Jeff Tran - jt982
Ryan Prater - rp22566
Eric Thompson - ebt297

Routy

How to Use the App

Routy is a route finding app that helps you find the optimal route between an origin and a set of 1 to 5 destinations.

The Basics:

1. Click on the custom icon to launch Routy.
2. Marvel at the splash screen and loading bar!
3. You'll be presented with the origin enter screen. Here, enter in the address of the origin location (where you'll be starting your trip you'd like Routy to route) or hit "Find me" to try to have Routy find your location through GPS or the network (give it a moment while it finds you!).
4. Once you've set the origin, click done to proceed to the next screen where you enter in the destinations for your trip. Enter them in any order and click "Add another destination" to add dynamically more destination boxes. If you want to remove one, click "Remove" to dynamically remove it. For our example, you can click on "Test Default Destinations" to get 3 preset destinations for your route (this will be removed for the beta - it's just there to help you out). **Remember:** all addresses need to be complete addresses; you can't just type "H-E-B" - Routy needs the explicit addresses.
5. Once you've selected your destinations, click "Route it!" to proceed to the results screen. Here you can see the optimal order of your destinations in your route. You can also see the distance of your total trip, ending at the last destination, on the right side.
6. You can click on a "Click to view segment" button between two points to be taken into the native Google Maps app on your phone to see the route between the two destinations (or origin and first destination). To return to the app, just click the standard android "back" button **twice**.
7. If you would like to restart, change your destinations, or change origin, you can simply press the standard android "back" button.

Error Checking / Validation:

Jeff Tran - jt982
Ryan Prater - rp22566
Eric Thompson - ebt297

1. We've built in some common error checking for network/wifi/gps. Try turning these off and we'll help you figure out what's going wrong.
    a. On load, we check for a connection to some network (since Routy requires it to run).
    b. When you click "Find me" at the origin screen, we check to see that your gps or network is enabled so we can locate you.
2. We've built in address validation for all addresses entered. At either the origin or destination screen, try typing in a bad address (or multiple addresses on the destination screen) and we'll let you know which ones we had a problem with by highlighting them in red.
    a. For addresses that work, we provide you with a toast notification to let you know they worked!

Jeff Tran - jt982
Ryan Prater - rp22566
Eric Thompson - ebt297

List of Completed Features

1. Pull down and calculate distance differences for optimal route designation
2. Allows selection of up to 5 destinations
3. Allow GPS to find location of phone for origin determination
4. Show error popup when addresses are wrong, can't geo-locate
5. Show route overview after calculation
6. Links to Google Maps for individual route directions
7. Splash Logo (any kind of place-holder)

List of Features from Assignment 3 *Not* Completed

All features we specified for the alpha are completed (and then some - see below)!

List of Features Added *Not* a Part of Assignment 3

8. Make, add, and remove destinations be dynamic
9. Mascot Logo

List of Code from Other Sources

None

Jeff Tran - jt982
Ryan Prater - rp22566
Eric Thompson - ebt297

List of Classes and Major Code “Chunks”

- **Routy** (bread and butter)**:**
  - **Destination Activity -** Creates the Destination Enter screen, takes input for destinations, add/remove destinations to your route, validates addresses, and calls the CalculateRouteTask to generate the route itself.
  - **Main Activity -** Opens when the icon is clicked for our app; it current preloads the ConnectivityManager and looks for available networks. We currently sleep for 2 seconds (this will eventually include a splash screen.
  - **Origin Activity -** Creates the Origin Enter screen, takes input from the user, calls the LocationService to find the user’s location (if necessary), validates addresses (if entered), and displays error messages.
  - **Results -** Creates the Results screen, displays calculated route/information, and makes calls to Google Maps.
- **Exception** (exception handling help)**:**
  - **AmbigiuousAddressException -** thrown in the event that an address is ambiguous or returns more than one address.
  - **GeocoderAPIException -** when the geocoder returns a bad status.
  - **NoInternetConnectionException -** thrown when the internet connection can’t be found (cell or wifi).
  - **NoLocationProviderException -** thrown when NetworkProvider and GPSProvider return nothing - no way to get the user’s location.
- **Fragments:**
  - **RoutyDialog -** provides information for the building of dialog boxes. Is the parent class for three types of alert dialogs.
    - **OneButtonDialog**
    - **TwoButtonDialog**
    - **ThreeButtonDialog**
- **Model** (needed Java utility objects)**:**
  - **AppProperties -** saved global static values used by other objects.
  - **Distance -** data object that holds the duration and driving distance as found through API calls.
  - **Route -** maintains overall distance of route and addresses of the route, in order.
  - **RouteOptimizerPreference -** way to select between distance or duration for routing preference.
  - **RouteRequest -** packages a request for Android async task to perform the route calculation before Results Screen.
- **Service** (classes that interface with Google/Android API)**:**
  - **LocationService -** uses LocationManager to find providers for finding the user’s location and performs logic to deal with different providers/no known provider/ using recent location.
  - **AddressService -** uses the Geocoder to create addresses from latitudes and longitudes, and creates lats and longs out of addresses. It also parses the

webcalls in the event that it can't find a Geocoder backend so that it can find or validate addresses.

- **DistanceMatrixService** - gets distances between the destinations and origin from Google Distance Matrix API.
- **InernetService** - gets an http resons from a given URL. Used for any web based API.
- **RouteService** - performs the actual routing algorithm, using other services and their associated API calls.

- **Task:**
    - **CalculateRouteTask** - an async task that calculates the route. Placing it here keeps it from overloading the UI related activities so that responsiveness doesn't suffer.