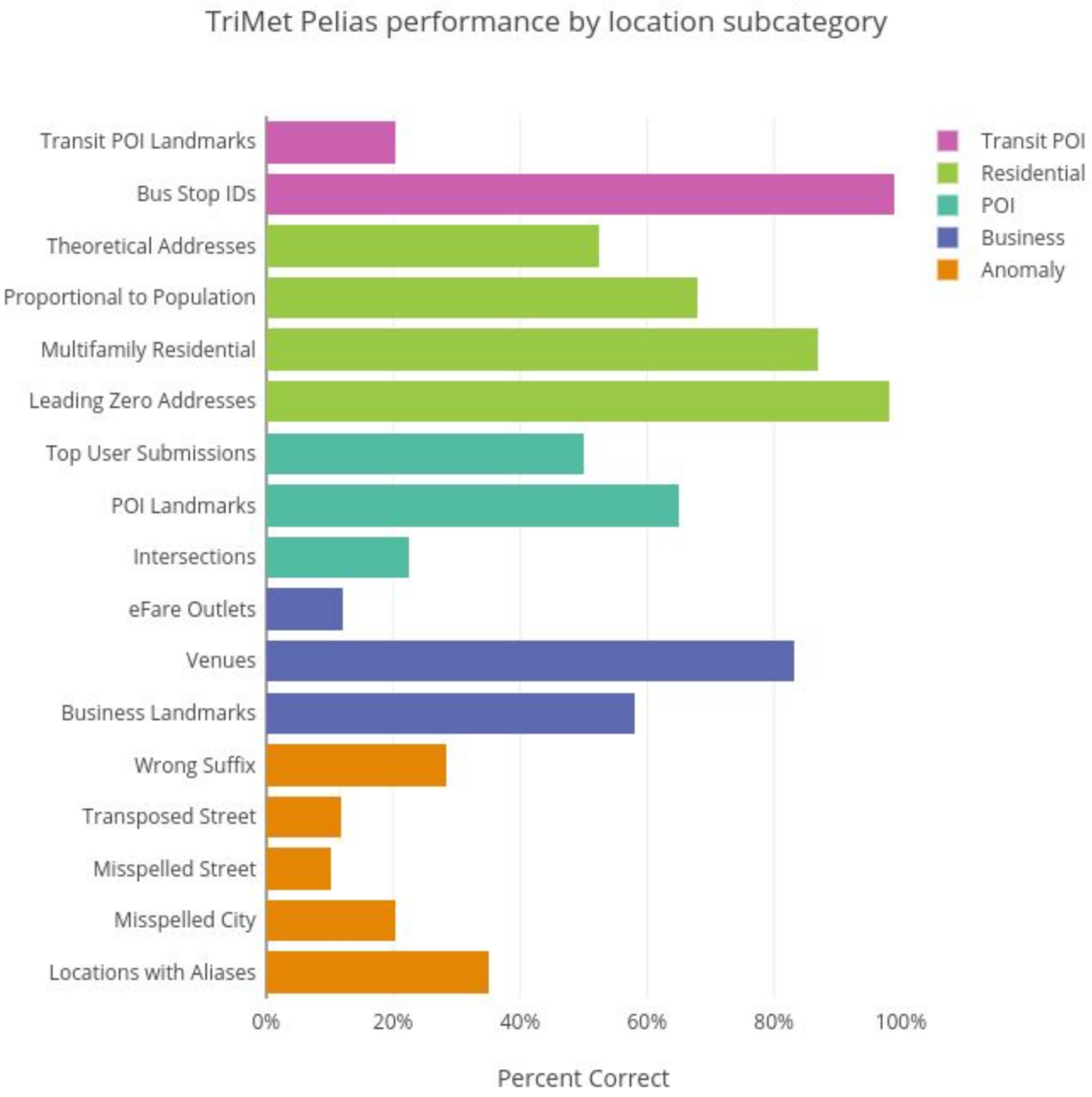
1. Each geocoder determines the geocoded location differently, e.g., middle of a building, front of a building, on the street in front of a building, etc. Choosing a single point as the "correct" location will invariably bias some geocoders over others;
2. Some locations are so large and complex, e.g., Portland International Airport, that a single point is inadequate.
3. Some locations that users search for are amorphous, e.g., intersections such as SW 3rd Avenue and SW Pine Street, or neighborhoods like "Chinatown" or "Downtown". Polygons better reflect these locations than points.
4. A polygon-based method allows us to better determine the accuracy of geocoder responses beyond just correct/incorrect. Responses are correct if they fall within the validated polygon, but incorrect responses are measured by their distance from the polygon.



The above chart illustrates the initial results of the polygon-based evaluation. What is most striking is the improvement that TriMet's instance of Pelias represents over the base instance of Pelias (Mapzen Search) and TriMet's current geocoder, SOLR (CAVEAT: these results do not fully reflect SOLR's performance because they do not incorporate SOLR's autocomplete function). We expect TriMet Pelias to further improve in performance as we add in the ability to define location aliases, e.g., TV Highway instead of Tualatin Valley Highway.



It is great to see TriMet Pelias outperform TriMet SOLR in every category. In addition, TriMet Pelias performs better than every other geocoder in correctly identifying Transit Points of Interest (POI). We are examining the results to determine how we can further fine tune TriMet Pelias to improve performance.



We are examining the results illustrated in this chart to determine how we can further fine tune TriMet Pelias to improve performance.

Appendix B - Task 3 Milestone 2 Documentation

(4 pages total)

opentripplanner / otp-react-redux

Commits on May 24, 2017

Merge pull request #34 from opentripplanner/bikeshare-overlay ... landonreed committed on GitHub on May 24 ✓	488435b	<>
fix(BikeRentalOverlay): add missing key to iterator landonreed committed on May 24 ✓	0e5cdfe	<>
fix(example): include BikeRentalOverlay in example landonreed committed on May 24	29a3a8e	<>
Merge branch 'dev' into bikeshare-overlay landonreed committed on GitHub on May 24 ✓	2e13641	<>
style(bike-rental-overlay): simplify mapDispatchToProps and destructu... ... landonreed committed on May 24 ✓	2ebc2ab	<>
style(fix lint): landonreed committed on May 24 ✓	1f84907	<>
Merge pull request #32 from opentripplanner/settings-screen ... landonreed committed on GitHub on May 24 ✓	2eba142	<>
refactor(remove inline func, simplify mapDispatchToProps): landonreed committed on May 24 ✓	b6d62d8	<>
fix(modes-panel): refactor mode button into separate component landonreed committed on May 24	551cf75	<>

Commits on May 10, 2017

fix(example): Fix bundled example demory committed on May 10 ✓	0f4af4c	<>
style(form): Make linter happy demory committed on May 10 ✓	3909593	<>

Commits on Apr 26, 2017

feat(map): Only show bike-rental overlay when bike-rental mode is active demory committed on Apr 26 ✗	e9282f7	<>
feat(map): Add svg icons for bikeshare overlay demory committed on Apr 26	60ad3c4	<>
feat(form): Allow optional expansion caret in settings-bar demory committed on Apr 26 ✗	f8fcaf0	<>
refactor(form): Remove css-based mode icons demory committed on Apr 26	abf082e	<>
feat(form): Support custom mode icons demory committed on Apr 26	ec1181b	<>
fix(form): Set keys for settings-bar mode icon list demory committed on Apr 26	380c3d1	<>


feat(api): Export setShowExtendedSettings action via API
demory committed on Apr 26
18bf8ed


feat(form): Add interactivity to settings/modes panel ...
demory committed on Apr 26
f054d89

Commits on Apr 24, 2017


Merge branch 'dev' into settings-screen
demory committed on Apr 24
cb52f54


feat(form): Initial work on detailed, mobile-ready settings panel
demory committed on Apr 24
b7f0928


Merge pull request #30 from opentripplanner/itinerary-refactor ...
demory committed on GitHub on Apr 24 ✗
6841ce2


feat(narrative): Add mode icons to CSS
demory committed on Apr 24 ✓
c1fe5bd

Commits on Apr 21, 2017


feat(narrative): Allow custom itinerary renderers in itinerary-carousel
demory committed on Apr 21 ✗
3d3bfb4

Commits on Apr 19, 2017


feat(api): Export selected utility libraries via API ...
demory committed on Apr 19 ✗
446d01b


refactor(narrative): Refactor itinerary narrative rendering to allow ... ...
demory committed on Apr 19
1929b5e

Commits on Apr 17, 2017


fix(form): Fix boolean/bool typo in mode-selector propTypes
demory committed on Apr 17 ✓
3413bc7


feat(overlay): Refine/enhance bike station overlay ...
demory committed on Apr 17 ✗
83ebd5c


feat(overlay): Initial work on map overlay
demory committed on Apr 13
00e7230


Merge pull request #27 from opentripplanner/mobile-support ...
demory committed on GitHub on Apr 17 ✗
3a4f0d0


refactor(form): Refactor props definition in mode-selector ...
demory committed on Apr 17 ✗
c7e9a8e


Merge branch 'dev' into mobile-support
demory committed on Apr 17 ✓
118226b


refactor(api): Move autoPlan to config
demory committed on Apr 17
f5abca1

Commits on Apr 13, 2017


style(form): Fix lint errors in DateTimeSelector
demory committed on Apr 13 ✗
8f44e46


fix(api): Add action for setAutoPlan ...
demory committed on Apr 13 ✗
8a9f98f

opentripplanner / otp-react-redux

Commits on Apr 13, 2017


feat(narrative): Add option for expansion listener to ItineraryCarousel
demory committed on Apr 13
b787b53


feat(form): Enhancements to PlanTripButton for TriMet mobile UI ...
demory committed on Apr 13
595cc08

Commits on Apr 12, 2017


feat(form): Enhancements to ModeSelector for TriMet mobile UI ...
demory committed on Apr 12
7f17239


refactor(form): Restructure DateTimeSelector ...
demory committed on Apr 12
ec006ea


feat(api): Add autoPlan setting ...
demory committed on Apr 12
e6786f7

Commits on Apr 5, 2017


Merge pull request #24 from opentripplanner/form-control ...
evansiroky committed on GitHub on Apr 5 ✓
7049707


docs(form): add code comments to add clarity
evansiroky committed on Apr 5 ✓
fef275f


refactor(form): update debounce upon state change
evansiroky committed on Apr 5 ✓
2dfb1dd

Commits on Apr 4, 2017


feat(form): add config options for form control ...
evansiroky committed on Apr 4 ✓
fded353

Commits on Mar 28, 2017


Merge pull request #22 from opentripplanner/master ...
evansiroky committed on GitHub on Mar 28 ✓
c5fbfce


Merge pull request #18 from opentripplanner/css-fix ...
evansiroky committed on GitHub on Mar 28 ✓
c655533


refactor(css): move some css around
evansiroky committed on Mar 28 ✓
58d7c71


Merge pull request #20 from opentripplanner/graceful-server-error-han... ...
evansiroky committed on GitHub on Mar 28 ✓
c8f9877


refactor(api): use async/await with fetch ...
evansiroky committed on Mar 28 ✓
db2c4d3


test(api): fix test snapshot
evansiroky committed on Mar 28 ✓
169453f

feat(error-handling): display error-message upon server fails ...
evansiroky committed on Mar 28 ✗ — 1653bdb

Merge pull request #19 from opentripplanner/itin-carousel ...
landonreed committed on GitHub on Mar 28 ✗ — b0370f0

refactor(NarrativeItinerary): make this.props.onClick override defaul... ...
landonreed committed on Mar 28 ✓ — eee8e80

refactor(ItineraryCarousel): use more concise code
landonreed committed on Mar 28 — 30d8de3

Commits on Mar 27, 2017

fix(api): gracefully handle bad server response ...
evansiroky committed on Mar 27 ✓ — e28e5a3

refactor(ItineraryCarousel): add hideHeader prop to itinerary carousel
landonreed committed on Mar 27 ✗ — e4125cd

chore(example): re-comment out initialQuery
landonreed committed on Mar 27 — f631d60

build(yarn): add prestart
landonreed committed on Mar 27 ✓ — 22aec13

build(deps): add react-swipeable-views
landonreed committed on Mar 27 — c7d8811

feat(ItineraryCarousel): add itinerary carousel component to narrative
landonreed committed on Mar 27 — ecc3974

Commits on Mar 24, 2017

fix(css): bundle all css together
evansiroky committed on Mar 24 ✓ — 46f98eb

Commits on Mar 23, 2017

Merge pull request #17 from opentripplanner/dev ...
evansiroky committed on GitHub on Mar 23 ✓ — 85c5f4f

Merge pull request #16 from opentripplanner/npm-fix ...
evansiroky committed on GitHub on Mar 23 ✓ — 8bdc7bf

fix(npm): include built css file in npm package
evansiroky committed on Mar 23 ✓ — 882f523

refactor(npm): include dist folder when publishing
evansiroky committed on Mar 23 ✓ — 50df95e

fix(npm): add .npmignore file
evansiroky committed on Mar 23 ✓ — 28fb359

Merge pull request #15 from opentripplanner/dev ...
evansiroky committed on GitHub on Mar 23 ✗ — 07f2631

Merge pull request #14 from opentripplanner/switching ...
evansiroky committed on GitHub on Mar 23 ✗ — 53393d9

refactor(example): rename and alphabetize some things ...
evansiroky committed on Mar 23 ✗ — 136ed60

Commits on Mar 22, 2017

feat(actions): add ability to switch locations
evansiroky committed on Mar 22 ✓ — 967a995

Appendix C - Task 4 Milestone 4 Documentation

(2 pages total)

Mapzen Milestones

Mobility on Demand Grant

Local Installation Packages

Status

Completed (100%)

Background

See the original statement of work document for context.
Additionally, see TriMet's analysis of the problem.

Overview

Implement a simple setup system for agencies wanting to install a local instance of the search engine using either all or a subset of the OpenAddresses/OpenStreetMap/Who's on First data. This can allow for easy testing of the specified data sources. It would also provide a solution for those needing higher rate limits than the public Mapzen Search API can support. Must at minimum support the operating systems identified as critical by user research: Windows/Ubuntu/MacOS.

Deliverables

Source Data Filtering for Importers

Add support for new configuration options that allow for setting up Pelias using data for a limited region. Each data source will have its own implementation for filtering the source data. Each corresponding importer will have a new `download` script to be used for downloading either the full or filtered dataset.

1. OpenStreetMap will use PBF extracts of the data, from Metro-Extracts or Geofabric
2. OpenAddresses will allow specifying a list of source files to be used
3. Who's on First will allow specifying an ID of the region of interest, such as the city or state where coverage is desired

Manual (Unpackaged) Setup

1. Improved documentation and configuration for a step-by-step traditional installation and setup process of the Pelias geocoder
    - See here

Containerized Setup

1. Docker-compose setup for orchestrating all the individual containers appropriately, along with relevant documentation and configuration examples
    - See here

2. Documentation outlining the installation process using containers on a personal computer as well as recommendations for a hosted setup on AWS
    - See here

3. Docker containers for each component of the target system with relevant documentation
   Note: The relevant Dockerfile(s) will exist under each repository for ease of setup independently of the rest of the system
    - *Interpolation*
    - *PIP*
    - *Placeholder*
    - *API*

Logistics

Task	Weeks (40 hours)	Cost
Local installation package	7	$28,000
Total	7	$28,000

Appendix D - Additional Task 4 Milestone 4 Documentation

(6 pages total)

pelias / dockerfiles

Containerized Local Installation Package for the Pelias geocoder http://pelias.io

52 commits 5 branches 0 releases 4 contributors

dianashk committed on GitHub Merge pull request #10 from 3vivekb/patch-1 ... Latest commit 1e68574 3 days ago

api	add libpostal baseimage and fix interpolation script	14 days ago
baseimage	revert pelias.json	14 days ago
elasticsearch	move all the dockerfiles back to a flat structure	15 days ago
geonames	fix typo	15 days ago
interpolation	add libpostal baseimage and fix interpolation script	14 days ago
libpostal_baseimage	add libpostal baseimage and fix interpolation script	14 days ago
openaddresses	remove CMD from non-long-running containers	14 days ago
openstreetmap	new lines	14 days ago
pip	move all the dockerfiles back to a flat structure	15 days ago
placeholder	refactor placeholder container to match changes in repo	14 days ago
polylines	new lines	14 days ago
schema	remove CMD from non-long-running containers	14 days ago
valhalla	remove CMD from non-long-running containers	14 days ago
whosonfirst	remove CMD from non-long-running containers	14 days ago
.env	env: set default env vars for safety	3 months ago
build.sh	remove no-cache switch	15 days ago
docker-compose.yml	add libpostal baseimage and fix interpolation script	14 days ago
example.sh	add valhalla	3 months ago
pelias.json	add ability to specify git revision for each component	15 days ago
prep_data.sh	remove CMD from non-long-running containers	14 days ago
readme.md	Typo	3 days ago
run_services.sh	move all the dockerfiles back to a flat structure	15 days ago

readme.md

Dockerfiles for Pelias services

Prerequisites

1. Docker version `1.10.1` or later.
2. A directory for storing downloaded datasets. Set `DATA_DIR` to the folder's path in `.env` file.

3. **OSX Only**

    i. In Docker > Preferences > Advanced, set the CPU to `4` and memory to `12 GB`. This ensures that Docker has enough memory to run the imports and API.

Create a Directory for Your Data

Each of the containers will be able to access this directory internally as `/data`, source data downloaded by the containers will be stored here.

> note: the data can be fairly large, make sure you have at minimum ~15GB free space available on this volume

```
mkdir -p /tmp/data
```

If you wish to change the location of your data directory you can simply change the `DATA_DIR` environment variable.

Each importer and service has a range of different options, detailed installation and configuration instructions can be found here: https://github.com/pelias/pelias/blob/master/INSTALL.md For an up-to-date references of supported options you can also view the README files contained in each repository on Github.

Getting Up and Running

First you'll need to create (or edit) the provided `pelias.json` file at the root of the repository. This is where you will specify all the details of your desired Pelias instance, such as area of coverage and data sources. You can reference the individual data sections below for more details on configuration.

Once that's ready, the following command will build all the images and containers required:

> NOTE: this command can take several hours depending on your network, hardware, and the size of the region of coverage selected in pelias.json.

```
./build.sh
```

once the process is complete you can list the running services:

```
$ docker-compose ps
        Name                   Command           State                Ports
-------------------------------------------------------------------------------------------
pelias_api             npm start                Up       0.0.0.0:4000->4000/tcp
pelias_baseimage       /bin/bash                Exit 0
pelias_elasticsearch   /bin/bash bin/es-docker  Up       0.0.0.0:9200->9200/tcp, 9300/tcp
pelias_geonames        /bin/bash                Exit 0
pelias_interpolation   npm start                Up       0.0.0.0:4300->4300/tcp
pelias_openaddresses   /bin/bash                Exit 0
pelias_openstreetmap   /bin/bash                Exit 0
pelias_pip             npm start                Up       0.0.0.0:4200->4200/tcp
pelias_placeholder     npm start                Up       0.0.0.0:4100->4100/tcp
pelias_polylines       /bin/bash                Exit 0
pelias_schema          /bin/bash                Exit 0
pelias_whosonfirst     /bin/bash                Exit 0
```

Checking that Services are Running

All the services should be up and running after the build script completes. The ports on which the services run should match the configuration in `docker-compose.yml`. You can confirm this worked correctly by visiting each one at the corresponding URLs.

API

http://localhost:4000/v1/search?text=portland http://localhost:4000/v1/search?text=1901 Main St
http://localhost:4000/v1/reverse?point.lon=-122.650095&point.lat=45.533467

Placeholder

http://localhost:4100/demo/#eng

PIP (point in polygon)

http://localhost:4200/-122.650095/45.533467

Interpolation

http://localhost:4300/demo/#13/45.5465/-122.6351

Data Download and Import

There is a script that is actually used in the `build.sh` script but can also be executed independently to update the data and rebuild the ES index and other databases.

Note: if you are going to run it independently, it's important to make sure the docker containers have already been built. This script will also shut down any running services to avoid conflicts during imports.

It is ***VERY VERY*** strongly recommended that you use the `pelias.json` config file to limit the data downloads to a region no larger than a region (state in US). There is too much data in larger regions for a single machine to handle. Also keep in mind that the amount of time a download and import will take is directly correlated with the size of the area of coverage.

For TIGER data, use `imports.interpolation.download.tiger[]` (see interpolation repo doc)

```
mdkir -p /tmp/data
export DATA_DIR=/tmp/data
sh ./prep_data.sh
```

Individual Data Sources

Who's on First

note: this guide only covers importing the admin areas (like cities, countries etc.)

configuration

For WOF data, use `imports.whosonfirst.importPlace` (see whosonfirst repo doc)

```
"imports": {
  "whosonfirst": {
    "datapath": "/data/whosonfirst",
    "importVenues": false,
    "importPostalcodes": true,
    "importPlace": "101715829",
    "api_key": "your-mapzen-api-key"
  }
}
```

download

```
docker-compose run --rm whosonfirst npm run download
```

import

```
docker-compose run --rm whosonfirst bash -c 'npm start'
```

OpenAddresses

configuration

For OA data, use `imports.openaddresses.files` (see openaddresses repo doc)

```
"imports": {
  "openaddresses": {
    "datapath": "/data/openaddresses",
    "files": [ "us/or/portland_metro.csv" ]
  }
}
```

download

```
docker-compose run --rm openaddresses npm run download
```

import

```
docker-compose run --rm openaddresses npm start
```

OpenStreetMap

Any `osm.pbf` file will work. A good source is Metro Extracts, which has major cities and custom areas. Download and place the file in the data directory above.

configuration

Once you find a URL from which you can consistently download the data, specify it in the configuration file and the download script will pull it down for you.

For OSM data, use `imports.openstreetmap.download[]` (see openstreetmap repo doc)

```
"imports": {
  "openstreetmap": {
    "download": [
      {
        "sourceURL": "https://s3.amazonaws.com/metro-extracts.mapzen.com/portland_oregon.osm.pbf"
      }
    ],
    ...
  }
}
```

download

Using the download script in the container:

```
docker-compose run --rm openstreetmap npm run download
```

Or, download the data by other means such as `wget` (example for Singapore):

```
wget -qO- https://s3.amazonaws.com/metro-extracts.mapzen.com/singapore.osm.pbf > /tmp/data/openstreetmap/extract.osm.
```

import

```
docker-compose run --rm openstreetmap npm start
```

Geonames

configuration

You can restrict the downloader to a single country by adding a `countryCode` property in your `pelias.json`:

```
"imports": {
  "geonames": {
    ...
    "countryCode": "SG"
  }
}
```

download

```
docker-compose run --rm geonames npm run download
```

import

```
docker-compose run --rm geonames npm start
```

Polylines

configuration

```
"imports": {
  "polyline": {
    "datapath": "/data/polylines",
    "files": ["pbf_extract.polyline"]
  }
}
```

download

The extract of the polylines is done using the OSM pbf file so that must be downloaded first. See OpenStreetMap section for details on that. Once the pbf extract is in place, run the following command.

```
docker-compose run --rm polylines sh ./docker_extract.sh
```

import

```
docker-compose run --rm polylines npm run start
```

Setting Up Elasticsearch

This will take place as part of the build script, but in the case you'd like to manually manipulate the schema, the following command will install the pelias schema in elasticsearch:

```
docker-compose run --rm schema bash -c 'node scripts/create_index.js'
```

You can confirm this worked correctly by visiting http://localhost:9200/pelias/_mapping

Shutting Down and Restarting

To stop all the containers, `docker-compose down`.

Restart all the containers with `docker-compose up` or `sh ./run_services.sh`.