COSC 4355/6355 – Introduction to Ubiquitous Computing

Polling Station – Project Specification

December 5, 2019

Background

With how the world is changing today, it is more important than ever to be able to exercise our democratic right to vote. However, it is also more confusing than ever to figure out the correct polling area for your district. In these modern times, we'd like a convenient app to help users easily find their polling place and easily fulfil their civic duty. We're using GPS in phones and Google API to find the correct polling station quickly and easily. We also help create better informed voters by having a brief summary of all the candidates on the ballot for their district, and any ballot issues being voted on.

Team 4

Matthew Rodriguez – team leader

- Suggested the topic, researched usage of Google API, wrote starter code for beta-version of the app
- wrote code for functions of beta-version of the app
- found out the usage of GoogleMaps to replace Mapkit
- Fixed bugs and completed the final backend version of the app.

Hoang D Vo

- wrote code for functions and access to firebase database for the beta-version of the app
- helped in adjusting UI designs, drew the diagram of backend architecture, draw sequence diagrams, did project specification
- added marker features for final routing function, tested performances of the product and gave feedback

Lichao Duan

- wrote code for functions in backend, added in more features of markers on map to make the app more appealing and attractive
- helped fix bugs, test product, and complete the final backend version of the app

Ailey James

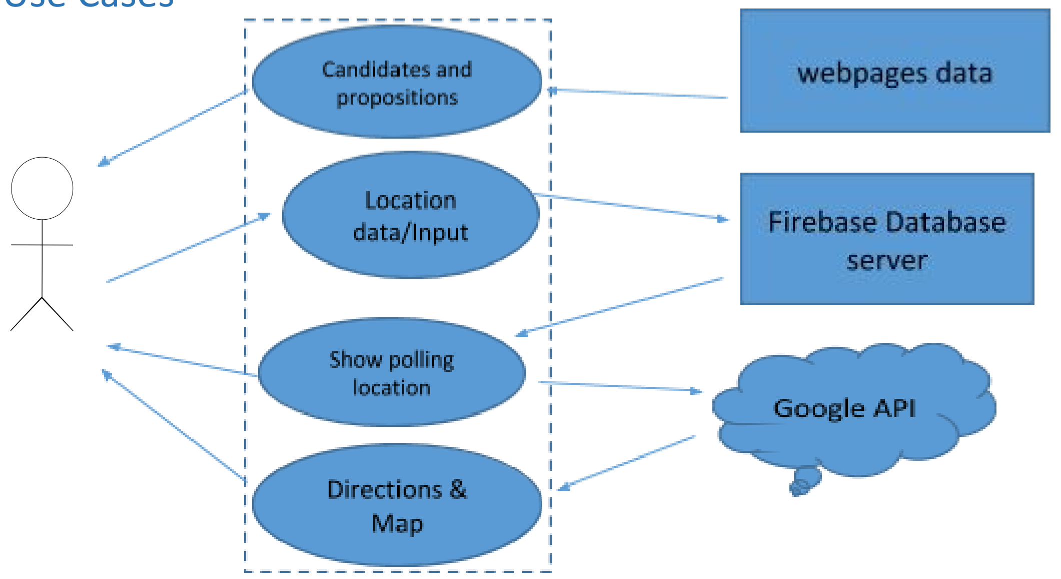
- Designed UI, created assets, found resources for candidates and propositions.
- Compiled and formatted information of voting locations, converted them into JSON files, found coordinates for locations, and created, maintained, and updated Firebase database.

- Helped create documentation and presentations.

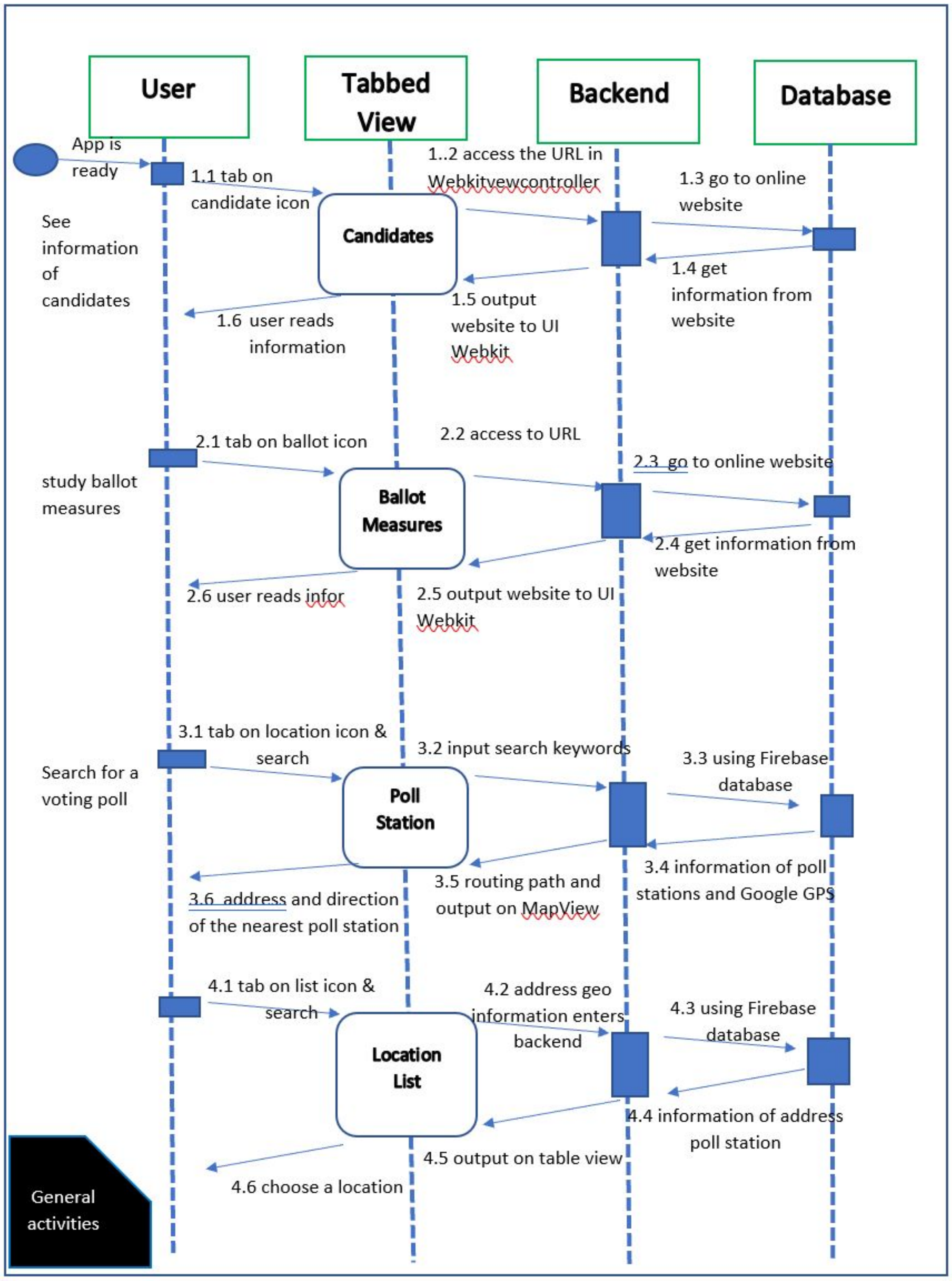
Features

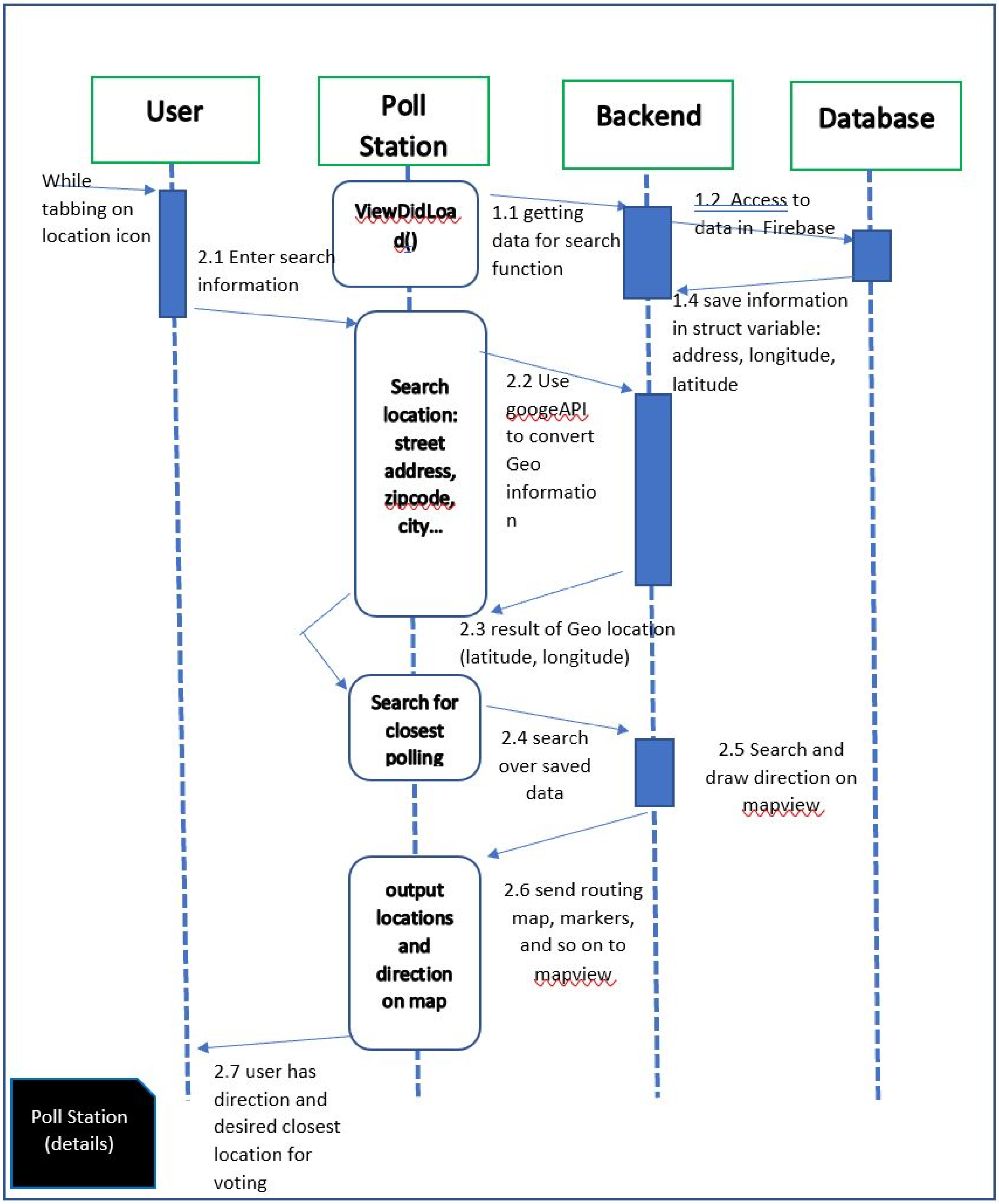
1. Find Polling Stations
   - With either GPS or Zip code search, it pulls from a database of polling station locations and brings up a list of the ones for your district. The search result is shown in map view of Google.
2. Map
   - Shows nearest polling location
   - Gives directions to the polling station.
3. List of nearby polling stations
   - show a list of near polling stations.
4. List of candidates
   - Gives information on all the candidates being voted for at that polling station.
5. Information on ballot issues being voted on
   - Also gives a summary of any propositions being voted on at that voting station.

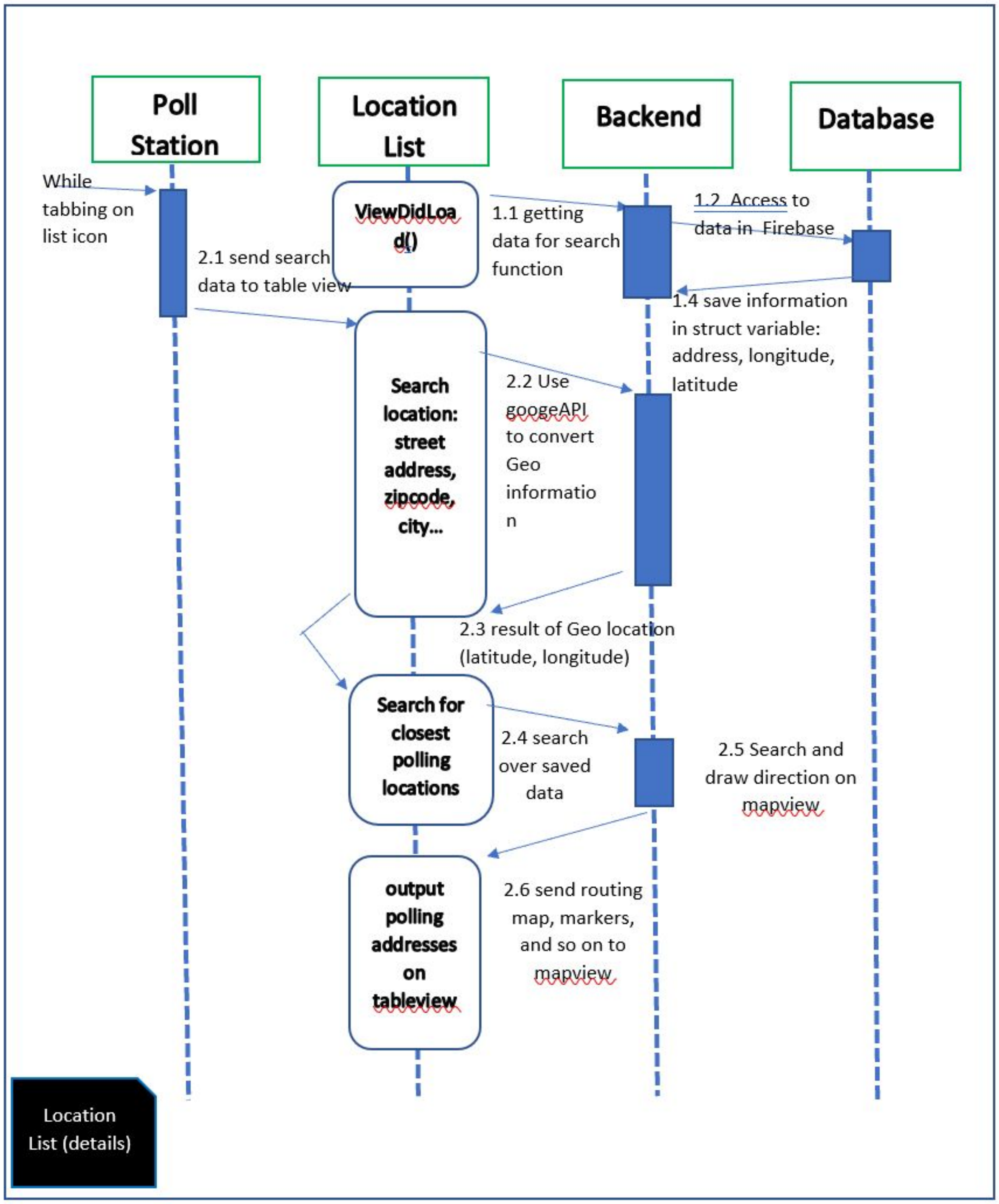
Use Cases



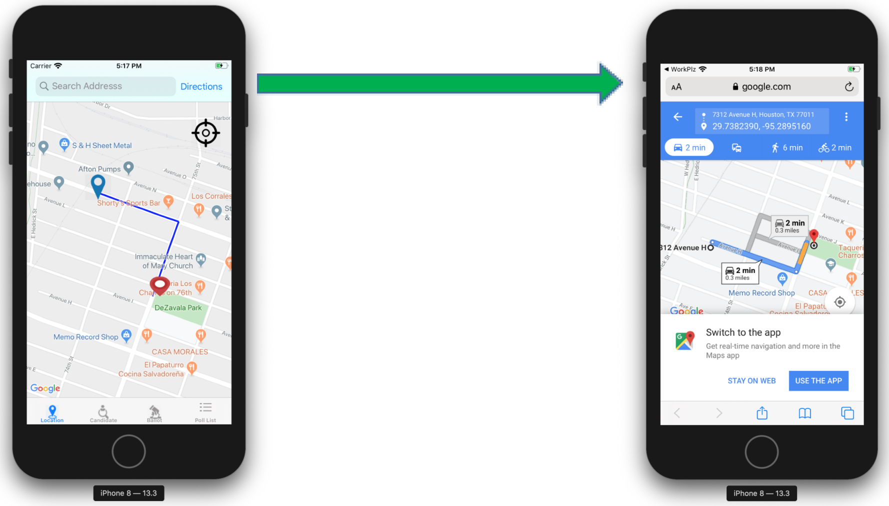
Sequence diagrams


User
Tabbed View
Backend
Database
App is ready
1.1 tab on candidate icon
1..2 access the URL in Webkitvewcontroller
1.3 go to online website
Candidates
See information of candidates
1.4 get information from website
1.5 output website to UI Webkit
1.6 user reads information
2.1 tab on ballot icon
2.2 access to URL
2.3 go to online website
study ballot measures
Ballot Measures
2.4 get information from website
2.5 output website to UI Webkit
2.6 user reads infor
3.1 tab on location icon & search
3.2 input search keywords
3.3 using Firebase database
Search for a voting poll
Poll Station
3.4 information of poll stations and Google GPS
3.5 routing path and output on MapView
3.6 address and direction of the nearest poll station
4.1 tab on list icon & search
4.2 address geo information enters backend
4.3 using Firebase database
Location List
4.4 information of address poll station
4.5 output on table view
4.6 choose a location
General activities

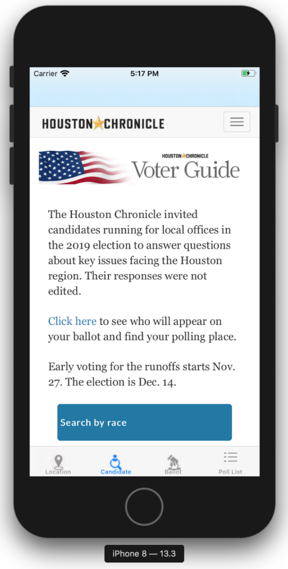

User
Poll Station
Backend
Database
While tabbing on location icon
ViewDidLoad()
1.1 getting data for search function
1.2 Access to data in Firebase
1.4 save information in struct variable: address, longitude, latitude
2.1 Enter search information
Search location: street address, zipcode, city...
2.2 Use googeAPI to convert Geo information
2.3 result of Geo location (latitude, longitude)
Search for closest polling
2.4 search over saved data
2.5 Search and draw direction on mapview
output locations and direction on map
2.6 send routing map, markers, and so on to mapview
2.7 user has direction and desired closest location for voting
Poll Station (details)


Poll Station
Location List
Backend
Database
While tabbing on list icon
ViewDidLoad()
1.1 getting data for search function
1.2 Access to data in Firebase
1.4 save information in struct variable: address, longitude, latitude
2.1 send search data to table view
Search location: street address, zipcode, city...
2.2 Use googeAPI to convert Geo information
2.3 result of Geo location (latitude, longitude)
Search for closest polling locations
2.4 search over saved data
2.5 Search and draw direction on mapview
output polling addresses on tableview
2.6 send routing map, markers, and so on to mapview
Location List (details)

Design Blueprint



In the first UI of tab view- Location, user can input an address information, and the map will show the routing from the searched address (location-A) to the nearest polling location. The blue marker is the starting point. The red marker is the destination. If user clicks on the link Direction, the app will lead user to the usage of GoogleMap website on Siri app, so User can have direction.

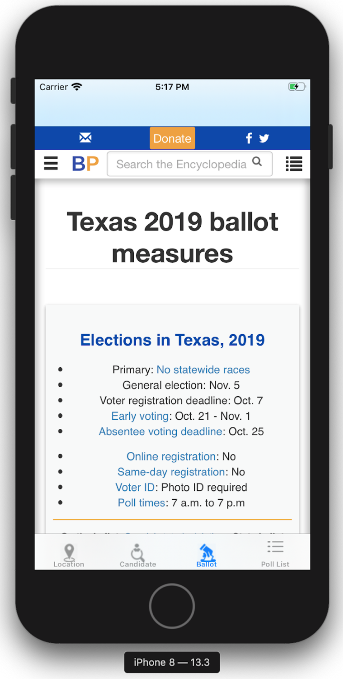


The Houston Chronicle invited candidates running for local offices in the 2019 election to answer questions about key issues facing the Houston region. Their responses were not edited.

Click here to see who will appear on your ballot and find your polling place.

Early voting for the runoffs starts Nov. 27. The election is Dec. 14.

In the second UI of tab view- Learn about candidates, user will see an online website with information of candidates. The user can interact with the website “Houston Chronicle” to know more about their candidates’ backgrounds.



Carrier 5:17 PM

Donate

BP Search the Encyclopedia

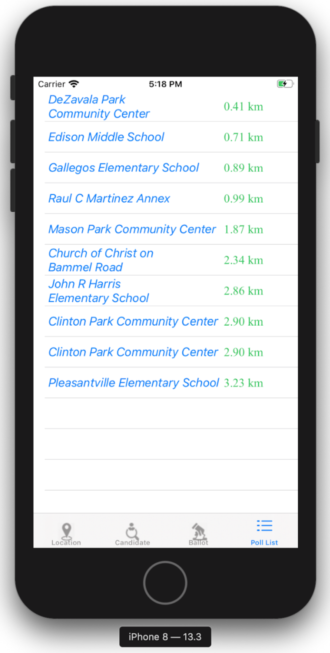
Texas 2019 ballot measures

Elections in Texas, 2019

- Primary: No statewide races
- General election: Nov. 5
- Voter registration deadline: Oct. 7
- Early voting: Oct. 21 - Nov. 1
- Absentee voting deadline: Oct. 25

- Online registration: No
- Same-day registration: No
- Voter ID: Photo ID required
- Poll times: 7 a.m. to 7 p.m

In the third UI of tab view- ballot measures, user will see an online website with information about ballot measures.

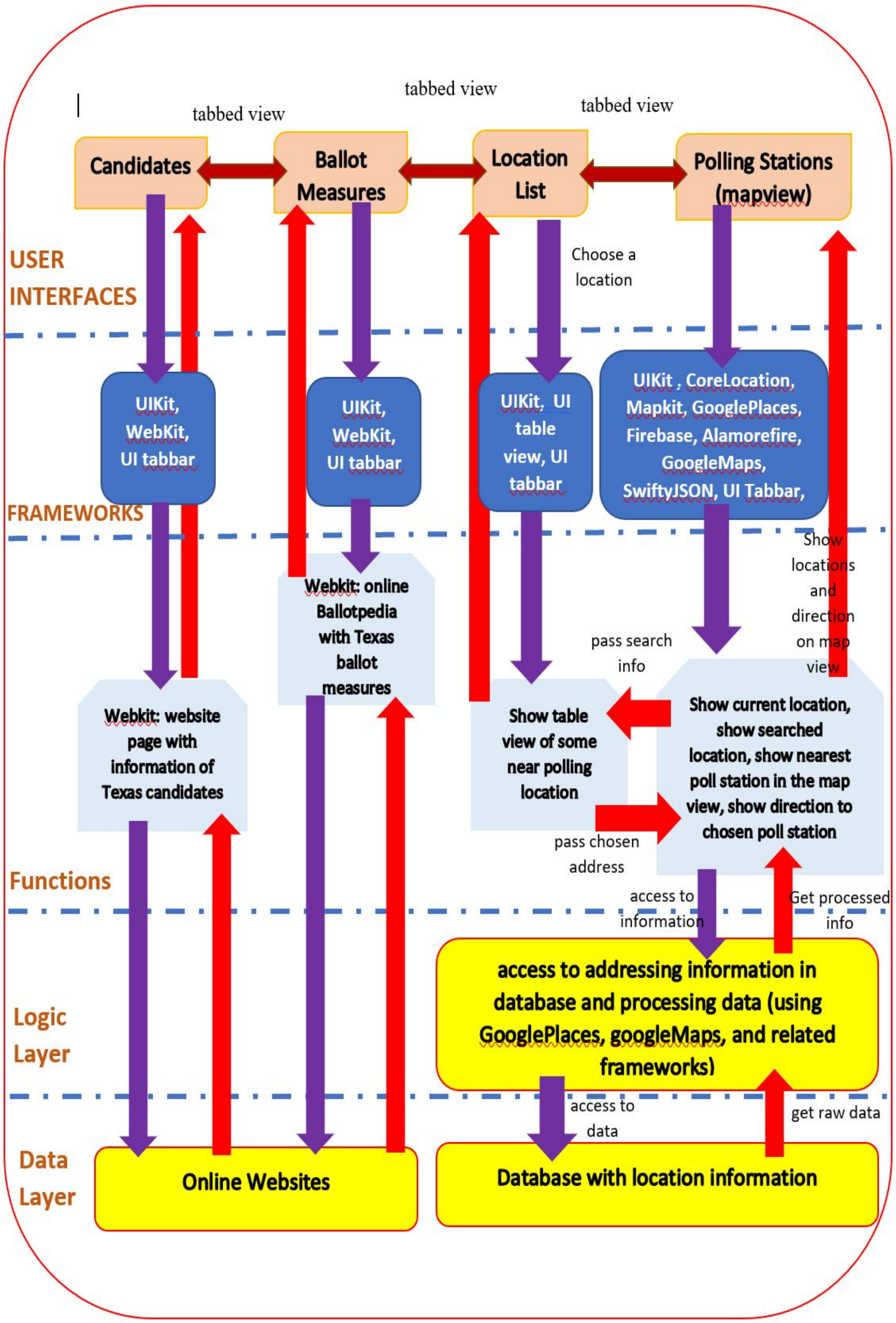


Carrier 5:18 PM

DeZavala Park Community Center	0.41 km
Edison Middle School	0.71 km
Gallegos Elementary School	0.89 km
Raul C Martinez Annex	0.99 km
Mason Park Community Center	1.87 km
Church of Christ on Bammel Road	2.34 km
John R Harris Elementary School	2.86 km
Clinton Park Community Center	2.90 km
Clinton Park Community Center	2.90 km
Pleasantville Elementary School	3.23 km

In the fourth UI of tab view- location list, user will see a list of some polling locations that are near the searched location. The locations are showed as a tableview. User just needs to tab on a location, location-B. After that, user can go back to first UI- location, the mapview will show route from the searched location-A to location-B.

Backend Architecture


tabbed view
tabbed view
tabbed view
Candidates
Ballot Measures
Location List
Polling Stations (mapview)
USER INTERFACES
Choose a location
UIKit, WebKit, UI tabbar
UIKit, WebKit, UI tabbar
UIKit, UI table view, UI tabbar
UIKit , CoreLocation, Mapkit, GooglePlaces, Firebase, Alamorefire, GoogleMaps, SwiftyJSON, UI Tabbar,
FRAMEWORKS
Show locations and direction on map view
Webkit: online Ballotpedia with Texas ballot measures
pass search info
Webkit: website page with information of Texas candidates
Show table view of some near polling location
Show current location, show searched location, show nearest poll station in the map view, show direction to chosen poll station
pass chosen address
Functions
access to information
Get processed info
access to addressing information in database and processing data (using GooglePlaces, googleMaps, and related frameworks)
Logic Layer
access to data
get raw data
Data Layer
Online Websites
Database with location information

There are 5 layers in the backend diagram. The first three layers work together mainly as presentation tier. The fourth layer is the logic tier. The final layer is the data tier

- Presentation Tier:

-User Interfaces (UIs) layer: This is the first layer that user will experience visually when using the app. There are simply 4 basic User Interfaces in this layer: Learned candidates, ballot measures, polling station, and location list. The content and activity of "User Interfaces" layer can be adjusted by the third layer-functions layer, and the fifth layer- data layer. Four UIs Candidates, Ballot Measures, location list, and Poll Station could be interchanged by tabbed view feature of Xcode. The Poll Station is associated with a Mapview for the routing feature.

-Framework Layer: Every user interface in Swift will need at least one framework to be connected to backend. This layer shows all frameworks that each user interface in "User Interfaces" layer will need.

-Functions Layer: This layer contains functions or activities that each UI will have. It could be either a link to some online websites, or it could be coded functions that perform search, sort, and draw direction on map. These works will need data in database layer, the fifth layer

- Logic tier:

-Logic Layer: the fourth layer receives requests to access database from the functions layer while processing raw data for functions layer. It gets information from total database. It provides the database to functions of UIs. It directly affects Poll Station UIs.

- Data Tier:

-Data Layer: provide total information to Logic Layer and functions layer.

+ Online websites: Website database will store candidate, and provision information.

+ Google API and the standard app library are also things we are using for our app. Firebase database to store address information of polling locations.

Dependencies

Data will be derived from database which the app will gather via URLs: websites with information of candidates and ballot measures; government website which has necessary information of polling addresses.

- Firebase backend for customized data storage: storing address and geo information of polling locations in Texas
- We added Google API in order to use the map, and Cocoapods Framework to manage libraries. The packages Alamorefire, Firebase, and SwiftyJson help our App access to Firebase database.
- We used Webkit to connect the app to online websites. The URLs will help Webkit show the websites in the app.

Assumptions

One major assumption that the Polling Station app system will have is that the google map is correct and the information on polling places and candidates/propositions are accurate (as we are getting the information from credible sources).

Another assumption is that the online websites are accurate and reliable in updating information about candidates and ballot measures on time before voting time. We must assume that the user could be able to interact with information system of online website. At the same time, the websites must have tight security, so no hacker would try to libel or false information, leading to bias results.

Finally, the database system is constrained for only polling locations and candidates in Texas. We assume that the user is registered to vote and knows where they live in Texas. In the future, we can develop and expand the App to wider geography area if we have help in database, information, and investment.

Testing Plans

Testing getting location information and that pulling any information the user needs works properly.

We tested performance while coding to complete and improve each feature of the app. By testing during coding, we could debug and quickly figure out errors in code, not waiting until the last minute. The testing started at the beginning of October 2019, when we wrote the first starting code.

During the mid-phrase of the project, we showed the basic behaviors of the phone in front of the class, a major step of testing plan. Thanks to that event, we could have feedbacks and experience to make necessary changes in backend and UI designs.

While we were testing, we check behaviors of UIs and performance of backend, so we can improve speed and find alternate solutions for errors.