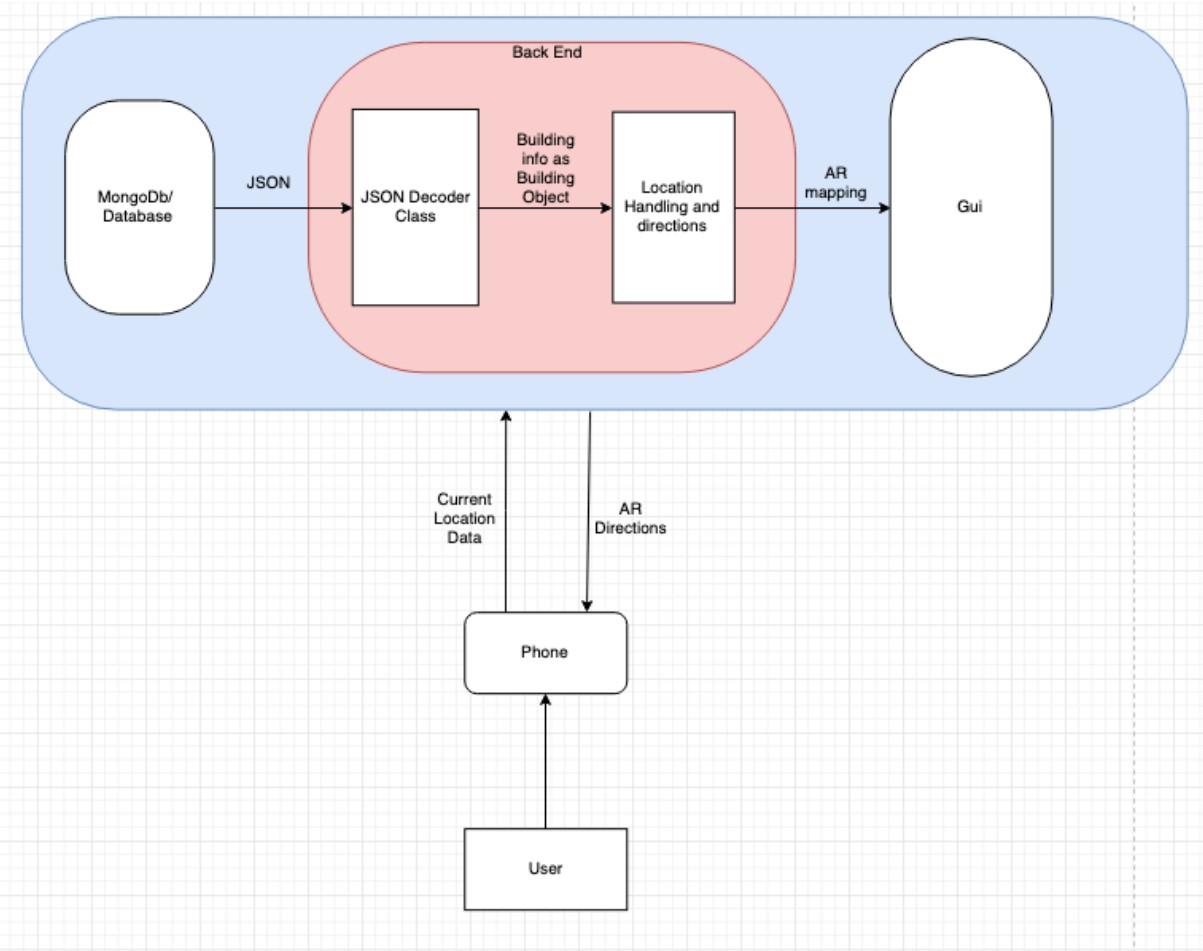
System Architecture



Description of overall architecture

Our App consists of a Firebase database, a database communication class using JSON, classes for handling the backend calculations for the Augmented Reality, and a GUI that has both normal UI and Augmented Reality. The overall system runs on any iOS device that supports AR.

User Story connections

- Stories 000, 001, 003, 004, 008 are handled by both the Augmented Reality UI and the supporting backend classes that filter data and perform calculations.
- Stories 005 and 006 are handled by the search bar UI, which will provide a list of favorites and suggested auto-completions.

- Stories 002 and 007 are handled by The AR UI and the backend class that searches data, pulls the info data and displays onto the UI.
- Story 009 is handled by the Location Manger class which finds the users current location
- Stories 010 and 011 are handled by XCode which allows the integration of pictures using their settings

Data Storage and Handling

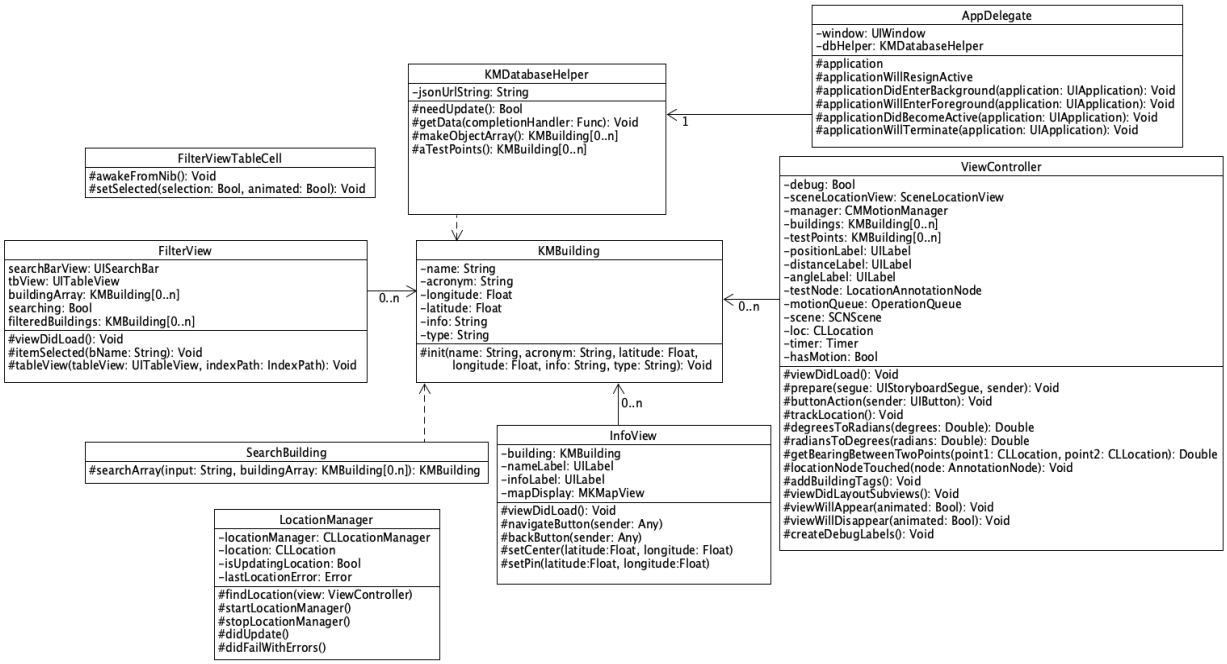
The data for the app is stored remotely in the database as JSON and accessed when the application launches.

Augmented Reality and Core Location

This app relies on the open-source, MIT-Licensed Cocoapod named ARCL. It handles fixing Augmented Reality tags to specific GPS locations in the real world. ARKit is the built-in iOS library that will be drawing objects to the screen while Core Location is the built-in iOS library that allows us to easily find and track the device.

The GPS coordinates will be stored in a database that is linked in to the app, rather than storing all locations locally on the user's device. The database will hold the name of the location, the acronym for the location, the location's latitude and longitude, and other information such as hours of operation and a small description. This information will be pulled by the app and used to populate the AR tags floating above each building.

Class Diagrams



Major Classes

FilterView

Handles building search bar. Filters and displayed the building data based on user input.

Addresses requirement 014, which relates to user story 001.

FilterViewTableCell

Placeholder class for additional functionality.

KMDatabaseHelper

Handles the database containing application data. Data is stored as JSON in a Google Firebase database and accessed using a REST API. Sample data is also provided for testing purposes.

Addresses requirement 007.

KMBuilding
Object representing a building. Contains building data: name, acronym, latitude, longitude, info, and type.

Helps address requirement 007.

SearchBuildings
Given a name or acronym, retrieves the corresponding KMBuilding from a [KMBuilding]. Acronyms are assumed to be less than four characters, while names are assumed to be four or more characters.

Addresses requirement 001, which relates to user story 006.

LocationManager
Begins tracking user location after asking for permission if necessary.

Addresses requirement 009.

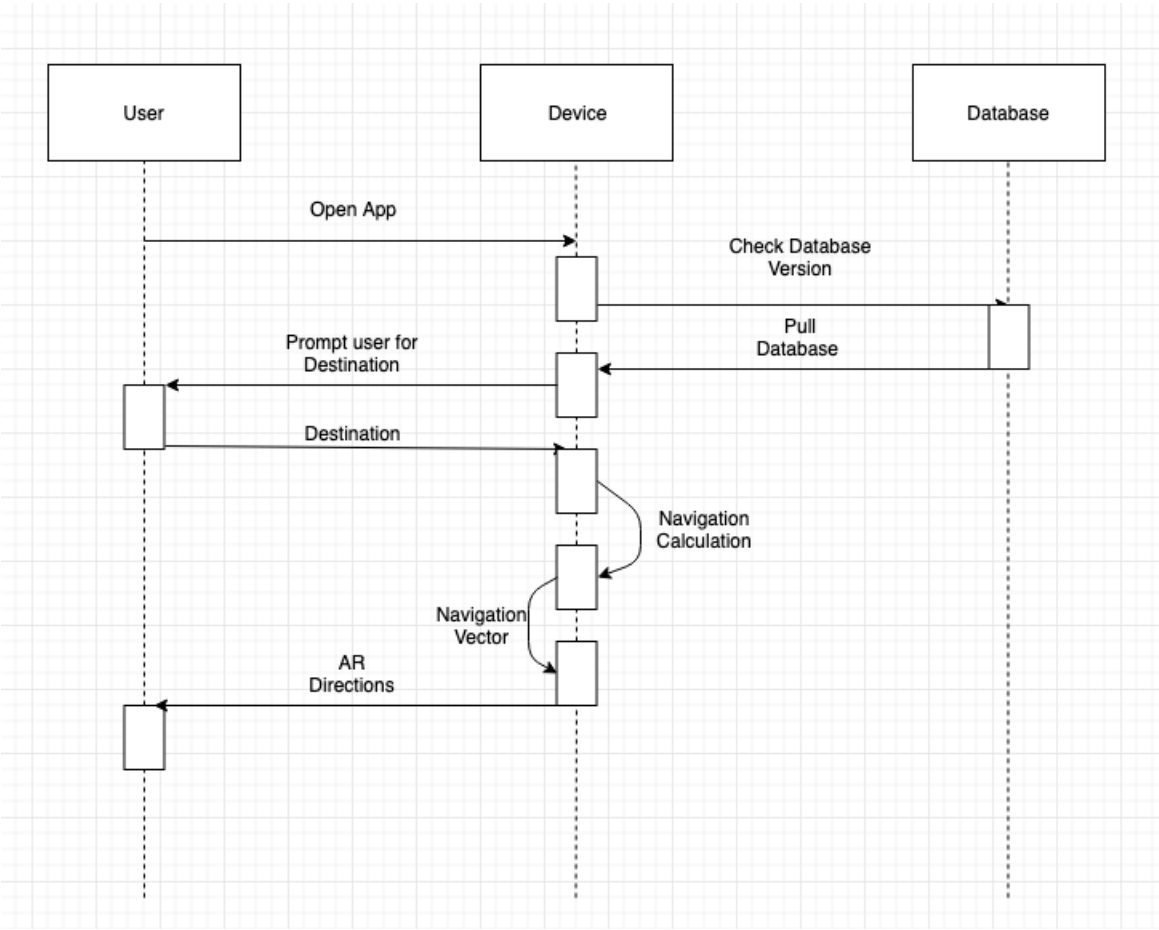
AppDelegate
Handles application launch options and initial configuration. Configures connection with Firebase.

Helps address requirement 007.

ViewController
Handles UI views and miscellaneous app logic.
- Loads data and sets up UI after loading. Initializes UI components, loads building data, and begins location tracking.
- Contains utility functions to track location, calculate direction of coordinate vector, convert between radians and degrees, and add tags to buildings.

Addresses requirement 005, which relates to user story 005.

Sequence Diagram



When the user opens the app, the app runs the viewcontroller.swift viewDidLoad() function which will run the database version check to test if the local data matches the hosted data. If the local database needs an update the app will run the function getData() to pull data in the form of JSON from the database and store into a local array of Building Objects after using the parseJSON function.

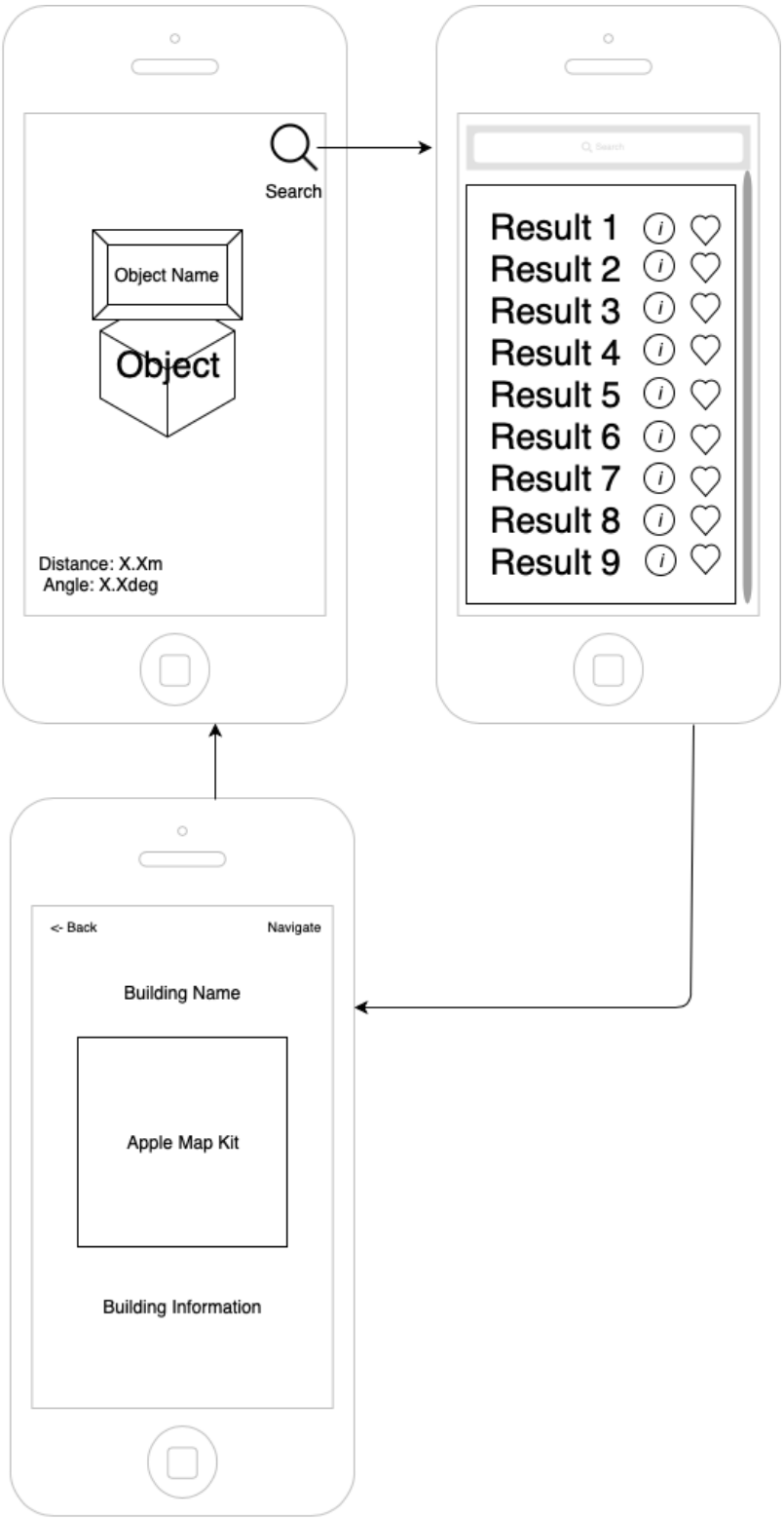
After the data check, the app will load the GUI where a user can type in a location on campus to travel to or they can click a location to see information about the building. Once the user types in and hits enter, the app will call a function that searches the local storage for the building name or acronym, if it is not found the app will notify the user in the GUI, otherwise if it is, the function will return

and another function uses the coordinate data from the object to calculate the path for the user.

Once the navigation data is calculated, the AR function draws the arrow onto the GUI and provides the user with directions to travel. As the user navigates, their location is periodically updated on the device and the navigation data is recalculated until the destination is reached

Once the destination is reached, the AR function will draw the name of the building on screen so the user can confirm they reached their location. At the destination they will have another chance to view information on the building by clicking an onscreen element that will display the information.

UI Diagram


Search
Object Name
Object
Distance: X.Xm
Angle: X.Xdeg
Result 1
Result 2
Result 3
Result 4
Result 5
Result 6
Result 7
Result 8
Result 9
<- Back
Navigate
Building Name
Apple Map Kit
Building Information

Main Screen

The main screen displays the camera input with nametags overlayed above valid objects. The upper right corner contains a search icon which, when clicked, opens the search bar. The lower left corner contains the distance and direction to the destination.

This relates to requirements 002, 005, 006 which correspond to user stories 003, 002, 000.

Search bar

The lower and center portions of the screen contain a list of search results. When clicked, the result is selected, and the app returns to the main screen. Each entry in the search results also has an info button, the button with the i in the circle, and a favorites button, the heart.

When the info button is taped the user is taken to the info page for that building.

When the favorites button is taped the object is added to the user's favorites list. This is able to be viewed by taping the "Favorites" option in the nav bar. When the button is pressed again it removes the object from favorites.

The upper left and center portion of the screen contains a search bar, the contents of which filter the below search results. The upper right portion of the screen contains the word 'cancel', which returns to the main screen without changing the selection.

This relates to requirements 001, 014, 003 which correspond to user stories 001, 006, and 005

Info Screen

The Info Screen displays the building name, information, and location on an apple maps screen when the user taps on the info button in the cell on the

search screen. The Upper left has a back button that returns the user back to the search screen and the upper right has a “Navigate” button that when pressed returns the user to the AR screen with new directions to the building they were viewing.

This relates to requirements 005, 006, 013 which correspond to user stories 005, 006, 007.