iOS Application Development

B.U.EYEs

Linli Chen/99173294
Jinwoong Lee/12661930
Zhongtao Chen/12640874

IOS Application

CONTENTS

01. Target audience

02. The pain points

03. Solution

04. Video description

05. Pleasant to use:

06. Technology

07. Question time

App Description

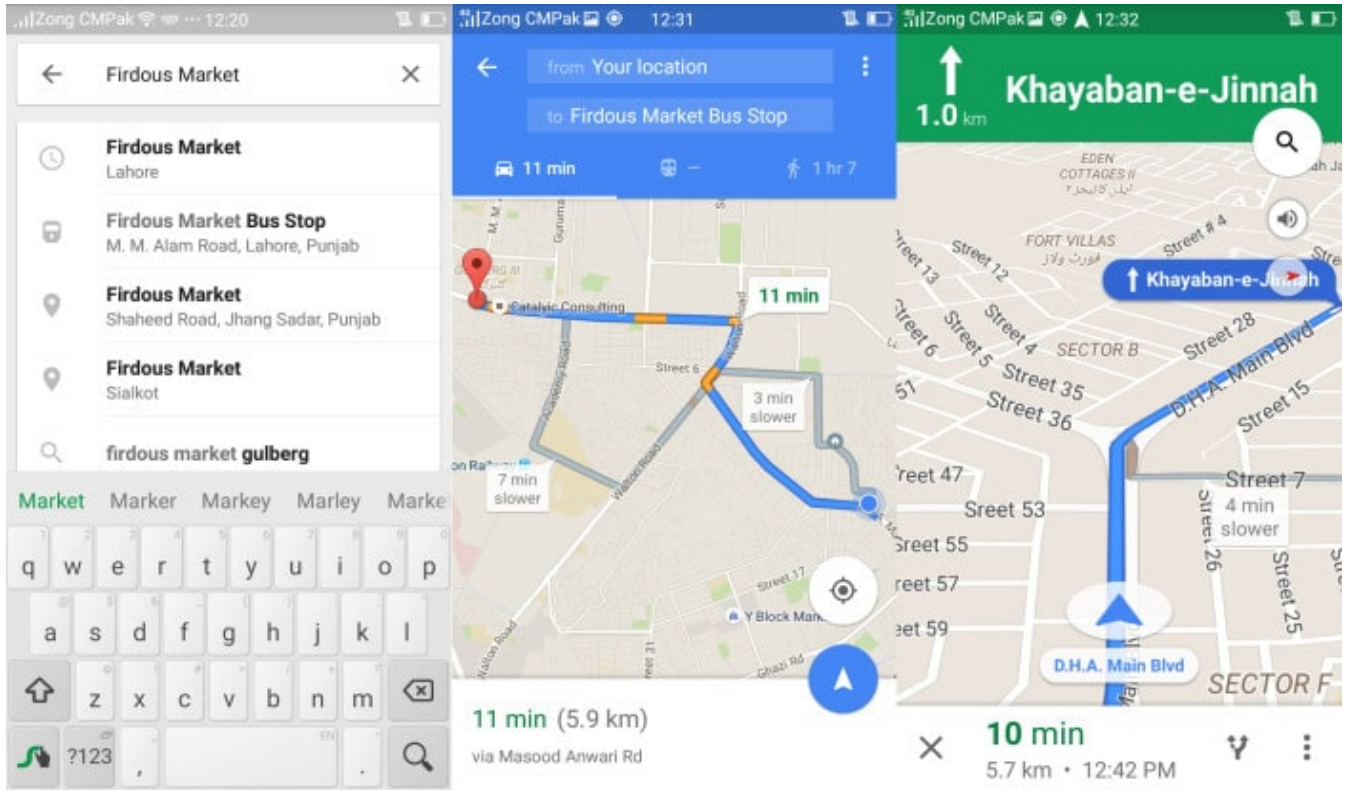
Target audience: blind people

https://gdb.voanews.com/4B5B0413-B66C-4C90-8D97-5AFACA974E35_w1023_r1_s.jpg

http://www.graphis.com/media/uploads/cfe/entry/188e64bc-249e-4a65-bf58-e57c9846de0c/AT&T%20Blind%20Texting.jpg

The pain points:

- ✓ **normal map application only could do navigation or location report when user input the destination or a specific location**
- ✓ **cannot report nearby places, only showing the pin on the map**

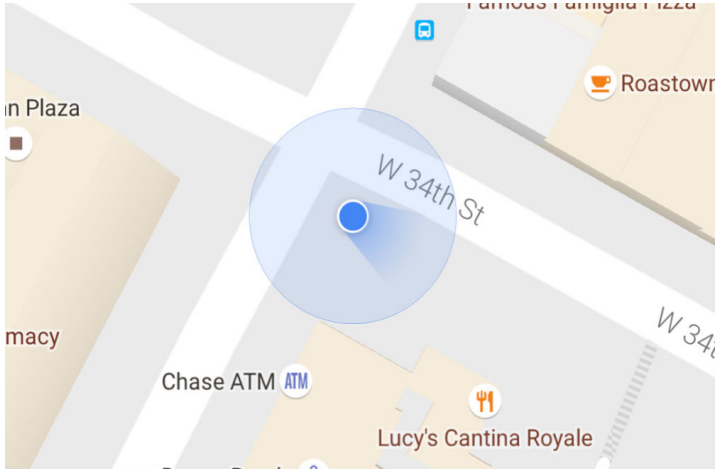


https://www.pakistankakhudahafiz.com/pkkhnew/wp-content/uploads/2016/03/google-maps-introduce-turn-by-turn-navigation-in-pakistan-700x415.jpg

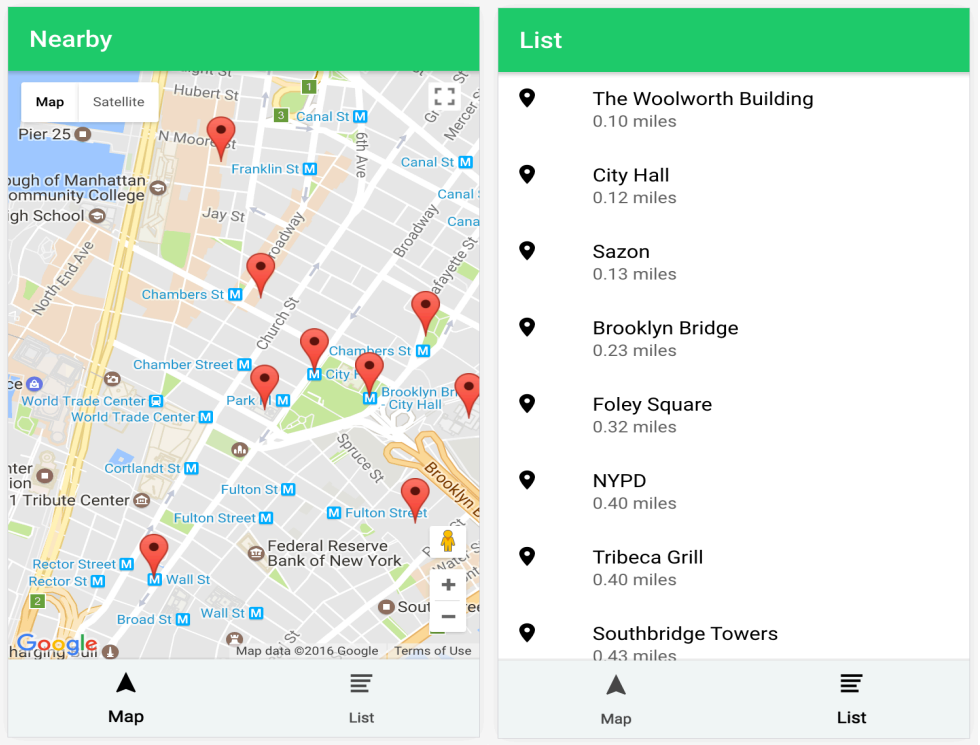
App Description

Solution in this application:

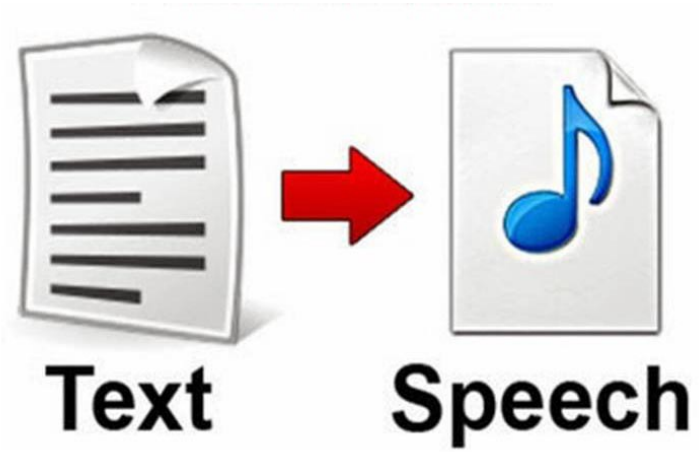
- ✓ report current location in a specific period
- ✓ search nearby places, and estimate the course, then list the top hot places
- ✓ all the location and information will provide to user via sound



https://1.bp.blogspot.com/-wzZvzYTo7Dc/WR72BmeRqeI/AAAAAAAATX4/ISNvUa8V_kcQZpO-HRXnqRpCCCgXmq_8QCLcB/s1600/Google-Maps.jpg



https://www.joshmorony.com/wp-content/uploads/2016/12/nearby-finished.png



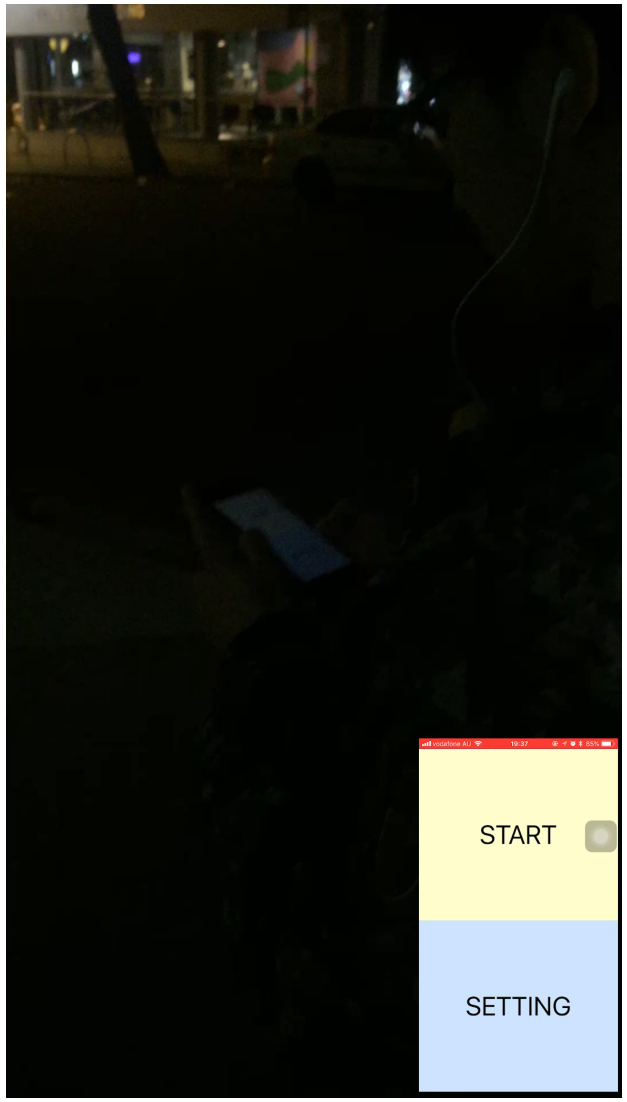
https://cdn-images-1.medium.com/max/741/1*ZvBSWwvLOCaIz29G8tAwUQ.jpeg

App Description

In use via video

(Screen recording and video shooting at same time)

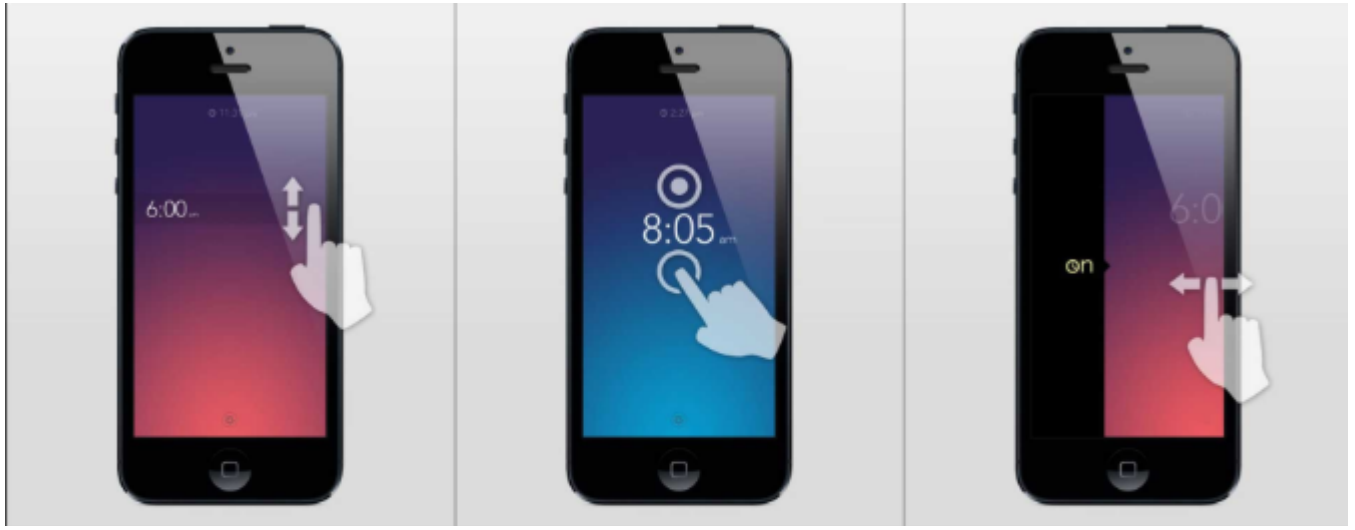
Video link: https://drive.google.com/file/d/1p2LYIQT93FXkf8PLL7IEzgdORK7U8H-Y/view?usp=sharing



App Description

Pleasant to use:

- ✓ simple UI for blind people
- ✓ gesture control
- ✓ audio introduction and guide



http://babich.biz/content/images/2016/03/1-25X_C_jfo39jV9ZrpT3P_g.png

Technology:

- ✓ Foundation
- ✓ UIKit
- ✓ CoreLocation (get the location)
- ✓ AVFoundation (Text to speech)
- ➢ Third-party Framework
  - ✓ GooglePlaces (search the nearby places, get JSON result)
  - ✓ Alamofire (deal with Swift-based HTTP networking, and receive result)
  - ✓ SwiftyJson (deal with JSON data)
  - ✓ MediaWiki (nearby places description)

App Description

Technology:

- **UIKit**
  - ✓ **Gesture**

```
override func viewDidLoad() {
    super.viewDidLoad()

    // init props
    reset()

    // init gesture
    registerSwipe()

    // init button action
    registerButtonTap(button: frontButton, singleTapAct: .frontButtonST, doubleTapAct: .frontButtonDT)
    registerButtonTap(button: rightButton, singleTapAct: .rightButtonST, doubleTapAct: .rightButtonDT)
    registerButtonTap(button: backButton, singleTapAct: .backButtonST, doubleTapAct: .backButtonDT)
    registerButtonTap(button: leftButton, singleTapAct: .leftButtonST, doubleTapAct: .leftButtonDT)
    registerButtonTap(button: midButton, singleTapAct: .midButtonST, doubleTapAct: .midButtonDT)

    // show button
    showMidButton()
}
```

```
    // single & double
    func registerButtonTap(button: UIButton, singleTapAct: Selector, doubleTapAct: Selector) {
        let singleTap =  UITapGestureRecognizer(target: self, action: singleTapAct)
        singleTap.numberOfTapsRequired = 1
        button.addGestureRecognizer(singleTap)

        let doubleTap = UITapGestureRecognizer(target: self, action: doubleTapAct)
        doubleTap.numberOfTapsRequired = 2
        button.addGestureRecognizer(doubleTap)

        singleTap.require(toFail: doubleTap)
//        singleTap.delaysTouchesBegan = true
//        doubleTap.delaysTouchesBegan = true
    }

    // swipe
    func registerSwipe() {
        registerDownSwipe()
        registerRightSwipe()
    }

    func registerDownSwipe() {
        let downSwipe = UISwipeGestureRecognizer(target: self, action: .downSwipeAct)
        downSwipe.direction = .down
        downSwipe.numberOfTouchesRequired = 2
        self.view.addGestureRecognizer(downSwipe)
    }

    @objc func handleDownSwipe(_ sender: UISwipeGestureRecognizer) {
        guard let sd = self as? SwipeDelegate else {
            fatalError("Type 'ViewController' does not conform to protocol 'SwipeDelegate'")
        }
        speechUtil.speakTextImmediately(text: sd.getPageIntroInDetail());
    }

    func registerRightSwipe() {
        let rightSwipe = UISwipeGestureRecognizer(target: self, action: .rightSwipeAct)
        rightSwipe.direction = .right
        rightSwipe.numberOfTouchesRequired = 2
        self.view.addGestureRecognizer(rightSwipe)
    }
```

App Description

➢ AVFundation

✓ Text --> Speech

```
class TextToSpeech : NSObject, AVSpeechSynthesizerDelegate
{
    var text : String!
    var voice_language : String!
    var voice_rate : Float! // from 0.5 to 2, default 1
    var voice_volume : Float!
    var voice_gender : Gender!
    var speechSynthesizer = AVSpeechSynthesizer()

    override init()
    {
        text = ""
        voice_language = "en-US"
        voice_rate = 0.6
        voice_volume = 1
        voice_gender = Gender.female
    }
    func speakText(text : String)
    {
        voice_rate = appSetting?.rate.rawValue

        if !speechSynthesizer.isSpeaking
        {
            let utterance = AVSpeechUtterance(string: text)
            utterance.voice = AVSpeechSynthesisVoice(language: voice_language)
            utterance.rate = voice_rate
            utterance.volume = voice_volume
            // set the time before handling the next queued utterance
            utterance.postUtteranceDelay = 0.1
            speechSynthesizer.speak(utterance)
        }
    }

    // Created by Linli Chen on 2/6/18.
    func speakTextImmediately(text: String) {
        stopSpeech()
        speakText(text: text)
    }
```

```
func pauseSpeech()
{
    if speechSynthesizer.isSpeaking
    {
        speechSynthesizer.pauseSpeaking(at: AVSpeechBoundary.word)
    }
}
func stopSpeech()
{
    if speechSynthesizer.isSpeaking || speechSynthesizer.isPaused
    {
        speechSynthesizer.stopSpeaking(at: AVSpeechBoundary.immediate)
        speechSynthesizer = AVSpeechSynthesizer()
    }
}
func continueSpeach()
{
    if speechSynthesizer.isPaused
    {
        speechSynthesizer.continueSpeaking()
    }
}
func setVoiceType(voiceLanguageType : String)
{
    voice_language = voiceLanguageType
}
func setRate(rate : Float)
{
    voice_rate = rate
}
func setVolume(volume : Float)
{
    voice_volume = volume
}
```

App Description

➢ CoreLocation

✓ Location

```
class MapControl: NSObject, CLLocationManagerDelegate {

    let placeSearchKey = "AIzaSyCXjncpMkQAeoNQRGgDYjoWjLI5MOT7aoI"

    // location manager to get location
    var locationManager = CLLocationManager()
    var currentLocation: CLLocation?
    var currentAddress: String?
    var currentHeading: CLLocationDirection!
    var placesClient: GMSPlacesClient!
    var nearbyPlaces: [NearbyPlace] = []
    var frontPlaces: [NearbyPlace] = []
    var rightPlaces: [NearbyPlace] = []
    var backPlaces: [NearbyPlace] = []
    var leftPlaces: [NearbyPlace] = []
    var numUpdate = 0

    let speak = speechUtil
    var canSpeak = true
    var autoSpeaking = true
    var enable = false

    override init() {
        super.init()
        self.locationManager.delegate = self
        // the accuracy of the location, set it the best
        self.locationManager.desiredAccuracy = kCLLocationAccuracyBes
        //         set distance and heading filter, decrease the frec
        self.locationManager.distanceFilter = 10
        self.locationManager.headingFilter = 45
        locationManager.requestWhenInUseAuthorization()
        //        self.locationManager.requestAlwaysAuthorization()
        locationManager.startUpdatingLocation()
    }
```

```
func locationManager(_ manager: CLLocationManager, didUpdateHeading newHeading: CLHeading)
    {
    currentHeading = newHeading.trueHeading
}

func locationManager(_ manager: CLLocationManager, didUpdateLocations locations:
    [CLLocation]) {
    print("num of update: \(numUpdate)")
    numUpdate += 1

    // the most recent location update is at the end of the array, and the accurancy is
        most best
    let location = locations[locations.count - 1]
    print("longitude = \(location.coordinate.longitude), latitude = \
        (location.coordinate.latitude)")
    currentLocation = location
    convetToAddress(location: currentLocation!)

    if currentLocation == nil {
        return
    }
    let searchLocation: String = "\(currentLocation!.coordinate.latitude),\
        (currentLocation!.coordinate.longitude)"
    print(searchLocation)
    let searchParams: [String: String] = ["location": searchLocation, "radius": "50",
        "key": placeSearchKey]

    // get nearby places, save in array,
    fetchLocationInfo(parameters: searchParams)
}
```

App Description

- ➢ **Google Places Search**
- ➢ **Alamofire & SwiftyJSON**
  - ✓ **Nearby Places**
  - ✓ **Deal with the data**

```
func fetchLocationInfo(parameters:[String: String]) {
    guard let url = URL(string: "https://maps.googleapis.com/maps/api/place/nearbysearch/
        json")
        else {
            print("wrong url")
            return
    }
    // url request from google place search web, get json, the order of the result depends
        on the importance
    Alamofire.request(url, method: .get, parameters: parameters).responseJSON {
        response in
        if response.result.isSuccess {
            print("Success!")
            guard let data = response.data else {
                print("data nil")
                return
            }
            do {
                let jsonResult = try JSONDecoder().decode(NearbyPlaceJson.self, from: data)
                self.nearbyPlaces = jsonResult.results

                self.decideCourse(nearbyPlaces: self.nearbyPlaces)
            } catch {
                print("error2: \(error)")
            }
        }
        else {
            // networking has problems
            print("Error: \(response.result.error!)")
        }
    }
}
```

App Description

- **MediaWiki**
  - ✓ **Nearby places description**

```
func fetchDescription(spot: String) {
    parameters["titles"] = spot

    guard let url = url
        else {
            print("wrong url")
            return
    }
    var description: String?
    Alamofire.request(url, method: .get, parameters: parameters).responseJSON {
        response in
        if response.result.isSuccess {
            print("Success!")
            guard let data = response.data else {
                print("data nil")
                return
            }
            print(data)
            let jsonData = JSON(data)
            print(jsonData)
            for (_, value) in jsonData["query"]["pages"] {
                description = value["extract"].stringValue
            }

            guard var description = description else {
                speechUtil.speakTextImmediately(text: "sorry, do not have enough data")
                return
            }

            // description do not have enough info
            if self.checkEnoughData(string: description) == false {
                speechUtil.speakTextImmediately(text: "sorry, do not have enough data")
                return
            }

            // get enough data
            description = self.cleanString(string: description)
            print(description)
            speechUtil.speakTextImmediately(text: description)
        }
        else {
            // networking has problems
            print("Error: \(response.result.error!)")
        }
    }
}
```

Current Problem:

- **Asynchronous problem**
  - ✓ **All the API search and location method are asynchronous**
  - ✓ **Cannot block main thread**
- **Description data not enoughc**
  - ✓ **Description data form wiki, but the data not enough, some place cannot get data**
  - ✓ **Need complex data source and algorithms to decide the data source, which data could be used**
- **Energy saving**

Following Work:

- **Find a better way to deal with the data**
  - ✓ **Using various data source**
  - ✓ **Find which places should use which database**
- **Add route navigation function**

Question time

THANK YOU