Before 02/02/2026

- Downloaded Docker
- Downloaded WinSCP
- Downloaded PuTTy
- Started to research how to develop with React and the syntax. (W3 Schools)
- I looked at Spencer's website from last year to see some of his process and his advice for other people with a similar project. I looked at the different applications he used. I was surprised to see that a lot of it I'm familiar with. I know of Node.js, and MongoDB from a class I took last year dealing with databases, and just from personal projects I've done. I enjoyed those applications and am looking forward to using them again if needed. He used Swift which I will have to be mindful of since I will be using React to code my application.
- Started to figure out what I'll do for the website. WordPress or HTML?
    - Started a templated HTML website
        - How do I make this look like mine?
        - How do I make this look like it's for my project and professional without it taking up too much time? (e.g. taking time away from working on the actual capstone project.)
    - Able to move the finished WordPress website, but what about in between then?
        - Again, with using a template, how do I make it cool, professional, and related to the project?

02/02/2026 - ≈1 hour

- Started this document
- Followed this website to set up React environment on my windows pc: https://code.visualstudio.com/docs/nodejs/reactjs-tutorial
    - This included doing a "Hello, world!" project with it.
    - Using VS code, NPM, Node.js
- I started to research React's syntax and documentation using W3 schools. It uses Vite instead of VS code. Is a bit different in terms of files and some code.
- Found out how to connect to the app on my phone. The Wi-Fi must be set to Private from properties (using PC wifi settings). Both devices must be connected to the same network. The phone connects using the IP address of the computer. Today (most likely always the same?) it was http://192.168.12.147:3000, where 3000 is the localhost address and 192.168.12.147 is the IP address. Phone and Computer both use browsers to look at the application (for me, Firefox).
- Next steps, I need to know what an initial UI will look like for the base application. I will sketch some drafts for as many screens as this will need, and then start to code a screen. The thought is that I am unable to code anything until I know *what* I'll be coding, so I need to think this through a bit.

- I am thinking I will build an application to navigate through GMS and split it into the different sections: Geology, Physics, Psychology, Biology, etc. I'll base the clues off of where those locations will be. If there's ideas for a single room implementation, I'm all ears because I have no idea where to even begin for that.

02/09/2026 - ≈4 hours

- Lesson learned – Take a break!
- Unfortunately, didn't document right after, but the blog is up under week 2.

02/17/2026 - Intermittently throughout the day

- Started to Implement a Expo Development Build app instead of an Expo Go app.
- I used the Android devices for this because it was easier to obtain Developer Options, basically no effort. Whereas Apple I signed up to be a developer on the software, but there's procedures in place, so I stopped trying until I get approval from Dr. McVey and Dr. Meyer.
- Because I thought that the issue was in Expo Go I tried following this website again: https://expo.dev/blog/how-to-build-a-bluetooth-low-energy-powered-expo-app
- Had I actually read the article correctly the first time, I would have seen that it says Expo Go cannot handle BLE.
- This didn't run and produced errors.
- I was able to get Elliot's project up and running after following his documentation.

02/18/2026

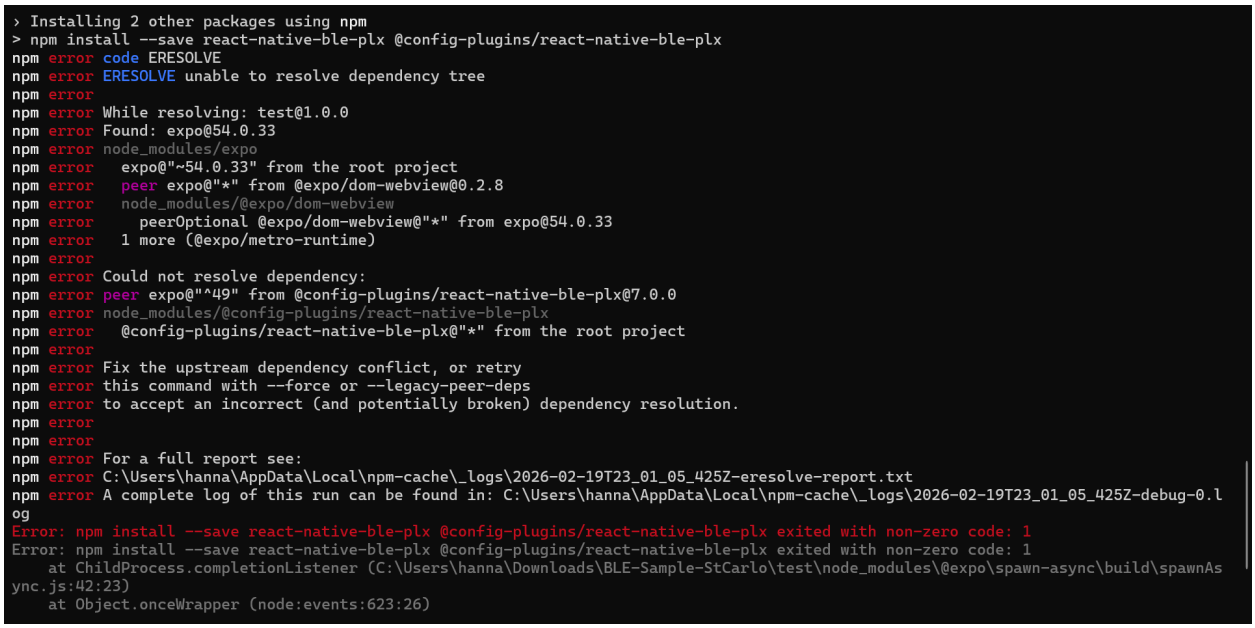
- Met with Dr. McVey and Dr. Meyer. We went through some Expo environment things with the development build. We also weren't able to get anything to run from the two applications I found from the author of the blog above.
- I am switching gears to use Android Studio and bare React Native instead of trying to do this within a framework.

02/19/2026

- Met with Dr. McVey today. We sat down and looked at this video here: https://www.youtube.com/watch?v=UuHLPsjp6fM
- We followed through with the code he had on Github as well as making an entirely new project from scratch using the video. Both gave us deprecated version errors.
- We were able to successfully compile and run Elliot's 2024 capstone project, but were unable to successfully connect to a Beacon using it. The scan is finding different bluetooth objects but cannot find any of the Beacons even though they are fully charged, on, and close to the phone. It errors out with the log saying that the name cannot be null.
- At this point, I am switching to using bare React Native with Android Studio, since we have had success compiling in that environment. However, I am going to focus on

creating the UI of the app rather than the technical details. This then can be put into the final application once the BLE issues are somehow figured out.

- Image of one of the errors from the blog code:



02/25/2026

- I started to modify Elliot's project to get a feel for React Native code and I started to build the UI using his style for now. I will go back and modify the style later.
- I was able to connect the android phone to my computer and was able to get from the Home screen to the Clues screen.
- The Home screen will have a "Start Game" button, and when a user clicks on that button it will lead to the Clues screen.
- The Clues screen will have the clue text, two debug texts (for me, for now) these will eventually be the UUID/MAC address of the Beacon and the GPS Coordinates of where the beacon is, distance to the clue, and a "Found it!" button that will act as a stand in for the background activity for finding the beacon I need.
- Nothing is dynamic quite yet, but it is a starting point finally!
- I met with Dr. McVey today as well, and we talked about using an array of clues (with beacon, gps coordinate, and text information) within the application at first then moving it to be a file somewhere on CompSci04 later. The app should load this info from the server once.
    - Helper App -->Pushes data to server, Main App -->Pulls data from server.
    - Build application replace background bluetooth process with a button for now, will code bluetooth in later once issues are hashed out.

    - We talked about sequencing the application, getting the "clue" screen built, tap button once you "found" the beacon, and then grabbing the next clue. Once this is done, then I can either do GPS or put the data in CompSci04 and try that.

02/26/2026

- I finally was able to scan for Beacons with Elliot's project! Turns out the code was great, but the hardware was not. I tried running his project with another phone, and it scanned and found all the beacons I turned on!
- I started to implement an array for the clues. I was looking at the React Native documentation for this, and they seem quite a bit different than what I am used to from C# and C++.

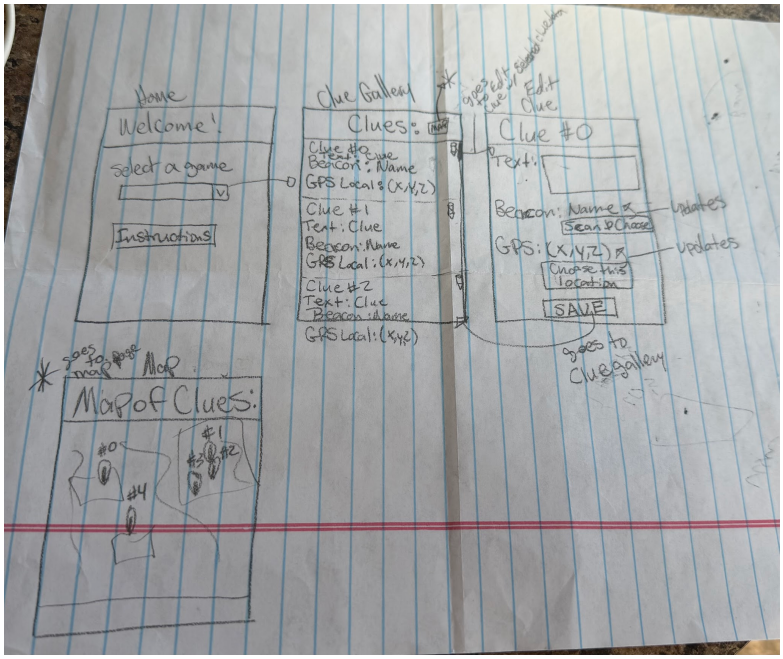
02/27/2026

- Today, I was able to make a lot of physical progress. I created the clue screen with more detail, and added some clues to an array. I added this array, and then added code in order to loop through it and display the whole array.
    - Button when clicked updated the count var that then grabbed the corresponding clue and displayed it.
    - Made a scan button - Scans and shows only Beacons
    - Made a stop scan button - Stops the scan manually for now
- One thing that I find difficult with working in React Native is that it seems different than anything I've ever coded in. For example, in the return() portion of the page, in order to comment I can't use the standard // or /*Comment*/. I must use {/*Comment Here*/} in order to comment. I find this quite annoying as I test and debug because it slows me down as I try to rearrange lines of code. I also think that the structure for some things is strange as well.

```
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -51
LOG  Calculated average distance for device BeaconL6: 0.5888436553555889
LOG  FOUND BEACON: BeaconL10 DD:34:02:06:74:49 -81
LOG  Calculated average distance for device BeaconL10: 7.7624711662869155
LOG  FOUND BEACON: BeaconL7 DD:34:02:06:74:17 -94
LOG  Calculated average distance for device BeaconL7: 52.480746024977286
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -52
LOG  Calculated average distance for device BeaconL6: 0.5888436553555889
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -59
LOG  Calculated average distance for device BeaconL6: 0.6025595860743579
LOG  FOUND BEACON: BeaconL10 DD:34:02:06:74:49 -73
LOG  Calculated average distance for device BeaconL10: 7.7624711662869155
LOG  FOUND BEACON: BeaconL10 DD:34:02:06:74:49 -80
LOG  Calculated average distance for device BeaconL10: 9.120108393559102
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -51
LOG  Calculated average distance for device BeaconL6: 0.6025595860743579
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -53
LOG  Calculated average distance for device BeaconL6: 0.512861383991365
LOG  FOUND BEACON: BeaconL10 DD:34:02:06:74:49 -79
LOG  Calculated average distance for device BeaconL10: 9.120108393559102
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -57
LOG  Calculated average distance for device BeaconL6: 0.5888436553555889
LOG  FOUND BEACON: BeaconL10 DD:34:02:06:74:49 -82
LOG  Calculated average distance for device BeaconL10: 10
LOG  FOUND BEACON: BeaconL7 DD:34:02:06:74:17 -93
LOG  Calculated average distance for device BeaconL7: 53.7031796370252444
LOG  FOUND BEACON: BeaconL10 DD:34:02:06:74:49 -70
LOG  Calculated average distance for device BeaconL10: 7.7624711662869155
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -56
LOG  Calculated average distance for device BeaconL6: 0.64565422903465557
LOG  FOUND BEACON: BeaconL6 DD:34:02:08:FC:88 -51
LOG  Calculated average distance for device BeaconL6: 0.5370317963702529
LOG  Scan stopped.
```

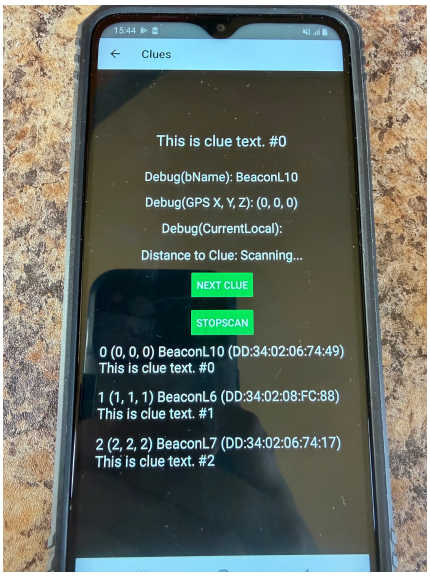
03/04/2026

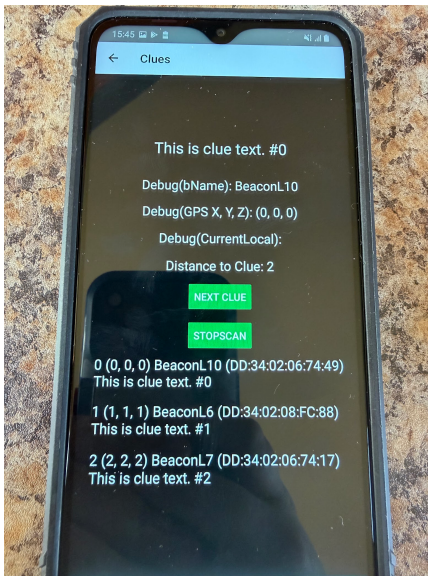
- I drafted up a UI for the helper app. I planned out how it would look and what buttons would go where. I have not started to put this into implementation yet, but once I have more information for clues and games, I will start to do so.



03/08/2026

- Today, I got a working distance to clue stat to run. I am able to see if I am getting closer or farther away from the beacon. I don't know if the math is completely accurate yet, since it tends to have outliers when the phone stays in place.
- I am also able to loop through each clue and only get that beacon's distance. This shows on the phone screen and shows in the console.log().
- The application just takes the RSSI of the beacon when it finds it in the scan. The phone never actually connects to the beacon.
- Example of scan and distance:





- Examples of some added code:

```
//Function onPress
//Parameters: None
//Will be changed to be an automatic process, but this allows the button to manually change to a different clue in the CluesArray
const onPress = () => {
  const nextClue = (count + 1) % cluesArray.length;
  setCount(nextClue);
  setBeaconDistance(0);
  stopScan();
  startScan(cluesArray[nextClue].bName);
};
```

This is a manual button to progress through the clues. This will eventually be an automatic process once a user gets close enough to a beacon it will activate.

```
//Function startScan
//Parameter: beacon: string beacon name (ex. BeaconL10)
//Scans for the beacon the parameter sends in. Does not have a timeout, keeps scanning until a new clue is selected or user backs out of clue page.
const startScan = async (beacon:string) => {
  //#region…
  var permissions = await requestBluetoothPermission;
  if(!permissions) return;

  console.log("Starting BLE scan for ",beacon," ...");

  manager.startDeviceScan(null, null, (error, device) => {
    if (error) {
      console.log("Scan error:", error);
      return;
    }

    if (device?.name?.includes(beacon)) {
      //console.log("RSSI:", device.rssi);
      if(device)
      {
        DistancetoClue(device);
      };
    }
  });
};
```

This function runs the scan of the bluetooth devices. It keeps scanning through every device in the area, when it comes to a device with the name of the beacon it's searching for then it calculates the distance between the phone and that beacon and keeps scanning.

```
//Function stopScan
//Parameters: none
//Stops the BLE scanning
const stopScan = async () => {
  manager.stopDeviceScan();
  console.log("Scan stopped.");
}
```

This stops the scan and tells the console that it stopped. This makes it easier to just call a function rather than writing these lines every time.

```
//Function updateRssiReadings
//Parameters: deviceId: string beacon name, newRssi: new RSSI number from beacon data
//Stores the RSSI with beacon info & previous data for calculating & updating
const updateRssiReadings = (deviceId: string, newRssi: number): void => {
  const readings = rssiReadingsByDeviceRef.current[deviceId] || [];
  readings.push(newRssi);
  if (readings.length > 10) {
    readings.shift();
  }

  rssiReadingsByDeviceRef.current[deviceId] = readings;
};
```

This stores an RSSI reading (how powerful the signal the beacon is giving off – less negative = closer and more negative = farther away) of a beacon into its own array. It only stores 10 readings per beacon at the moment.

```
//Function: calculateAverageRssi
//Parameters: deviceId: string
//Returns: Average RSSI over the past readings, Null if no data
//Note: Only keeping 5 readings per device, because want movement
//Using the array of beacon data, calculate average RSSI for a device
const calculateAverageRssi = (deviceId: string): number | null => {
  const readings = rssiReadingsByDeviceRef.current[deviceId] || [];
  if (readings.length === 0) {
    return null;
  }
  const sum = readings.reduce((acc, val) => acc + val, 0);
  console.log("Readings:", readings);
  return sum / readings.length;
};
```

This calculates the average RSSI reading from the above RSSI readings array. It sums all the RSSIs together and then divides by how many there were in the array and returns this value.

```
// Converting RSSI to distance
// Source: StackOverflow
const MeasuredPower = -59; // Measured power 1 meter from beacons
const pathLossExponent = 2.5; // (2-4) 2 for open area, 3 for indoors, 4 for GMS/rebar walls
const calculateDistance = (averageRssi: number): number => {
  if (averageRssi === 0) {
    return -1; // if we cannot determine RSSI, return -1
  }
  return Math.pow(10, (MeasuredPower - averageRssi) / (10 * pathLossExponent));
};
```

This is the function that actually converts the RSSI into a distance measured in meters.

```
//Function: DistancetoClue
//Parameters: device: Device
//Returns: Void
//Ties all the above functions together to find the distance in meters between the phone and beacon
const DistancetoClue = (device: Device) => {
  if(device.rssi != null){ //makes sure the rssi is not null/empty
    updateRssiReadings(device.id, device.rssi); // Update the array of RSSI readings
    const averageRssi = calculateAverageRssi(device.id); //get average of RSSIs
    const distance = averageRssi != null ? calculateDistance(averageRssi) : null; //convert rssi to meters
    console.log(rssiReadingsByDeviceRef.current[device.id].length); //output to console
    if(distance){ //if distance hold something
      if(rssiReadingsByDeviceRef.current[device.id].length == 10){ //if the current beacon's rssi readings array has a length of rssiReadingsSize then
        setBeaconDistance(Number(distance.toFixed(0))); //Set the variable BeaconDistance to the rounded distance
      }
      console.log(`Calculated average distance for device ${device.name}: ${distance}`); //output to the console
    }
  }
}
```

This function ties all the above functions together to find the distance between the phone and beacon. This runs in StartScan() once the targeted beacon is found.

03/10/2026

- I met with Dr. McVey today to show her the distance feature on the application. We walked through it, and it worked well. We discussed having a weighted average instead of a straight calculated average. This would put higher priority on the most recent scan rather than the oldest scan.

- We also discussed putting the Clues.json into a file in the project, which would eventually go to CompSci04 and just live there. However, to the surprise of both of us, we found out it doesn't take much to read a file from a server using React Native. We found a piece of code using Gemini (in search) that did exactly this! So we were able to get that functionality working as well.
- The next thing we discussed were next steps. At the present moment, connecting the read data from the JSON to an array to get my CluesArray back and running is my first priority. Next, I would like to get the user's current location to start setting up for GPS mapping and guidance when the beacon is simply too far to read. The third priority is to be able to read the file from CompSci04 and be able to write back to the server with the contents of the CluesArray but with a different name. This ensures that I am able to do this for my helper application when I implement that. The last thing I would like to do is redesign the application so that it fits the theme I am working with instead of using Elliot's design. This will make the app uniquely mine.

03/15/2026

- I was able to connect the JSON file from the server to the cluesArray! The function below gets the file from CompSci04 and resets the cache. I have to reset the cache at least in order to test because if I don't the function won't grab the new data from the server and just use what was there before. I set up the clues page to have a parameter be passed to it as well. This parameter is the array details from the file.
- I also started to work on trying to send back a file or create a new file to the server, but I couldn't get it to work yet. I will pivot for now until I can chat with Dr. McVey about it. I may have to use PHP instead in order to write the file to the server. For now, I will focus on the other tasks from earlier.

On the home page:

```
const getFileFromServer = async () =>{
  try {
    const response = await fetch(
      'https://compsci04.snc.edu/cs460/2026/hannahthiry/Clues.json?cacheBust=' + Date.now()
    ); //Copilot was used to "break" the stored cache.
    if (!response.ok) {
      throw new Error('Network request failed');
    }

    // To get a JSON object
    const fileJSONData = await response.json();
    console.log(fileJSONData);
    navigation.navigate("Clues", {fileData: fileJSONData});

  } catch (error) {
    console.error("There was an error fetching the file:", error);
  }
}
```

On the clues page:

```
type CluesRouteProp = RouteProp<RootStackParamList, 'Clues'>;
```

```
//gets JSON data from server and home page using route:
const route = useRoute<CluesRouteProp>();
const { fileData } = route.params;
const cluesArray = fileData;
//console.log("cluesArray: ",cluesArray);
```

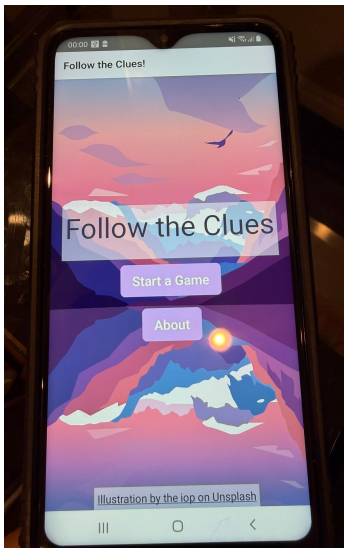
03/17/2026

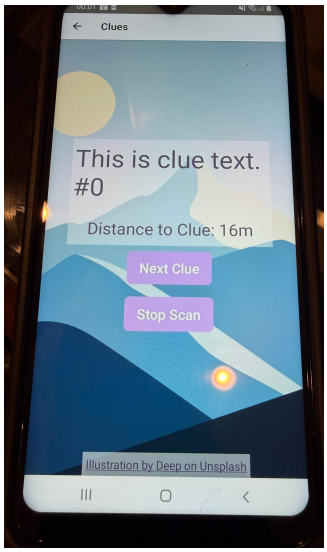
- I added the weighted average! I also tried to see what combination of numbers would allow for the least amount of error margin. So far I am finding that 10 readings, though lengthening the time taken for the initial scan, seems to be most accurate with the numbers jumping -5 or 5+ above where it likes to sit when the phone isn't being moved. It does seem that the weighted average shortens the time while a user is roaming around for the correct distance to show up. There is less misleading information with a weighted average.
- Here is an example of some code:

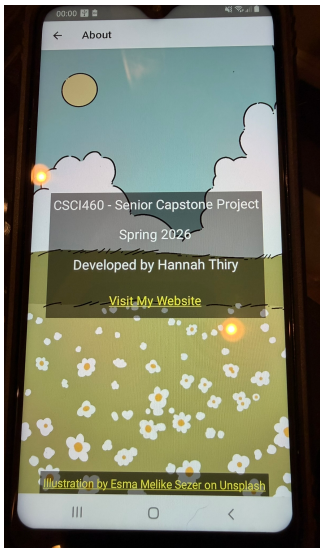
```
//Returns: Average RSSI over the past readings, Null if no data
//Note: Only keeping 5 readings per device, because want movement
//Using the array of beacon data, calculate average RSSI for a device
const calculateAverageRssi = (deviceId: string): number | null => {
  const readings = rssiReadingsByDeviceRef.current[deviceId] || [];
  if (readings.length === 0) {
    return null;
  }

  let sum = 0;
  for (let i = 0; i < readings.length; i++) {
    const currentValue = readings[i];
    //sum = sum + currentValue;

    //10 values:
      if(i>=0 && i<=3){
        sum = sum + currentValue * .05;
      }
      else if(i>=4 && i <=6){
        sum = sum + currentValue * .1;
      }
      else if(i>=7 && i<=9){
        sum = sum + currentValue * .15;
      }
    //6 values:
      // if(i>=0 && i<=1){
      //   sum = sum + currentValue * .125;
      // }
      // else if(i>=2 && i <=3){
      //   sum = sum + currentValue * .15;
      // }
      // else if(i>=4 && i<=5){
      //   sum = sum + currentValue * .225;
      // }
  }

  console.log("Readings:", readings);
  return sum;
};
```

03/19/2026

- Today my priority was to come up with a re-design for my application. I thought about what the design should look like since each game will be different due to the users creating each game through a helper application. Taking this into consideration, I wanted to give the vibe that one was going on an adventure or exploring somewhere. I found some nice pictures from Unsplash (with the free Unsplash license) that, to me, give off this theme quite well. The other thing I wanted to keep in mind was having a cohesive app that the color schemes go well together. I am not sure I did a great job at that yet, but this is just the first draft of a re-design, so it will likely change a few more times.
- I haven't done anything yet with font styling but with time that will also be changed to something more cohesive (but still readable!) to the theme. **Update: Font Styling was done the next day. It took a lot of time to figure out. The method I was using was in fact not automatic. So I've been manually updating the fonts within "/android/app/src/main/assets/fonts" and also deleting the file "/android/link-assets-manifest.json" then reruning the "npx react-native assets" command.
- If there's any suggestions for what to do, please let me know. It is hard for me to visualize in my head what this should look like going from a welcome screen to a more simple clues screen, and I am not an artist by any standards. This means I'm restricted to what's already out there that artists are willing to allow others to use. So if this current theme doesn't look good and needs some tweaks or a complete overhaul, please help me to fix it! I want it to look the best it can possibly be.
- I also learned about Pressables instead of using Buttons. For some reason, buttons are super limited styling wise in React Native (probably due to cross-platforming with Android and IOS), so I've switched to using Pressables for now in order to style them better. Eventually, I will not have the "Next Clue" or "Stop Scan" buttons on the Clues screen, but I need them for debugging yet. Below are the newly updated screens:







03/27/2026

- Today I was working on a few things that Dr. McVey and I talked about on Monday. I was able to change the scanning function to scan for all beacons, but only display the target beacon. This helps with loading time while the beacons are next to each other, but there's still a long delay when there aren't any prior scans.
- The other thing I've been trying to implement is a connect and scan for RSSIs. I was able to implement a scan and connect screen to one beacon. This worked well, but then when I tried to bring it back to my original project, I have been having issues with it.

Week of 03/29/2026:

- Check blog

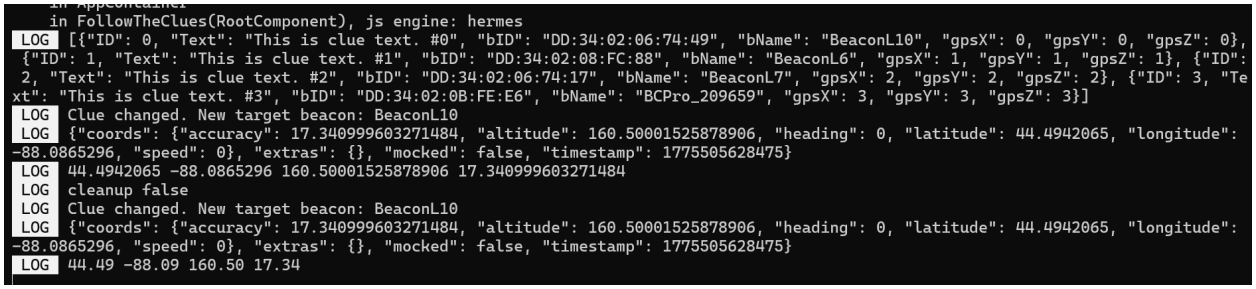
04/01/2026

- Meeting Recap with Dr. McVey:
- I demonstrated that I could connect to a target beacon, and there is now no need to have the scan function find other beacons while searching for the target beacon due to the fast RSSI readings when the target beacon is connected.
- I re-designed my app during Spring Break and implemented an 8 readings array with a weighted average. We thought that maybe having a 4 readings array weighted average before getting to the 8 might be helpful because of slow loading times, but with connecting that doesn't seem to be a problem now.
- While scanning, connecting, and disconnecting, the more the program ran the more times the scan function was called. Originally, we thought this was a recursive function, but it was actually due to the way the disconnect listener was set up. During the connection of a device, it was always setting up the disconnect listener, but apparently this is a function that only needed to be called once per device. So since it was setting up this function multiple times, it thought it needed to run Scan multiple times. I fixed this with a Ref variable that keeps track when the listener is running or not.
- The fix largely fixed the issue; however, when clicking through the clues really fast, it still runs multiple times and crashes the scanning function. Since the clues will eventually be automatic, and harder to find. It is likely this won't be an issue later on.

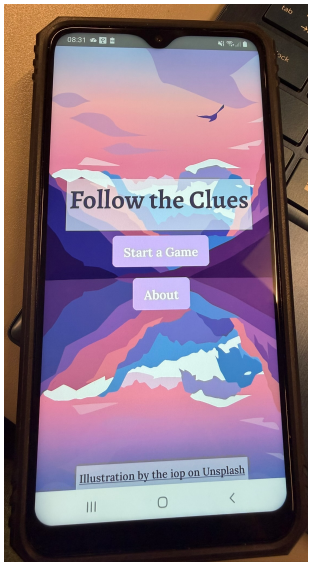
04/06/2026

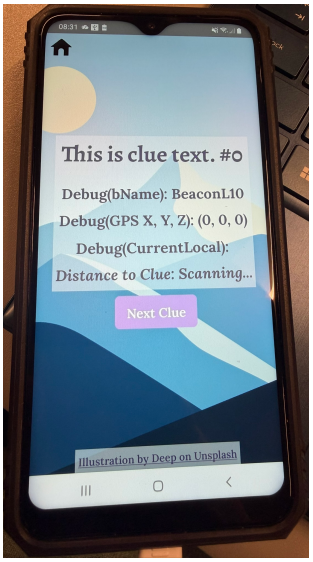
- I added a home button on the About and Clues page!
- I fixed up code from 4/1 and tested a lot to make sure I didn't have any other oddities. I did this for a bit and touched up my bluetooth section. Then I added a timeout to the scan function that stops the scan after 20 seconds of looking for the target beacon. This is a segway into the GPS functionality because if the scan cannot find the target beacon, then the GPS will take over.

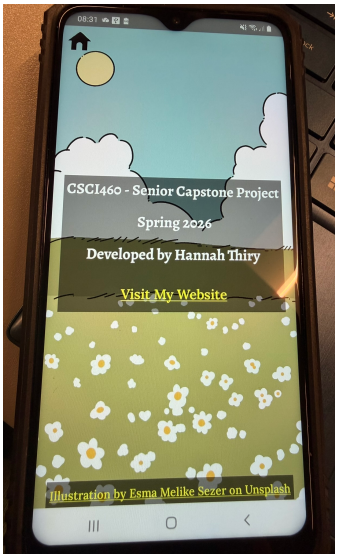
- I added geolocation to the project and was able to get the user's current location from it:

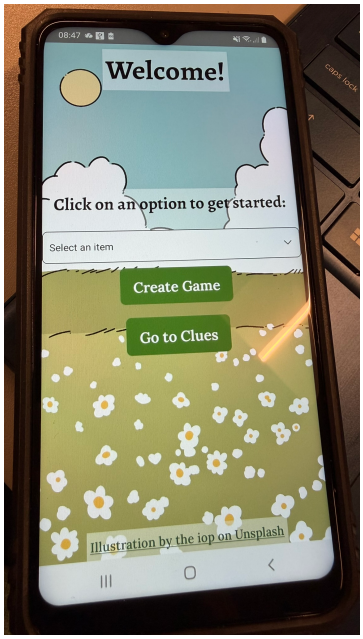


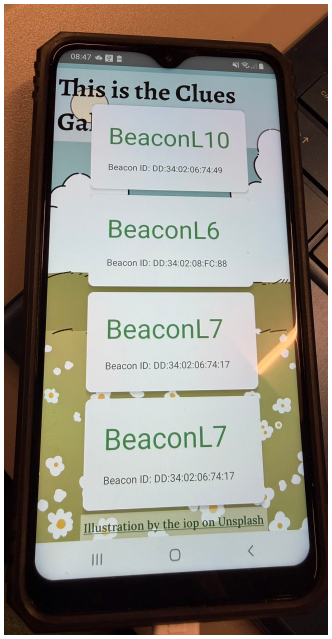
- The last line is in order: Latitude, Longitude, Altitude, and Accuracy fixed to the 2nd decimal place.
- I started the Admin application tonight and decided to go with the fields of flowers for now for the theme.
    - I started a new bare react native project and then started inputting some of the assets I used in the Main application.
    - For now, I have three working screens: Home, Create Game, and Clues Gallery. They don't do a lot right now as it's a lot of UI work at the moment.
    - Both Create Game and Clues Gallery are able to get passed the array from CompSci04.
    - I also created a dropdown, but currently do not have a file where all the games are kept yet.
    - The data structure of a game as of right now is an ID, Name, and an array of clue ids. I will also be adding a game id to the clues data structure.
    - The gallery idea I have for the Clues page will I think be difficult due to needing it to scroll and have information on all the clues. I wonder if I am thinking too PowerApps-y and wonder if there's a better way to display the information.
    - I used Copilot in order to get a start on the looks of the "cards" each clue will be displayed in. I also realized that I don't have an add clue button on my draft, so I have to figure out where I will want to put that.











04/09/2026

- I fixed up the design on the Admin app to be more like my drafts that I have. Create Game and Go to Clues look very similar.
- Design is quite hard in React Native, and I don't know how I am going to get each clue to have a button that is only tied to that clue, so that it can be edited.

04/13/2026

Capstone Meeting 4/13

Hannah Thiry <hannah.thiry1@snc.edu>
to Bonnie, Seth, Eric

* Updated on where I currently stand. User interface updates to the Admin app.
* Figured out together how to write to a file on CompSci04
* Needed: Beacon connection and GPS connection on Admin App.

04/20/2026

- I added functionality to delete clues from the game, and to delete all clues of a game. (only in create game screen right now)
- Added a confirmation modal based off of the device modal. One step for the user to not delete all clues on accident. Or one specific clue on accident. Will be useful later on for games as well.
- Logic for clues is almost the same for what deleting games will be.

04/21/2026

- I can delete games!
- Changed file structure to just use Games.json instead of Game#.json and Games.json.
    - Makes the process of deletion easier.
- I copied over the CreateGame screen structure to be the DisplayGame screen
- Organized, cleaned up, and commented code
- Show save button when there is at least 1 non space character and hide it when there isn't
- Found that GPS coordinates were more accurate at 4 decimal places than 2.

04/22/2026

- Notes for later or to save:

```
// let currentGames = []; //creates new Games array
// for(var i = 0; i < fileJSONData.current.length; i++){ //loops through the server data for games
//     if(i == gameID){ //if the looped id equals the game to be updated then
//         currentGames.push({
//          // ID: gameID,
//          // Name: currentGameName,
//          // Clues: cluesArray
//         }); //push the new game data instead of the old
//     }else{
//         currentGames.push(fileJSONData.current[i]); //otherwise just push the new data there
//     }
// }

const currentGames = [...fileJSONData.current];
currentGames[gameID] = {
    ...currentGames[gameID],
    ID: gameID,
    Name: currentGameName,
    Clues: cluesArray
};
console.log('currentGames:', currentGames);
```

```
BeaconL7  Readings:  ▸ (8) [-64, -62, -63, -63, -63, -63, -
Calculated average distance for device BeaconL7 : 4.130333
cleanup false
RSSI Loop Stopped
Beacon disconnected: DD:34:02:06:74:17 error: null
Beacon disconnected: DD:34:02:06:74:17 error: null
Beacon disconnected: DD:34:02:06:74:17 error: null
```

04/26/2026:

Notes:

```
// const startWatchingLocation = () => {
//     console.log("Watch Position");
//     watchIDRef.current = Geolocation.watchPosition(
//         position => {
//             const currentLocal: GPScoord = {
//                 X: Number(position.coords.latitude) ,
//                 Y: Number(position.coords.longitude),
//                 Z: position.coords.altitude ? Number(position.coords.altitude) : 0
//             };
//             console.log('Updated position:', currentLocal);
//             locationReadings.current.push(currentLocal);
//             if (locationReadings.current.length > readingsCount) {
//                 locationReadings.current.shift();
//             }
//             console.log('locationReadings', locationReadings.current);
//         },
//         error => {
//             console.log('Watch error:', error);
//         },
//         {
//             enableHighAccuracy: true,
//             distanceFilter: 1,   // meters before callback fires again
//             interval: 500,      // Android: update interval
//             fastestInterval: 500,
//         }
//     );
// };

// const stopWatchingLocation = () => {
//     console.log("Stop Watch:", watchIDRef.current);
//     if (watchIDRef.current != null) {
```

```
//         Geolocation.clearWatch(watchIDRef.current);
//         watchIDRef.current = null;
//     }
// };

// const calculateReadingsAverage = () => {
//     const count = locationReadings.current.length;
//     if(count == 0) return;

//     console.log("CalculateReadingsAverage");
//     let sumX = 0;
//     let sumY = 0;
//     let sumZ = 0;

//     locationReadings.current.forEach(element => {
//         if(element){
//             sumX += element.X;
//             sumY += element.Y;
//             sumZ += element.Z;
//         }
//     });

//     const NewGPSLocal: GPScoord = {
//         X: sumX / count,
//         Y: sumY / count,
//         Z: sumZ / count
//     };
//     console.log(NewGPSLocal);

//     return NewGPSLocal
// };

// const NewClueGPS = (id: number)=>{
//     const avgGPS = calculateReadingsAverage();
//     if(avgGPS){
//         const updated = [...cluesArray]; //update the clue
//         updated[id] = {
//             ...updated[id],
//             gpsX: avgGPS.X,
//             gpsY: avgGPS.Y,
//             gpsZ: avgGPS.Z,
//         };
//         setCluesArray(updated);
//     }
//     locationReadings.current = [];
// };
```

05/05/2026

- I haven't updated in awhile, but I went from not being able to run a game to running a game, presenting and now finalizing the applications.
- I added and finalized the comments in both applications
- I added a timer to the app, so it starts when the user sees the GPS feet distance initially and it pauses as a courtesy when the user switches to the new clue and gets the initial GPS distance there as well.
- I added a dropdown and cut config.json file functionality at the request of Dr. McVey. This allows for greater flexibility on the user's part in the main app.
- I also added a dropdown in both admin app pages to allow a game to be customized based on area of play. Outside, inside, inside a large building, inside a small building. I will then

control the feet of the clue and the feet of when to start scanning for bluetooth based on that.