Question 2 - Answer

2. Mobile Car Navigation Application to Estimate Trip Duration

Design of car navigation mobile application to estimation of trip duration.

Please think about modules you need and challenges you might face.

You can assume all server APIs are available to use.

For this question you can describe your design in plain English and pseudocode.

Answer:

The initial idea of how to build the application appears to be really straight forward, but there are some specific cases that, if not taken into consideration, can make the application not work properly. I decided to implement the application instead of doing a pseudocode. First I explain an overview of how I approached the problem, and then I go into more detail about how I implemented the app.

The overview:

1. We have to think of how to get user's permission for location and internet, choose the Maps API to work with, and have some way for the user to input the destination.
2. After the destination is input, geocode that destination and make an API server call to the maps API with the two pins: User's location and Destination coordinates. With these two pieces of information the Map API allows us to retrieve the ETA between them.
3. Considering some possible problems that can occur, there is the possibility app freezing while the geocoding is being done, because it is waiting for its response to continue. Or crashing if there is a problem that is not being handled when there is an error calling geocoding. In order to avoid these problems we have to guarantee that either we only try to calculate the ETA once the geocode information is received, or that we handle possible errors with not getting the information correctly or having some issue with the internet connection while the geocode request is made.
4. Another possible error is to have two locations between which the ETA cannot be calculated, for instance, trying to put the two pins between the United States and Europe and calculate the ETA of driving between them. Since there is no way to calculate that, we need to create an error handling for it, so the app won't crash. And finally accounting for the user to pass an address that doesn't exist.
5. After accounting for all those possibles errors, we can start to implement the code, create the UI, the buttons, layout, and connect the view with the viewcontroller.
6. Last, but not least, find a way to display the ETA in a "pretty date" format, since it would probably come in seconds, so we would need to convert it into a time format that the user would understand, including minutes, hours, days, etc.

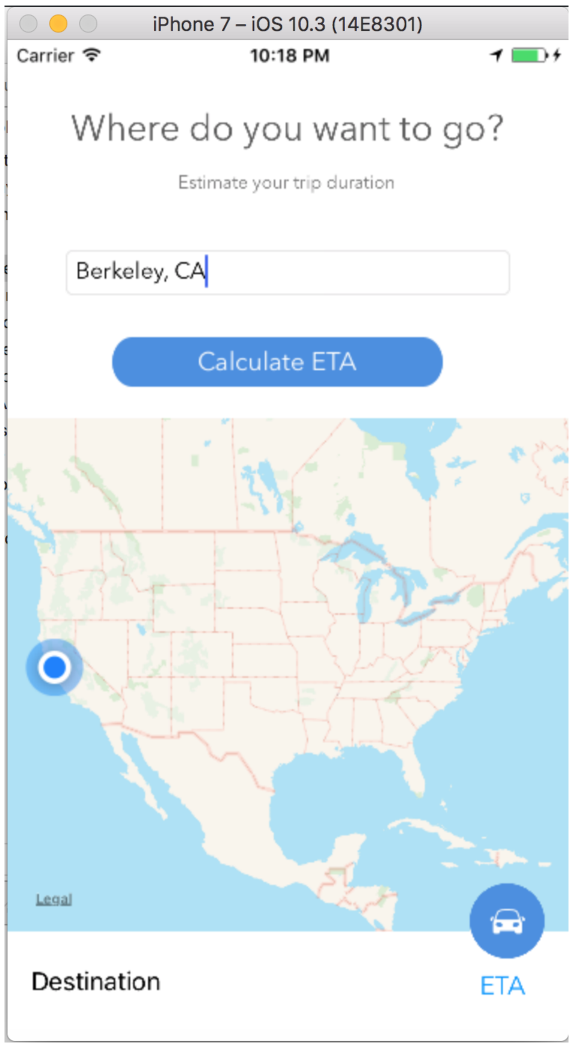
A more detailed explanation of how I implemented the app is as follows:

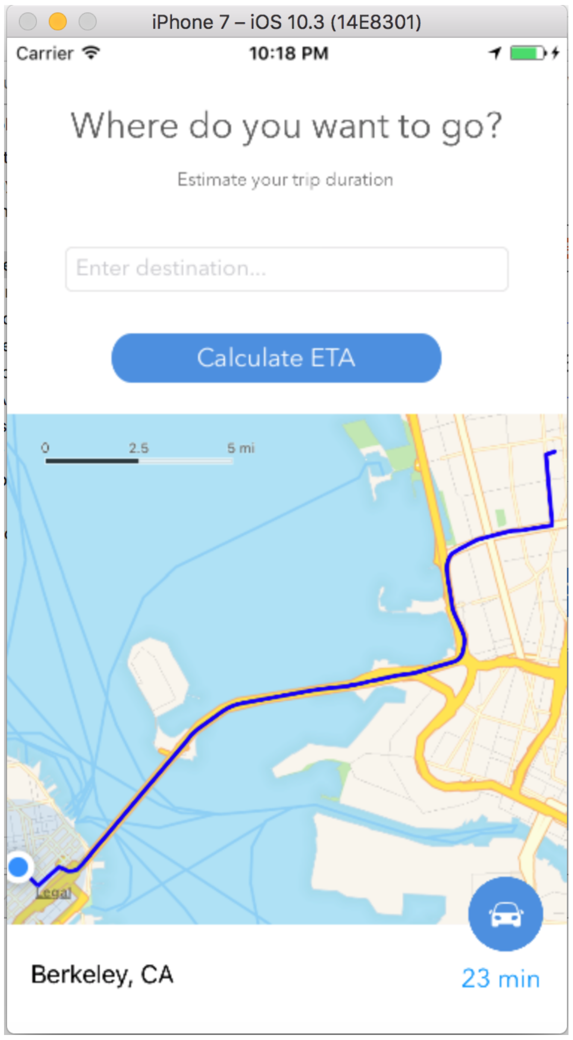
Initially I decided to set up a first prototype of the layout and thought about what information and permissions I'd need from the user. In this case I needed to have the user's permission to access its current location as well as the internet. The location permission is necessary to get the latitude and longitude coordinates of where the user is so we can have the start point of the trip. We could alternatively let the user input the start location, which I did not implement in the application. After the permissions were taken care of, I decided to use MapKit from Apple in order to render the map, display user's location, and, most importantly, get the ETA. Once I had in mind what I needed, I started working on the trivial parts.

Since iOS uses the MVC pattern, first I implemented the UIView with a Mapkit, a UITextfield, to enter the destination, a UIButton to trigger the map rendering, the Geolocation lookup, and the map rendering, and a few UILabels to guide the user and display the information. Once that was done, I implemented the UIViewController code so when the button was pressed, it would get the location from the UITextfield and pass it to the geocoder. Here there is a catch: the geocoding takes longer to process than calling the render map function does; therefore, if I called them in a normal order, the map would try to render itself before the locations were in, causing an error. So I embedded the call to rendering map inside the method for getting the geocode location, so it would only be called after the coordinates were in. After that I had to think of how many routes the api would give. I only took the first one, which is usually the fastest. I implemented the rendering map function embedded in a sort of completion handler set up, so when it was called, it was guaranteed that we had the user's location and destination coordinates already.

To finalize, I had to make sure that there was a pretty date display for the user, since the Maps API returns the ETA in seconds, so I implemented the time in minutes (still need to do it for hours, days, etc) to make it easier for the user to understand. The last piece was to dismiss the keyboard and clear out the location field, so a new location could be implemented, and I clear the map to add the new overlay.

Screenshots of the app:


iPhone 7 – iOS 10.3 (14E8301)
Carrier
10:18 PM
Where do you want to go?
Estimate your trip duration
Berkeley, CA
Calculate ETA
Legal
Destination
ETA


iPhone 7 – iOS 10.3 (14E8301)
Carrier
10:18 PM
Where do you want to go?
Estimate your trip duration
Enter destination...
Calculate ETA
0
2.5
5 mi
Legal
Berkeley, CA
23 min