NavSB Draft Project
Team KIWI

team members / email

Ben Ye: z_ye@ucsb.edu
Joon Lee: jlee11@ucsb.edu
Dawit Aboye: dawit@ucsb.edu
Kalyn Klimek: kalynklimek@ucsb.edu
Alex Rasla: alexrasla@ucsb.edu

FULL video demo: https://youtu.be/ErK_XL7KfKI
New Github: https://github.com/nageplz1/Kiwi-NavSB (Created on May 17, 2019)
Old Github: https://github.com/jlee122/KIWI_CS48
Slack: https://kiwics48.slack.com/messages/CHN0R5T6H/details/
Trello: https://trello.com/b/nxdetmJq/cs48-project
Travis-CI: https://travis-ci.org/jlee122/KIWI_CS48

What is our project about?

The problem and how we want to solve it:

Students in college are overwhelmed and confused about a variety of things. One main barrier that students have to face is getting to know their new home - both the school campus and the city. We feel this is an opportunity to address some shortcomings presented by the simplicity of other UCSB campus resources. For example, many online UCSB maps are a simple digital view, without any notable interactive features that ease the navigation process or help discern/give information pertaining to certain buildings. In addition, an AR feature with pins and arrows would make navigating and routing easy, eliminating the difficulty in judging your direction or relativity from one location A to location B. We want to build these features to make a user friendly iOS app that helps guide students through their time in college and minimizes these barriers.

Why the problem is important:

This problem is important because as students at UCSB, we know how frustrating and difficult it can be to find the fastest route to get to class or to navigate through Isla Vista and campus. We want to take this opportunity to make students' lives easier by creating an app that addresses these annoying barriers.

How the problem is solved today:

There are many apps that currently exist including GoGaucho and the official UCSB mobile app that try to address and facilitate easy navigation for their users through interactive maps. We are expanding on this by adding more features and information to the interactive 2D map and including the AR feature that allows a more functional interactive user tool and experience for students.

Identify the outcome of the project:

Users including both students and visitors will utilize our app and its features to help navigate and facilitate their experience at UCSB.

How do you plan to articulate, design, and implement a solution?

We plan on implementing our solution by utilizing the native iOS language, Swift, and building with the Xcode IDE. Swift is an object oriented language so we will be applying various OOP features, as well as incorporating C++/Objective-C subfeatures/sub-components. For our AR feature, we will be incorporating AR kit and APIs that work alongside our iOs app.

For 2D interactive map we will be using the “Mapbox” Software Development Kit. It includes a lot of features that will allow us to add more interesting features with more freedom and functionality. For our AR features, we will be using ARCL Framework that incorporates Cocoa’s ARKit and CoreLocation Frameworks.

Sprint Planning

Sprint 1 (April 15th - April 26th)

- Basic app that we can open with icon and at least a working 2D map
    - Where we are on the map
    - Swift has its own mapkit? Maybe we could use that also
    - Familiarize ourselves with google map API
    - Find out if we can use google map in our app
- Everyone on the team should get comfortable with using the XCode IDE and the Swift programming language - everyone make their own small app using Swift and XCode
- Research AR kit and how to utilize/analyze feed from iOS camera
- Assign specific features and tasks to people/groups of people
- Set up any possible tools and technologies we will need throughout the quarter
- Design and implement the user interface of our application, get it ready to demo
- Figure out/research routes to be used for AR
- Finish frontend / main screen for demo

Conclusion/Milestone: We finished every task in Sprint 1 except for finding routes to be used in AR and familiarizing ourselves with google maps API (highlighted in red above). Our project has changed a lot since the beginning in terms of which API’s we will be using. We used MapBox for the interactive map instead.

Retrospective: During this sprint we didn’t know how to divide the tasks and how to work on each part separately. To fix this, we assigned each member to a certain part (frontend, interactive map, AR). Whatever part the group member was assigned to, we dug deeper into learning that part and took responsibility for teaching it to the other group members.

Sprint 2 (April 29th - May 10th)

- Use Mapbox for 2D interactive map and have it ready for demo
    - Have interactive features working but add more later
    - Predefined pins as locations
- Learn how map API provides the route to a destination
- BasicUI: Develop Loading Screen View Controller with background/logo
- Implement basic AR features
    - Be able to incorporate pins with random pathing to make sure AR works -- working for sprint 2 demo!!
    - Make sure it can detect your coordinates + direction and assign random path from there
- Review and test main screen
- Connect main screen and 2D map
- Be able to move back and forth between view controllers
- Learn UI testing
    - Make tests for main screen
- Figure out how to use iOS location services

Conclusion/Milestone: We completed all our Sprint 2 tasks. We discovered and decided to use MapBox for our 2D interactive map and ARCL for our AR feature during this sprint. We were able to connect view controllers and begin UI testing. Also, we had basic interactive map and AR features prepared for the Sprint 2 demo. We made a lot of progress during this sprint.

Retrospective: During this sprint we had major Github issues. Merging swift files was much more difficult than we had expected and the master branch of our original github got messed up. We fixed this by creating a new Github for our project and took more care when merging these files. (Shoutout to Scott Chow for teaching us how to do this).

Sprint 3 (May 13th - May 31st)

- Learn and be comfortable with Mapbox sdk and ARCL library (all members)
- Finalize launch screen and make navSB logo
- Finish up interactive map
    - Create & add search bar that can hide and pop out

    - Connect AR to 2D interactive map - add AR button in Interactive Map callout that will segue to AR
- Add 'Detailed View' feature on buildings
    - Able to view exterior and interior of buildings with a short description
- Create pop up pinpoints for buildings on campus in AR
- Finish implementing paths for AR
    - Implement direction lines into AR UI
- Pass data between view controllers from Interactive Map to AR
- Optimize user interface, make it more appealing to navigate and use
- Finalize about us screen
- Finish UI testing
- Practice presenting
- Work on final draft project
- Create complete local/offline database of UCSB buildings
- Catch exception when location turns off
- Catch all exceptions for location on/off

Conclusion/Milestone: We finished all our sprint 3 tasks, except the ones highlighted in red (those were completed after sprint 3 ended). During sprint 3 we were able to complete our interactive map including the search bar feature, more detailed information on buildings, and an AR button that will bring the user to the AR view. We also were able to complete pinpoints in AR for buildings on the UCSB campus and direction lines for routes in AR. This sprint was consisted mostly of creating a cleaner UI view for our app and better functionality for both the interactive map and AR.

Retrospective: During this sprint we realized we were missing our C++/Java component of our project. To fix this we drafted ideas that would work for this component and we were able to complete it. We implemented a database that contains all the information about buildings and sights on campus. Testing was also something we chose to focus on and fix because it was a previous blocker. Unfortunately, we weren't able to completely fix this one. The UI testing for the main screen and about us page went smoothly but did not work for the map and AR page. This is apparently a bug in Xcode and unavoidable, so after many hours of trying to figure out how to fix this problem we decided to move forward. (Please ask Scott Chow for more details).

Challenges Faced/Overcome:

We faced many challenges throughout this quarter. Our first major challenge was learning how to use Xcode and learning Swift. Most of our team members had never worked with Xcode or Swift before starting this project, so learning these took up most of our Sprint 1. Next we faced many challenges with GitHub and merging swift files. We weren't sure which files should be ignored when committing changes and which shouldn't. After making a new GitHub for our project, we were more careful when merging files and we discovered which files should be ignored. Another challenge we faced was deciding which libraries would work best for our app's Interactive Map and AR features. Augmented Reality was difficult to learn and work with but ARCL (AR kit + CoreLocation) is what we ended up using and worked well for us. Testing was also a large challenge for us, as we didn't know what to test for the first two sprints because we had just started making our basic features. During sprint 3, we started better testing but ran into trouble with UI tests for the 2D map and AR feature (see retrospective for sprint 3 above for more details).

Missing/Remaining Features or Functionality:

There were a couple more features we hoped to integrate into our application but didn't have enough time to. The most immediate one was making better designs for our 'nodes' that would appear on buildings. We wanted to draw these for each building and maybe even take more advantage of ArKit to include 3D objects, not just 2D labels.

Another feature we didn't have a chance to get to is the touch functionality on our AR labels. We wanted users to be able to touch labels and view information, similar to our interactive map. Our AR library, ARCL, provides an interface for that functionality but it would have took too much time to learn and integrate into our project.

Requirements: Use Cases

1. Main Screen for NavSB app

Actors: Students, visitors

Pre Condition: User has opened the app and has the option to choose where to go from the main page (Kiwi About Us page, 2D Map, AR)

Basic Path

- Multiple buttons to navigate functionality of the app
- About Us: Team kiwi
- Interactive Map: Opens 2D map with search bar and ability to find walking route to location input, also has interactive info on ucsb campus buildings
- AR Feature: open AR and see pop up labels for ucsb campus buildings as you look at campus through iPhone
- Location Services must be turned on

Alternate Path

- If user has location services turned off, show alert to go to settings and turn it on

Post Condition

- Chosen screen pops up and there is an option to return back to default screen

Tests: UI test (testMain() in NavSBUITests.swift)

https://trello.com/c/zujMD5s4/32-use-case-1-main-screen-for-navsb-app

2. About Us Page

Actors: Students, visitors

Pre Condition: User has opened the app and has chosen the kiwi button

Basic Path

- Kiwi Mission statement is displayed on screen: shows the vision and meaning of our app
- team kiwi is shown with pictures of all members
- Option to return back to the main screen with a go back button

Alternate Path

- If wrong screen was pressed, return to main screen

Post Condition

- Chosen screen pops up and there is an option to return back to default screen

Tests: UI test (testAboutUs() in NavSBUITests.swift)

https://trello.com/c/upbAH1qr/72-use-case-2-about-us-page

3. Navigate to 2D map [2D]

Actors: Students, visitors

Pre Condition: User has opened the app, chose the option to view the 2D map, location enabled

Basic Path

- App uses location services to figure out exactly where you are on the map and will show it with a blue indicator
- Top of the view has a search bar feature with a list of possible locations on campus. When the 'Go' button and the '2D Map' button is pressed on one of them, the app will automatically direct the user back to the map with the route shown
- Bottom of the view will have a "View in AR" button that will direct the user to the AR function
- Bottom of the view will also have a back button that brings the user back to the main screen
- Long press on the map to drop a pin and calculate the walking route to that location
- Different views of the 2D map include Satellite, Streets, and Light

Alternate Path

- If no possible locations pop up when destination is entered, then that means that location does not exist on campus

Post Condition

- Path is shown on map after button is pressed or long press happens
- User can return to main screen

https://trello.com/c/oxaJt68j/2-use-case-3-navigate-to-2d-map-2d

4. Interactive (Detailed) Buildings Information [2D]

Actors: Students, visitors

Precondition: App is open, the 2D map is chosen on the main screen, location enabled

Basic Path

- When the user clicks the search bar button on the 2D map, it will include a list of buildings on UCSB campus
- Press the 'Go' button to view that building with more info including: exterior view, interior view, and a short description of that building

Alternate Path

- If no detailed information is available to certain small buildings, the user will not be able to click on them.

Post Condition

- The floor plan, or any other building information is displayed
- User can easily return back to the main screen using back buttons
- User can route to selected building in 2D map using the '2D map' button
- User can route to selected building in AR using the 'AR' button

https://trello.com/c/cywyJSzw/10-use-case-4-interactivedetailed-buildings-information-2d

5. Search Bar [2D]

Actors: Students, visitors

Precondition: App is open, the 2D Map is chosen, location enabled

Basic Path

- When 2D map is being viewed, the user can click on the search bar button in the top left corner of the map
- The search bar displays buildings on the ucsb campus
- User can type in destinations (ucsb buildings) in the search bar

Alternate Path

- If destination searched is not valid, the option to view or route to it is not available

Post Condition

- Press the 'Go' button to look at the interactive view of the searched destination (building)
- Press the 2D map button to get the route to the searched destination from current location on the 2D map
- Press the AR button to get directions to searched destination from current location in AR

https://trello.com/c/Zx9MzCa3/71-use-case-5-search-bar-2d

6. Navigate with AR [AR]

Actors: Students, visitors

Precondition: App is open, the AR feature is chosen, location enabled, camera access allowed

Basic Path

- App uses location services to figure out exactly where you are
- AR view shows indicators (pinpoints) on nearby UCSB buildings
- 2D map button on AR screen that will bring user to 2D map to chose route for AR (navigate back to AR using "View in AR" button

Alternate Path

- If camera access is not allowed or location is disabled, the user will not be able to see indicators.

Post Condition

- Pop up pins for nearby buildings
- User can return to main page
- Routes calculated from 2D map and direction line drawn in AR

https://trello.com/c/bdRa6qte/73-use-case-6-navigate-to-ar-ar

7. Indicators showing path [AR]

Actors: Students, visitors

Precondition: App is open, either “View in AR” button option is chosen on the interactive map after the route is displayed

Basic Path

- When AR feature is being viewed, the user can look around while the indicators are fixated to certain coordinates on screen
- Press 'View in AR' button on 2D map for route to appear in AR
- Users can follow the path to their destination, a big destination node will appear on the screen when they do so

Alternate Path

- If camera access is not allowed or location is disabled, the user will not be able to see indicators.
- If a path is being calculated/unable to be determined, the indicator will not a show a path.

Post Condition

- Direction line is displayed, allowing the user to see what direction to take.

https://trello.com/c/yEPz6Mav/24-use-case-7-indicators-showing-path-ar

8. Pop up pins in certain areas on campus [AR]

Actors: Students, visitors

Precondition: App is open, the AR feature is chosen, location enabled, camera access allowed

Basic Path

- When AR feature is being viewed, the user can look around while the pin is fixated to the coordinate destination on camera
- The indicators take into account building proximity, and won't overlap on a building.

Alternate Path

- If camera access is not allowed or location is disabled, the user will not be able to see indicators.
- If a path is being calculated/unable to be determined, the indicator will not show a path.

Post Condition

- The indicators are displayed, allowing the user to see what direction to take.

https://trello.com/c/ilbSo4wr/9-use-case-8-pop-up-pins-in-certain-areas-campus-ar

9. AR Button on interactive map

Actors: Students, visitors

Precondition: App is open, the interactive map feature is chosen, location enabled, directions/route to entered location in progress

Basic Path

- When interactive map is in use and directions are given for certain location entered by the user, there will be an option to switch to AR map
- Once this button is pressed, directions are set up in AR map setting with indicators showing

Alternate Path

- If camera access is not allowed or location is disabled, the user will not be able to see directions in AR setting
- If a path is being calculated/unable to be determined, show an error message and option to re-enter address/location

Post Condition

- AR Map is opened and camera access is allowed
- The indicators are displayed, allowing the user to see what direction to take.

https://trello.com/c/XMGCB0fk/33-use-case-9-ar-button-on-interactive-map

10. sqlite3 Database (implemented using c++)

Actors: Developers

Precondition: sqlite database is correctly set up, all the functions are free of errors and return the correct values

Basic Path

- We (the developers) can call getLatitudeCPP(building name) to get the latitude of a building to calculate routes and place AR pins
- We can do the same for getLongitudeCPP(building name)
- We can call getDesc(building name) to receive the string we will display on our interactive map.

Alternate Path

- If the building name provided does not exist, a cerr message is outputted on our console, "null" or empty values are returned so the app won't crash.
- null/empty values = 0.0 for latitude/longitude, "" for description

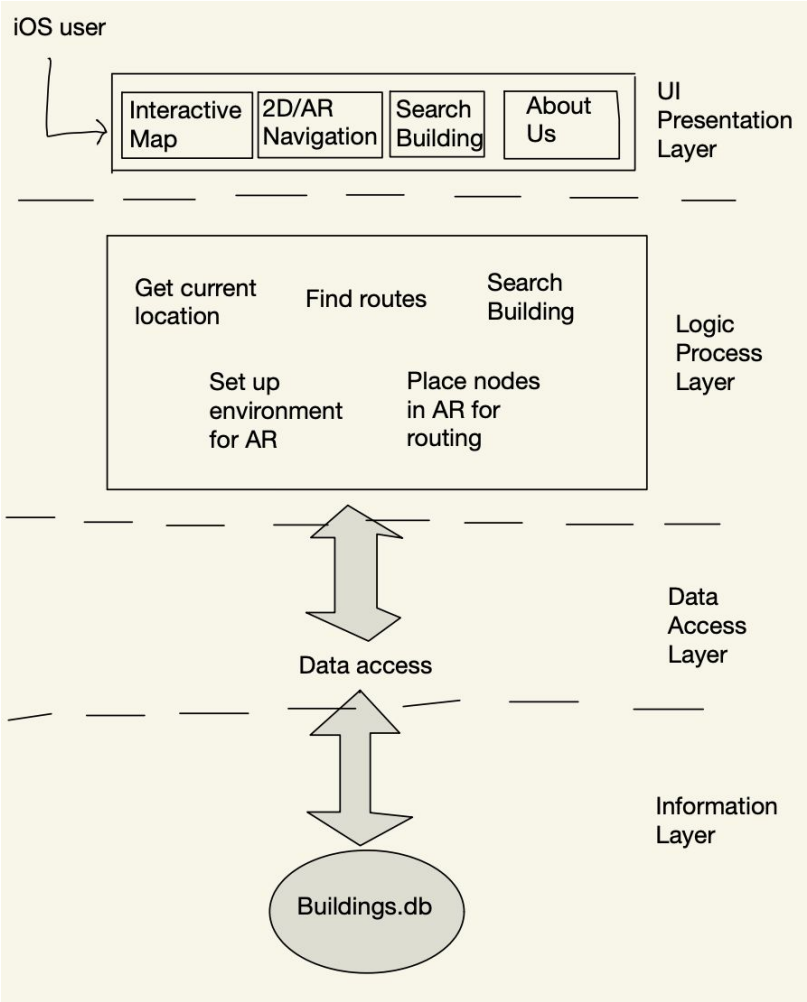
Post Condition

- We receive correct values for our queries

Tests: NavSBTests.swift

https://trello.com/c/SKmKnEL2/74-use-case-10-c-sql-database

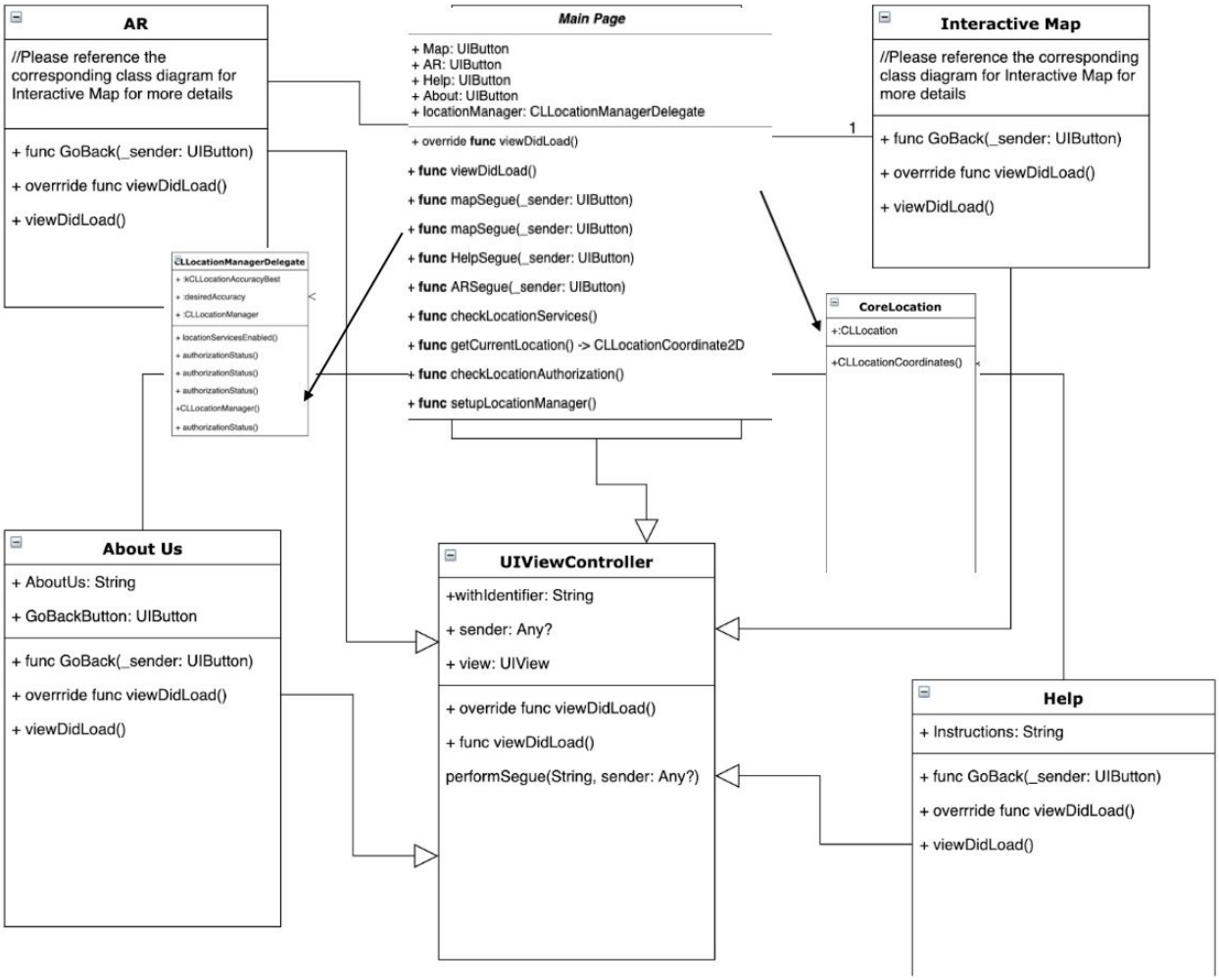
Architecture Design



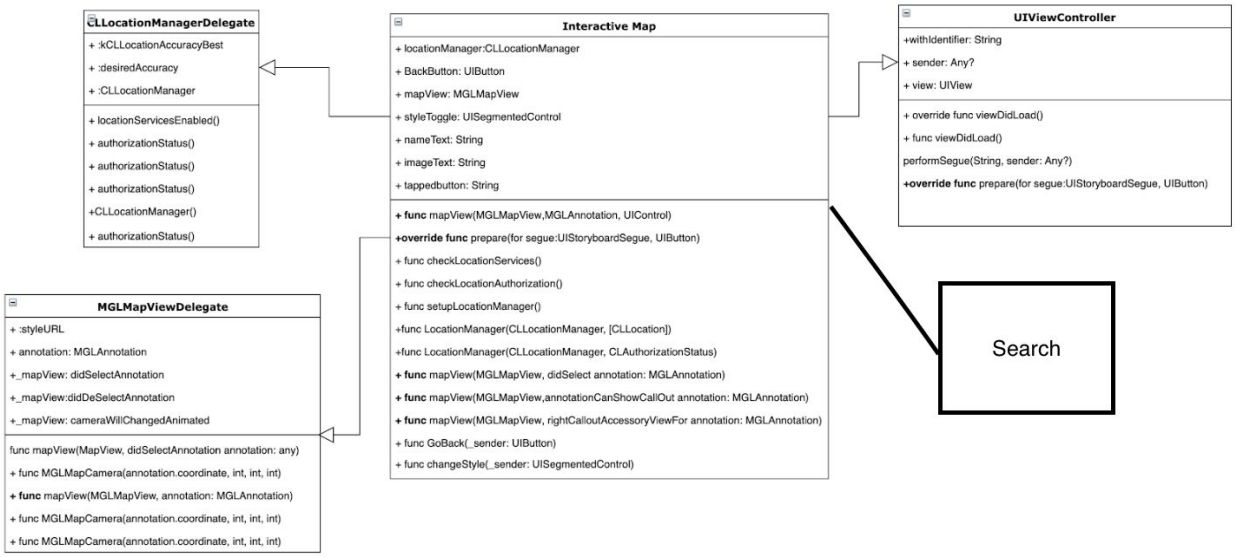
Our system architecture consists of UI presentation layer, Logic process layer, Data Access Layer, and Information Layer. UI presentation layer has all the user interfaces, such as Interactive Map, Navigation in 2D/AR, Query address, about us, and user history. Logic process layer deals with all the logic from user interfaces like getting current location of the user, finding routes to the destination, etc. Data Access Layer deals with accessing the data from the database from Information layer. Our data will mainly be the coordinates and descriptions of the buildings on our campus and possibly the user information if we decide to implement accounts for our app.

UML/Class Diagrams

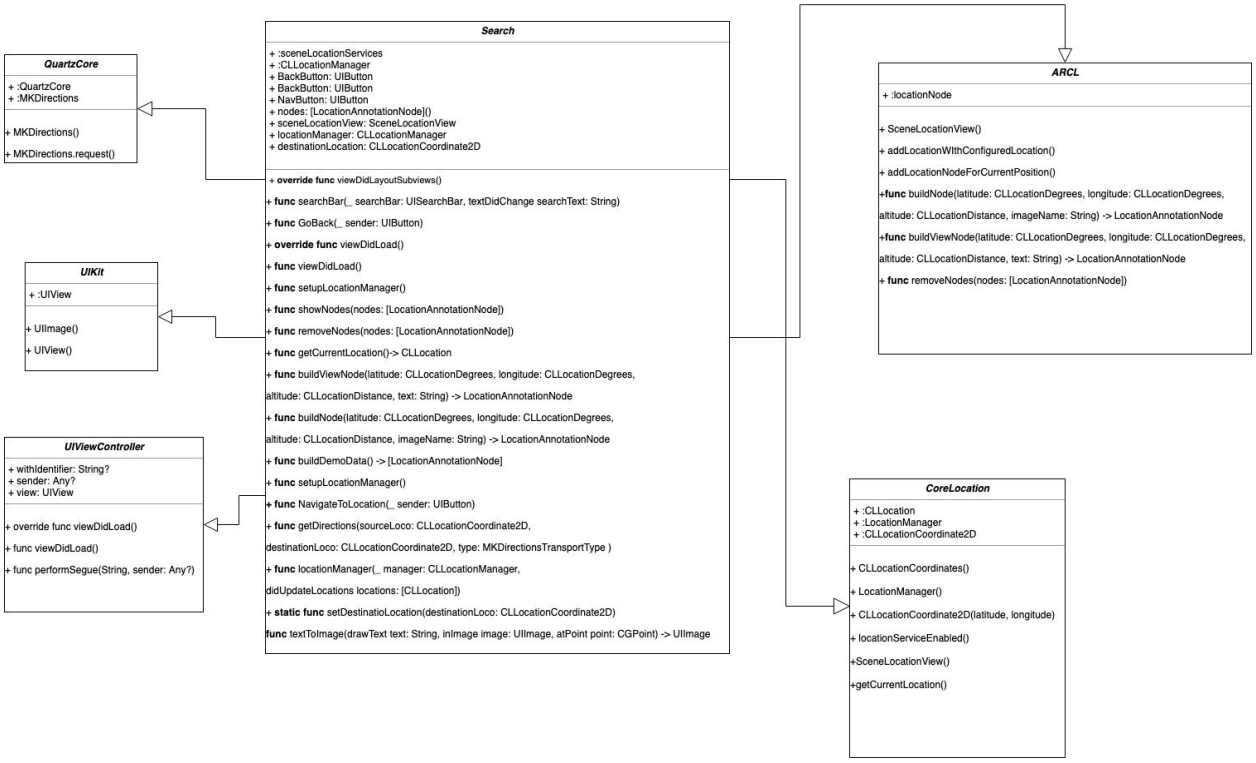
Main Screen:



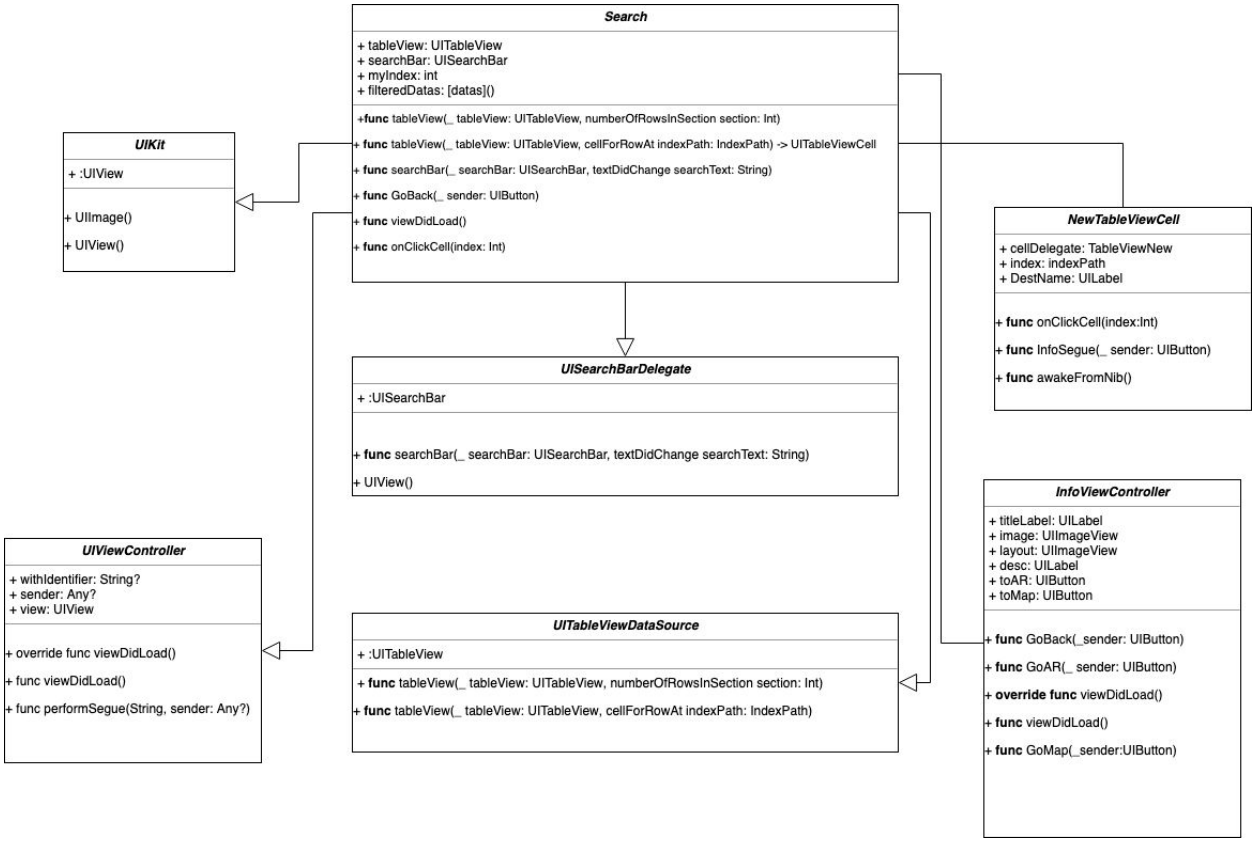
Interactive 2D Map:



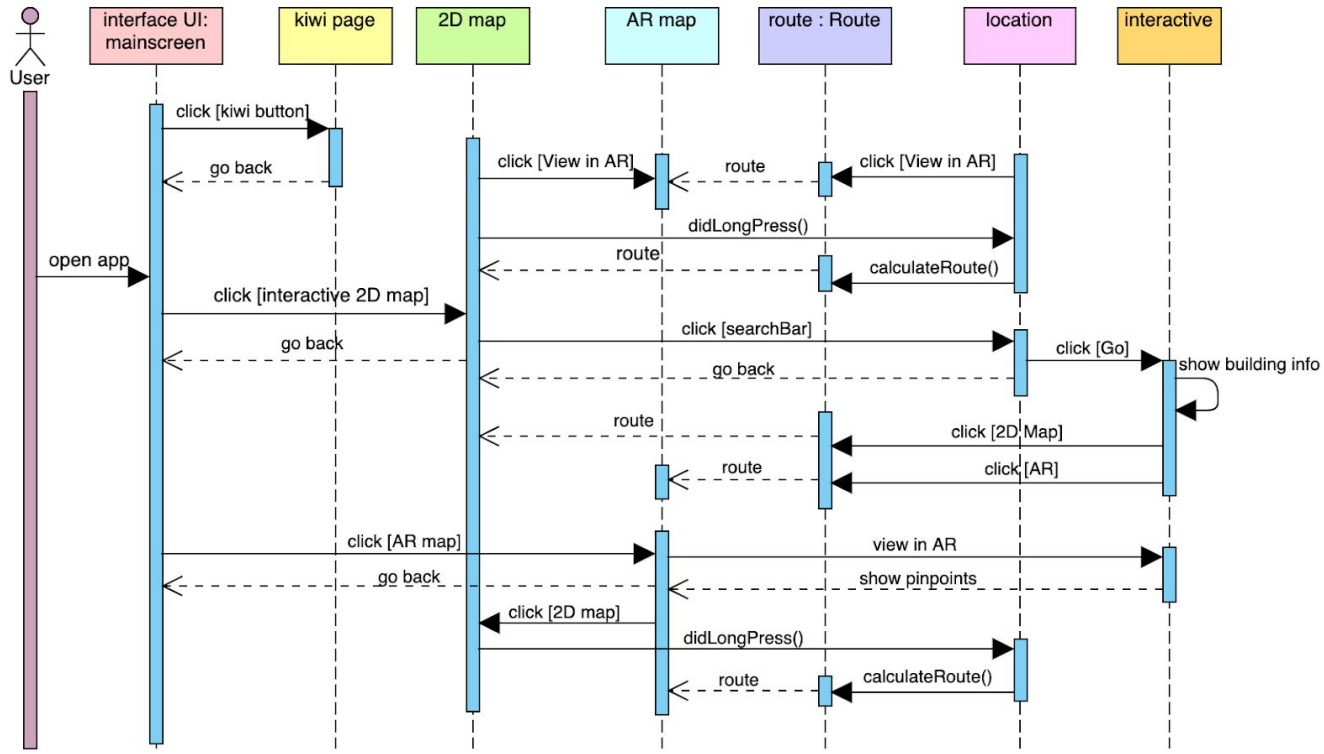
AR:



Search:


Search
+ tableView: UITableView
+ searchBar: UISearchBar
+ myIndex: int
+ filteredDatas: [datas]()
+func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int)
+ func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
+ func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String)
+ func GoBack(_ sender: UIButton)
+ func viewDidLoad()
+ func onClickCell(index: Int)
UIKit
+ :UIView
+ UIImage()
+ UIView()
NewTableViewCell
+ cellDelegate: TableViewNew
+ index: indexPath
+ DestName: UILabel
+ func onClickCell(index:Int)
+ func InfoSegue(_ sender: UIButton)
+ func awakeFromNib()
UISearchBarDelegate
+ :UISearchBar
+ func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String)
+ UIView()
InfoViewController
+ titleLabel: UILabel
+ image: UIImageView
+ layout: UIImageView
+ desc: UILabel
+ toAR: UIButton
+ toMap: UIButton
+ func GoBack(_sender: UIButton)
+ func GoAR(_ sender: UIButton)
+ override func viewDidLoad()
+ func viewDidLoad()
+ func GoMap(_sender:UIButton)
UIViewController
+ withIdentifier: String?
+ sender: Any?
+ view: UIView
+ override func viewDidLoad()
+ func viewDidLoad()
+ func performSegue(String, sender: Any?)
UITableViewDataSource
+ :UITableView
+ func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int)
+ func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)

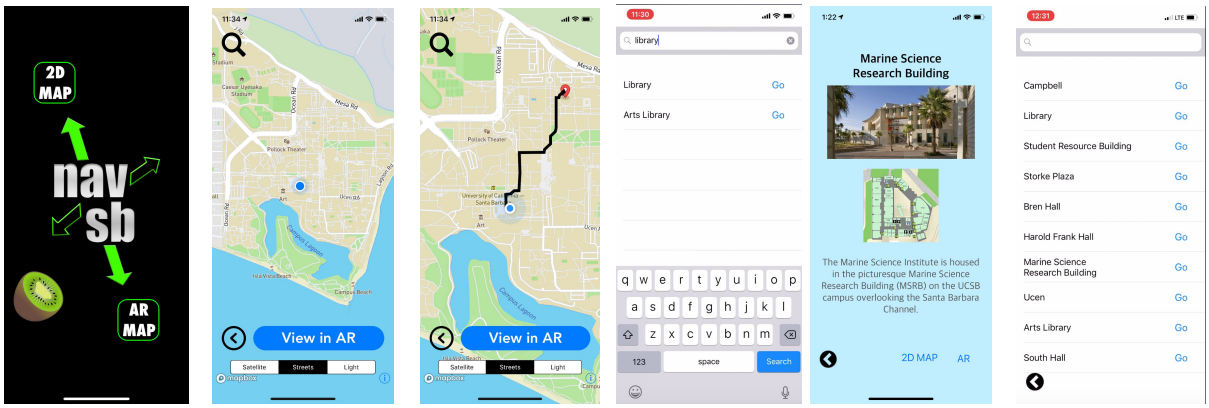
System Sequence Diagram



Working with the interactive 2D map:

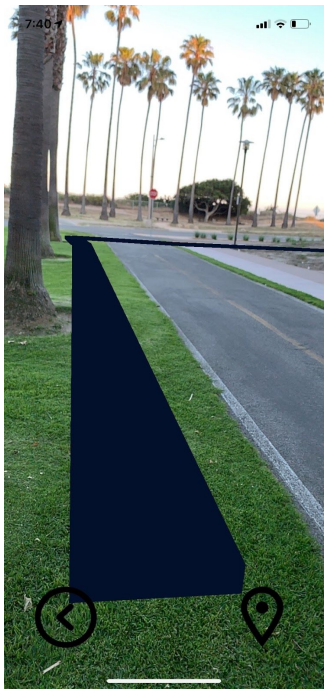
User has the option to choose 2D map option on the main screen of our navSB app. The user can click the search bar to search for their desired destination to get a detailed view of the building they search and get a route on the 2D map or AR map. The detailed view of UCSB buildings include a more interactive experience and more info on that location (ex: Phelps Hall, info button will show more information and a picture of Phelps Hall). This is represented by the “show building info” message on the interactive lifeline in the diagram above. When the user long presses anywhere on the map, they can get the route on the 2D map or they can click the “View in AR” button to get the route for the AR map. To go back to the 2D map from the interactive part and back to the main screen from the 2D map, user should click the back arrows. To switch to AR from 2D map, user presses the “View in AR” button.

Mockup for Main screen and 2D interactive map below:



Working with the AR Feature:

User also has the option to choose AR map option on the main screen of our navSB app. After clicking the AR map button on the main screen, AR will show indicators such as pinpoints when passing known landmarks on the UCSB campus. This is shown on the interactive lifeline in the diagram above as well. User can start directions in AR by clicking the 2D map button on AR screen, which will direct the user to the 2D map where they can long press for a destination or search for one. After the location is chosen, user should press the “View in AR” button for the route to appear in AR. Once the route is calculated, the AR feature will show a line in the direction of the specified destination.



Testing

Testing Files:

- NavSBTests/ NavSBTests.swift
- NavSBUITests/ NavSBUITests.swift

Functions Covered:

- UI Tests:
    - testAboutUs()
    - testMain()
- Tests (Database)
    - testGetLatitude()
    - testGetLongitude()
    - testGetDesc()
    - testGetLonEdgeCase()
    - testGetLatEdgeCase()
    - testGetDescEdgeCase()

We were only able to make UI tests for our main screen and about us page. We tried to make UI tests for the 2D map screen and AR screen, but a bug in xcode prevented us from completing this. For some reason it doesn't work with the mapbox and ARkit libraries (please ask Scott Chow for more details). The UI test for the main screen checks that all the buttons are there, to see if it is actually on the main screen. The UI test for our kiwi (about us) page tests the kiwi button on the main screen by checking the buttons and labels on the screen. It also tests the go back button by checking if it directs back to the main screen (with similar calls to the first UI test).

For our database, which is implemented using c++, we have a test suite that tests various normal as well as edge cases. We make sure that the getLatitudeCPP(), getLongitudeCPP() and getDesc() functions are working by calling them and making sure we are getting the correct return outputs. We test the edge cases by passing in invalid inputs (building identifiers that do not exist) and making sure they also return the correct "null" values (0.0 for getLongitudeCPP() and getLatitudeCPP(), "" for getDesc()).

Github Commits

Alex:

- Jun 11, 2019, latest
- Jun 3, 2019, Merge remote-tracking branch 'refs/remotes/origin/master'
- Jun 3, 2019, main.storyboard
- Jun 3, 2019, main.storyboard update
- May 30, 2019, added nice UI and segue to AR
- May 30, 2019, info.plist update
- May 30, 2019, swift warnings
- May 29, 2019, merge conflicts
- May 29, 2019, miscellaneous
- May 22, 2019, changed buttons from master/ long press on interactive gives pin point,
- May 22, 2019, added back and navigation button to AR. also can press and hold on interactive map
- May 22, 2019, current location to music lib
- May 21, 2019, current location to music lib
- May 20, 2019, hard coded route
- May 20, 2019, Working AR
- May 20, 2019, Update README.md

Ben:

- May 17, 2019, Update README.md and initial commit
- May 18, 2019, Update podfile
- May 18, 2019, Ported assets for assets.xcassets
- May 18, 2019, Update README.md instructions and included its assets (pictures)
- May 20, 2019, Continue updating README.md
- May 20, 2019, Update map layout
- May 20, 2019, Added UISegmentedControl to select different map layouts, and added customCallouts
- May 22, 2019, Added search bar w/ segue function for description. Need to add route annotation option, and segue to AR
- May 22, 2019, merge conflict resolved
- May 22, 2019, added the back and nav buttons in AR
- May 22, 2019, merged Alex’s branch with mine
- May 23, 2019, Updated deployment target of ARCL and Pods to 12.0 instead of 9.0 to fix it can link properly
- May 23, 2019, configured styling and front-end
- May 23, 2019, worked with scott to fix merge and linker issues, and pushed it to master
- May 28, 2019, touched up InfoView
- May 28, 2019, added route for interactive map
- May 31, 2019, removed stub ucsb annotation

- June 2, 2019, created Database.swift file to store our information
- June 10, 2019, fixed location issues that made it crash
- June 12, 2019, touchup and resolve all merge conflicts

Dawit:

- June 12: Added edge case testing
- June 12: Added testing for the database
- June 3: Merge ‘master’; pre merge
- May 29: additional changes
- May 28: changed interactive map storyboard
- May 28: changed label coloring
- May 28: added adjustments to locationNodes
- May 28: added removeNodes()
- May 28: added getCurrentLocation() to simplify stuff
- May 23: merging
- May 23: minor changes
- May 22: fixed buildNode() redundancy in AR for different …
- May 22: added functionality to call showNodes() everytime location is updated
- May 22: added CustomPointAnnotation class
- May 22: added function to makeAnnotations()

Joon:

- May 7, 2019: 2D map implemented & pinpoints to pre-set buildings added to mapview
- May 7, 2019: Pinpoints
- May 10, 2019: Create .gitignore
- May 14, 2019: Update .gitignore
- May 28, 2019: Mapview is centered at user's current location
- May 28, 2019: annotations reset when longpressed more than once
- May 29, 2019: Annotations reset when new longpress occurs
- May 30, 2019: 2D route button directs to interactive map and route is drawn automat…
- June 6, 2019: Database testing
- June 6, 2019: Updated database and data in database is accessible except the descri…
- June 9, 2019: IT WORKSS
- June 12, 2019: path is in variable now
- June 12, 2019: Database.swift calls C++ functions to get coordinates
- June 12, 2019: getting description from database
- June 12, 2019: database func error handled

Kalyn:

- May 22, 2019: Added NavSB icon
- May 22, 2019: Added routes to interactive map
- May 22, 2019: Changed loading screen, main screen, and buttons - frontend styling
- May 23, 2019: Added kiwi button and edited routes
- May 23, 2019: Resolved merge conflicts
- May 23, 2019: Thanks Scott x2
- May 27, 2019: edited about us page
- May 27, 2019: added team kiwi profile pictures to about us page
- May 27, 2019: finished about us page
- May 28, 2019: added button for interactive map
- May 30, 2019: added routes from go button on interactive map (Alex helped me with this one)
- June 2, 2019: added pictures on interactive map
- June 3, 2019: edits on info view controller
- June 8, 2019: added UI tests for main screen and kiwi (about us) page
- June 10, 2019: added UI tests for main screen and about us page

Appendix:

Xcode:

- IDE for iOS application development

Swift:

- Apple's native programming language

C++:

- Database to collect and store information on buildings
- Implemented SQLite as well with C++ features

Location Services:

- Use of Core Location to deal with real-world locations, coordinates, and current location

Mapbox Library:

- Use of Mapbox third party library to implement the interactive 2D map
    - Implements navigational and directional features
    - Implements 2D map interface and significant locations
    - Implements turn by turn navigation

ARCL:

- Use of ARKit library to implement camera and motion data as you move around
- Use of CoreLocation to use wifi and GPS data to determine your global location
- ARCL library combines high accuracy of AR with the scale of GPS data