GitHub Username: tundeaina

popEstimator

Description

Often times a quick and reliable estimate of the demographic composition of an area around a given location is desired. The location could be a school, a store, an apartment or even the site of an emergency.

popEstimator makes use of the location aware capability of mobile devices, traffic information and mapping services coupled with demographic data to compute a good estimate of population. The user requests these estimates by specifying a location or use current location and then defines the type and reach of the desired Area of Interest (**AOI**). The app responds by displaying the location as a pushpin and the catchment area as a polygon on a map, and presents the computed data as a bar chart with some descriptive text.

Intended User

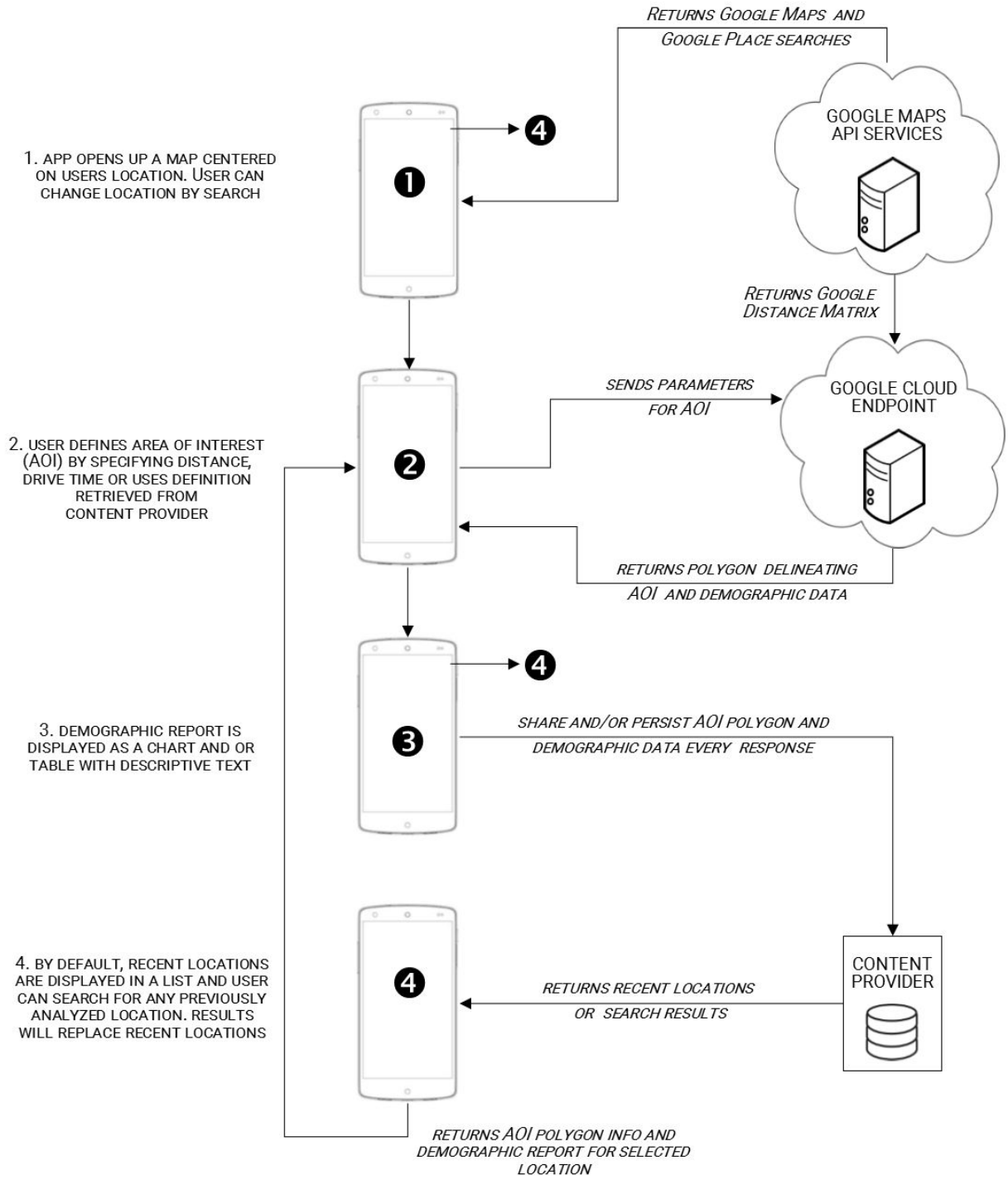
This is a general purpose informational app for anyone interested in knowing the counts of people with any given area anywhere in the United States. This will be particularly useful for retailers and marketers, emergency first responders, urban planners and real estate agents, amongst others.

Features

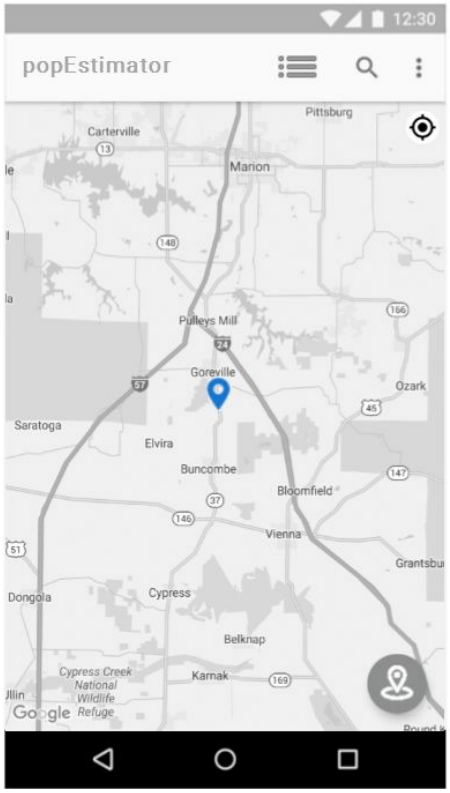
popEstimator is capable of the following:

- Sensing the location of the device
- Creating a catchment area defined by
    - a location and a radial distance
    - a location and a drive time from the target location
- Displaying the AOI and location on a map
- Computing an estimate of the demographic make up within the AOI
- Displaying the results in a Bar or Pie chart along with a short descriptive text
- Persisting input and results for all estimates requested to be saved
- Searching and retrieval of estimates for locations saved on device
- Sharing the estimates with other apps

Overview and Data Flow



User Interface Mocks

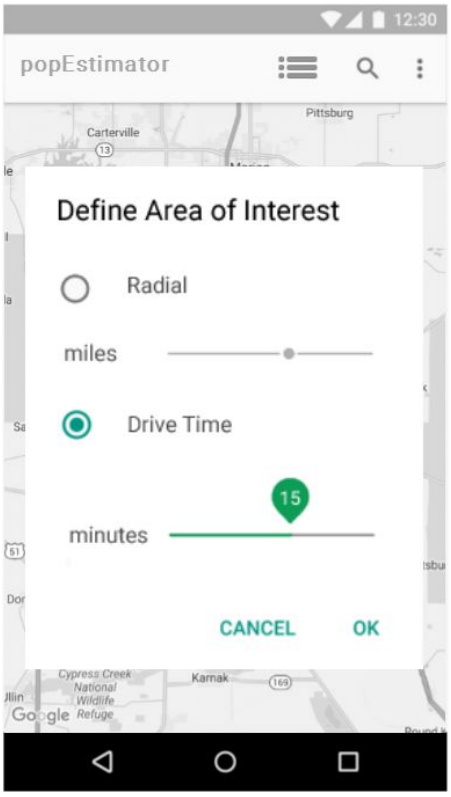


Map Fragment

This initial fragment shows a map centred on the user's current location. The user has the option of searching for a new location by requesting a Google Place Search or can go to a previously visited located store on the device.

Once the user has settled at a location, The AOI FAB is used to bring up the AOI definition dialog.

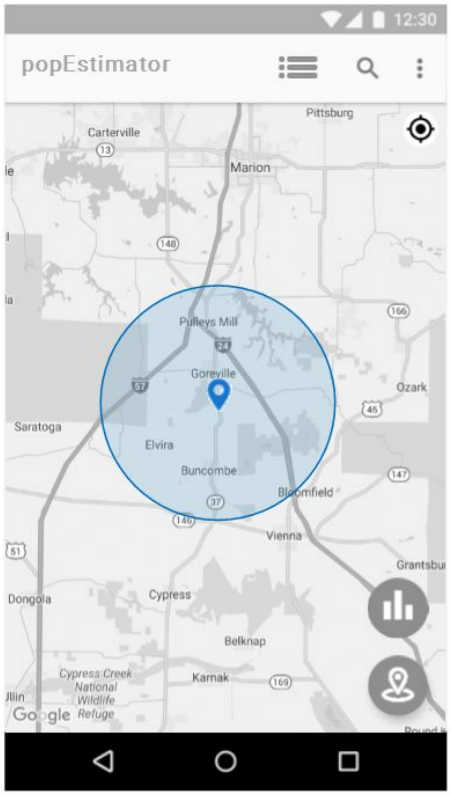
User can go back to current location by pressing on the My Location icon at the top right area of the map.
Please see Phone UI/Tablet UI flow below user interactions



Map Fragment with AOI Dialog

The user is presented with a dialog to collect the Area of Interest (AOI) definition. The user can specify a radial AOI based on distance in miles. A slider is used to select from some preset values.

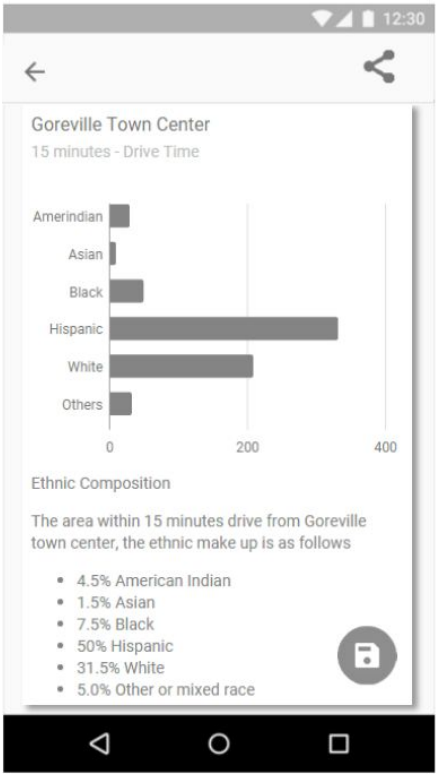
The other option is to define the AOI by selecting a drive time in minutes. This will create a zone encompassing the area the user can reach in the specified amount of time. The sliders are active only when the associated Radio button is clicked.



Map Fragment with AOI Polygon

On submitting the AOI definition, the location and AOI information is sent to a Google Cloud Endpoint which also make a call to the Google Distance Matrix API. The endpoint uses the results of the distance matrix to generate a polygon definition the AOI and uses this to compute an estimate of the population.

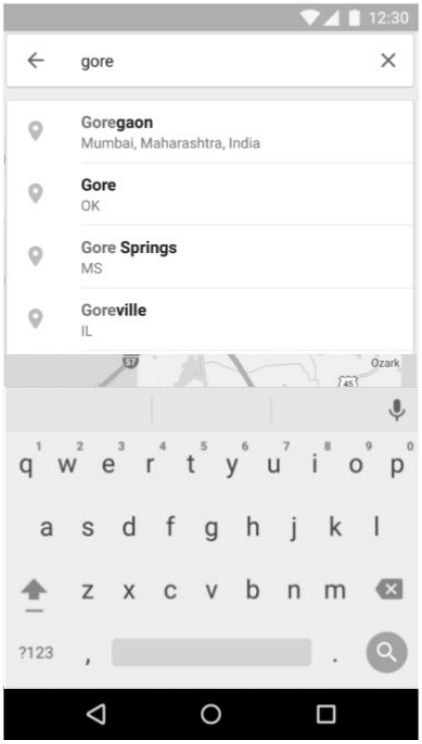
The endpoint responds by sending to the device the polygon and population counts. The polygon is then displayed on the maps. The user can now click on the Report FAB to transition to the report activity.



Report Fragment

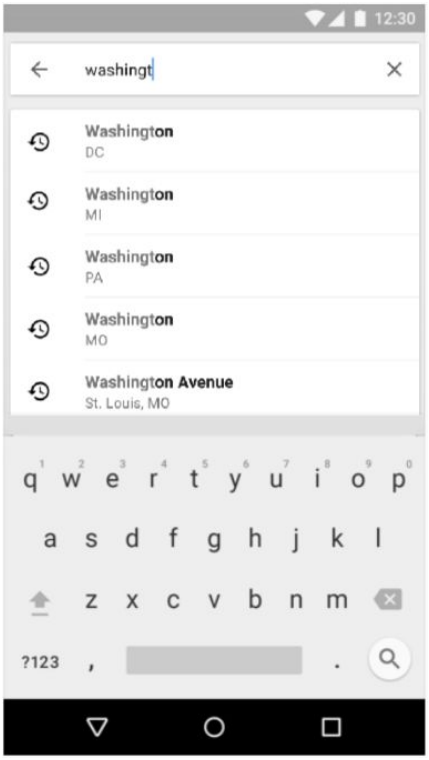
Apart from the visual display of the AOI on a map, the user can see the underlying data for the AOI. This fragment is a webview that displays a chart using the Google Visualization API. The Chart is interactive. It can reveals more information when user interacts with it.

A descriptive text is also included. This the info that will appear in the Widget.



New Search Fragment

In this activity the user can search for a new place or location by specifying an address or place name. This search activity also provides auto completion. Upon accepting a location, the user is returned to the Map fragment with a push pin at the location.

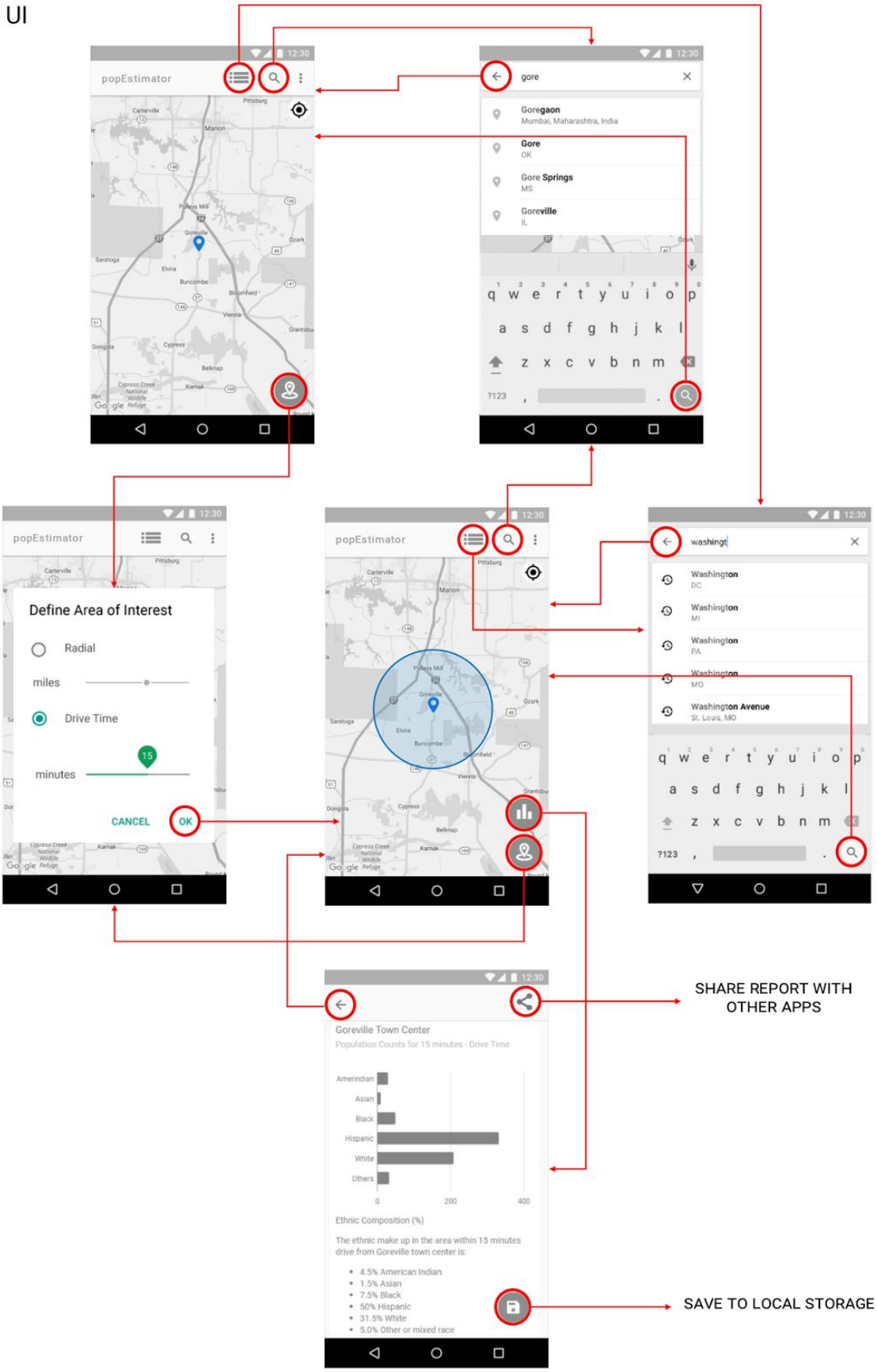


History Search Fragment

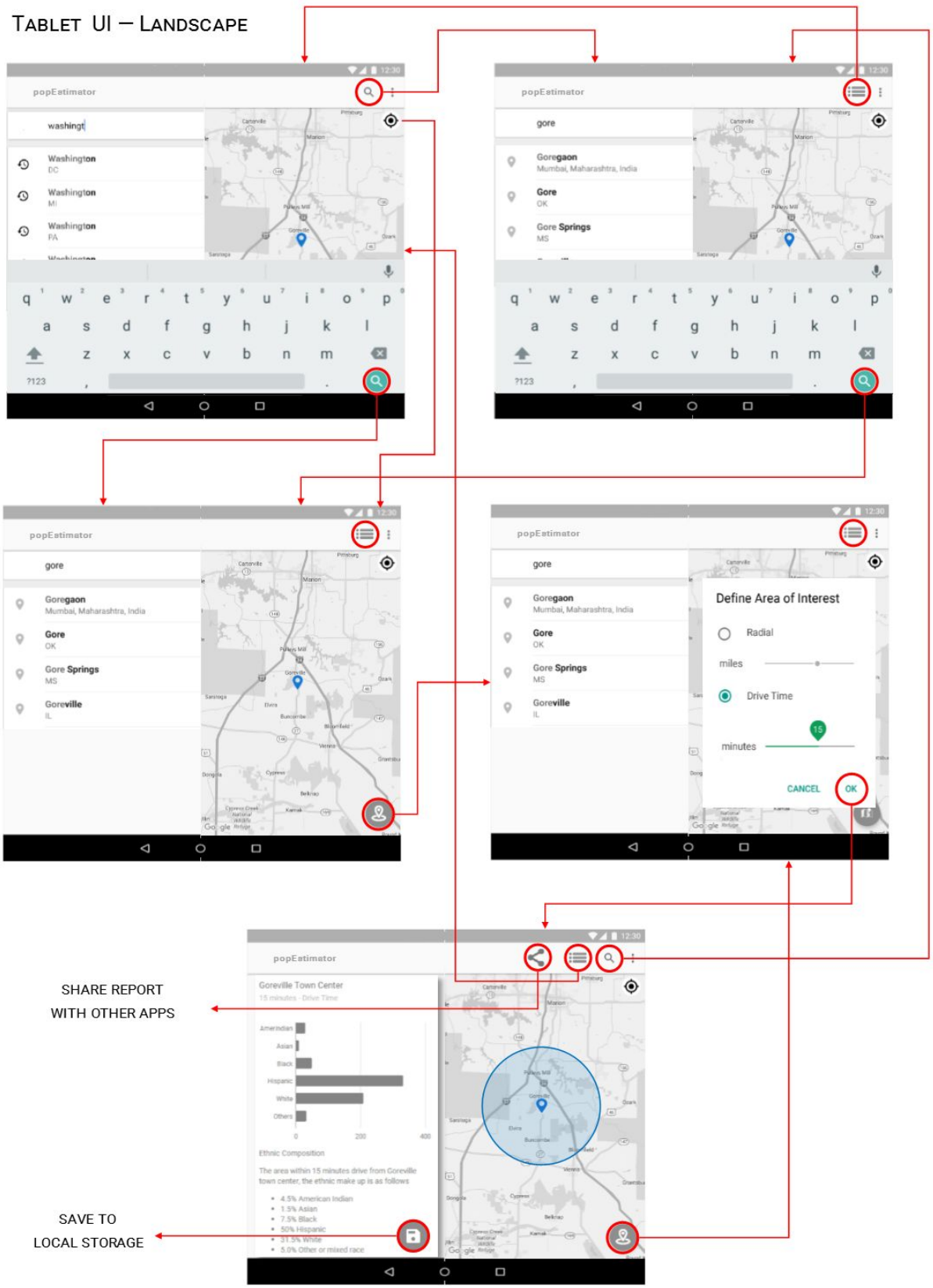
In this activity the user can search for a saved location. This search activity provides auto completion based on the data retrieved from the content provider. Upon accepting a location, all the saved information for the location is retrieved and passed to the Map and Report fragments.

The user is able to see the AOI polygon and the report is also ready for viewing.

PHONE UI


popEstimator
gore
Goregaon
Mumbai, Maharashtra, India
Gore
OK
Gore Springs
MS
Goreville
IL
Define Area of Interest
Radial
miles
Drive Time
minutes
15
CANCEL
OK
washingt
Washington
DC
Washington
MI
Washington
PA
Washington
MO
Washington Avenue
St. Louis, MO
SHARE REPORT WITH OTHER APPS
Goreville Town Center
Population Counts for 15 minutes - Drive Time
Amerindian
Asian
Black
Hispanic
White
Others
0
200
400
Ethnic Composition (%)
The ethnic make up in the area within 15 minutes drive from Goreville town center is:
4.5% American Indian
1.5% Asian
7.5% Black
50% Hispanic
31.5% White
5.0% Other or mixed race
SAVE TO LOCAL STORAGE

Tablet UI – Landscape



Key Considerations

How will your app handle data persistence?

A Content Provider will be created to store information for processed locations such as

- Date and Time
- Address
- Latitude and Longitude
- AOI definition
- Processed AOI polygon vertices
- Population estimates

The History search and Widget fragments will be bound to this content provider

Describe any corner cases in the UX.

None so far.

Describe any libraries you'll be using and share your reasoning for including them.

The app will call on a java library - **estimator**. This library creates the AOI polygon and computes the population estimate. **estimator** makes use **conrec** to interpolate the contours of travel times from a rectangular grid. The app will also use an open source library - **proj4j** to handle all map and geographic projections.

The estimator will reside in a Google Cloud Endpoint and will be accessed by the app asynchronously.

The following APIs will be used by the app

1. Google Places Search API
2. Google Distance Matrix API
3. Google Play Services - Location API
4. Google Play Services - Maps API
5. Google Visualization API

Next Steps: Required Tasks

Task 1: Project Setup

- Configure libraries
    - estimator
    - proj4j
    - conrec

Task 2: Create Java libraries - estimator; proj4j; conrec

- Create estimator
    - Implements classes
        - areaOfInterest
        - distanceElementMatrix
        - interval
        - interval2D
        - latLonUtils
        - pointInPolygon
        - pointUtils
        - quadTree
    - Unit test all methods
- Create proj4j
    - import source code
    - build .jar
    - unit test relevant methods

Task 3: Create Backend - GCE module

- set up a GCE development server
- introduce a project dependency between Java library- estimator and the GCE module
- modify the GCE starter code
- Create an AsyncTask to access access remote endpoints
- Create Tests to ensure stability and successful retrieval of data

Task 4: Implement UI for Each Activity and Fragments

- Build UI for Phone
    - Main Activity
        - Setup Map fragment
    - New Search Activity
        - Implement expandable search
    - History Search Activity
        - Implement persistent search
    - Report Activity
        - Set up WebView fragment
        - Create class to compose *javascript* or report display
- Build UI for Tablet Landscape
    - MainActivity
        - Setup Map fragment
        - Implement persistent search for New Search fragment
        - Implement persistent search for Historic Search fragment
- Build AOI definition dialog

Task 5: Create Content Provider

- Design data storage
- Design Contents URIs
- Create Database Helper class
- Implement the Contract class
- Implement the Content provider
    - Implement all required methods
- Test insert, delete and query of data in Content Provider

Task 6: Wire up all UI elements

- New Search
- History Search
- AOI FAB
- Report FAB
- MyLocation button
- OK/Cancel buttons on Define AOI dialog
- Save FAB
- Share button

Task 7: Integration Testing and bug fixes

- Test all functionalities
- Fix any bugs found

Task 8: Build, Clean and Sign app

- Create Gradle tasks to build and clean the install Release flavour
- Digitally sign the APK.
