Tyler Johnston
Aidan Monty
Sam Moon

[Help Me Find!] Resource Finder

- ✓ *Find a refrigerator to stash your lunch!*
- ✓ *Find a microwave to heat it up!*
- ✓ *Locate a comfortable, private restroom!*
- ✓ *Feed your young child in peace!*

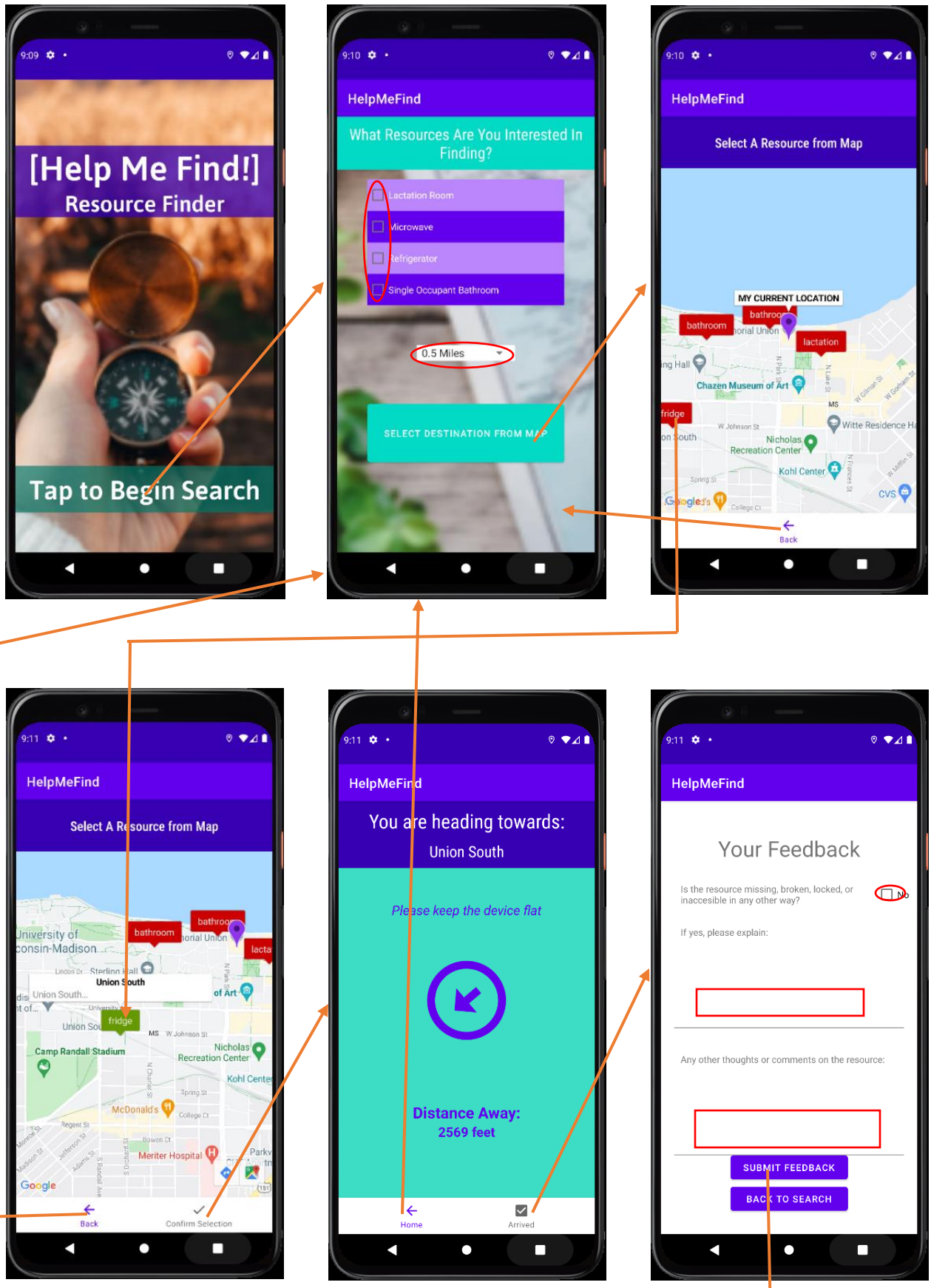
Designed to help students, faculty, and campus visitors locate important resources at UW Madison. Download the app, choose a resource, and follow the helpful pointer to your destination.

Style

For visual style of the application, we chose purple and teal as they are used commonly in themes. The current location is violet to help it stand out against Google's Places. We played with different marker visuals, as discussed in Features Left Out. Our compass image has a solid circle surrounding the arrow to keep the image tolerable to look at, as the arrow itself can move erratically.

Flow

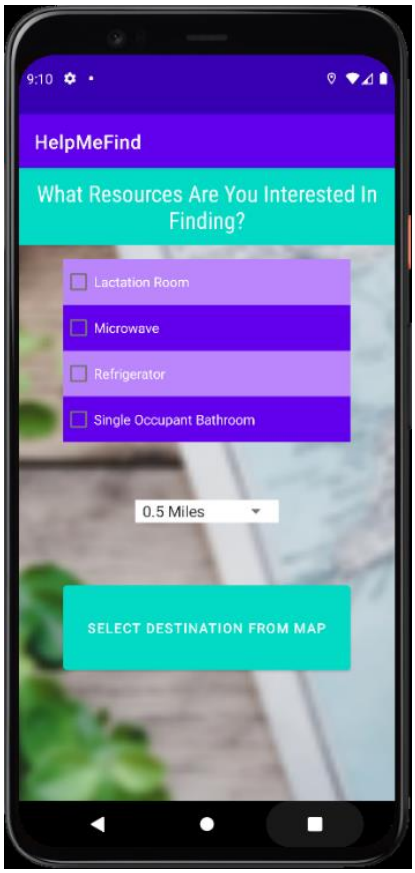
The flow of the application is linear; a user is guided down a straightforward path with the option to go back to the resource selection screen at any point. We also include an administrative site to add resources and view comments from users to check on resource status.

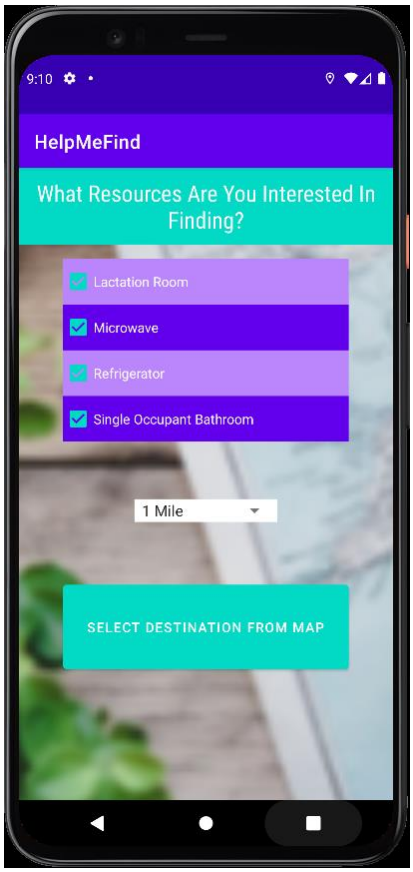

9:09
[Help Me Find!]
Resource Finder
Tap to Begin Search
9:10
HelpMeFind
What Resources Are You Interested In Finding?
Lactation Room
Microwave
Refrigerator
Single Occupant Bathroom
0.5 Miles
SELECT DESTINATION FROM MAP
9:10
HelpMeFind
Select A Resource from Map
MY CURRENT LOCATION
bathroom
lactation
fridge
Chazen Museum of Art
Nicholas Recreation Center
Kohl Center
CVS
Back
9:11
HelpMeFind
Select A Resource from Map
bathroom
fridge
Union South
Camp Randall Stadium
Nicholas Recreation Center
McDonald's
Meriter Hospital
Google
Back
Confirm Selection
9:11
HelpMeFind
You are heading towards:
Union South
Please keep the device flat
Distance Away:
2569 feet
Home
Arrived
9:11
HelpMeFind
Your Feedback
Is the resource missing, broken, locked, or inaccesible in any other way?
No
If yes, please explain:
Any other thoughts or comments on the resource:
SUBMIT FEEDBACK
BACK TO SEARCH

Activities and Implementation

Welcome Activity →

The welcome screen is our landing page for the app, designed to visually allude to our app's purpose. No work is done and no permissions requests are made at this point.



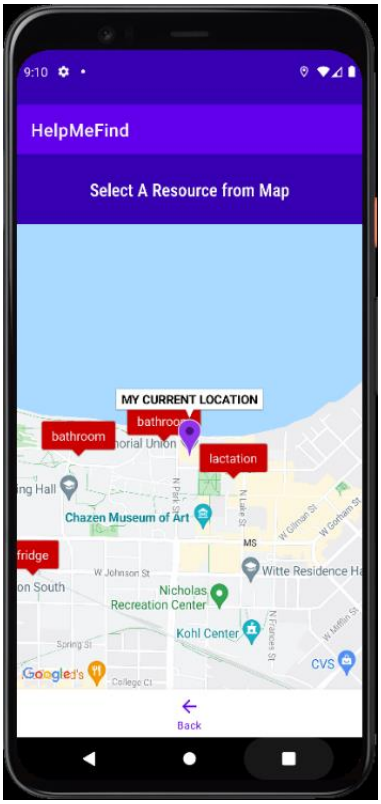


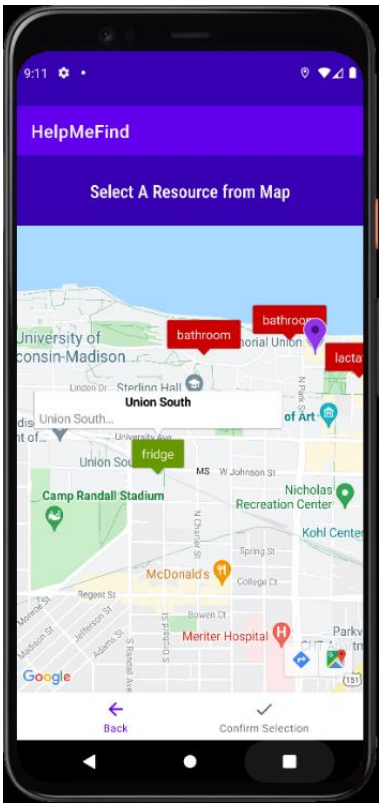
← *FilterSearch Activity*

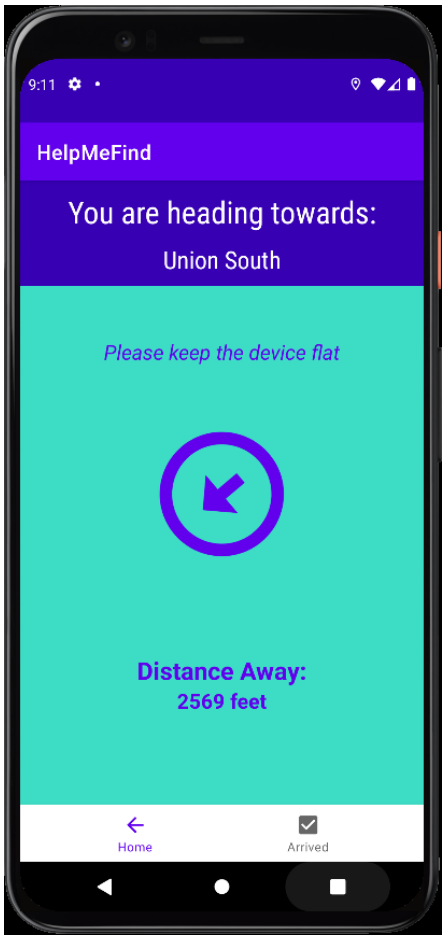
Here, users choose what types of resources to display and the search radius distance. These selections are used to query the Firestore database for a list of resources, which get passed to MapView. Location permissions requests are made here to get the user's current location.

MapView Activity →

The MapView activity displays the resources of the type(s) selected within the radius chosen in FilterSearch. The radius selected also controls the zoom on the Map. The resources and the user's current location are displayed as custom Google Maps markers with their type above. When a user selects a resource, the color changes from red to green and the bottom navigator adds a "Select" option.





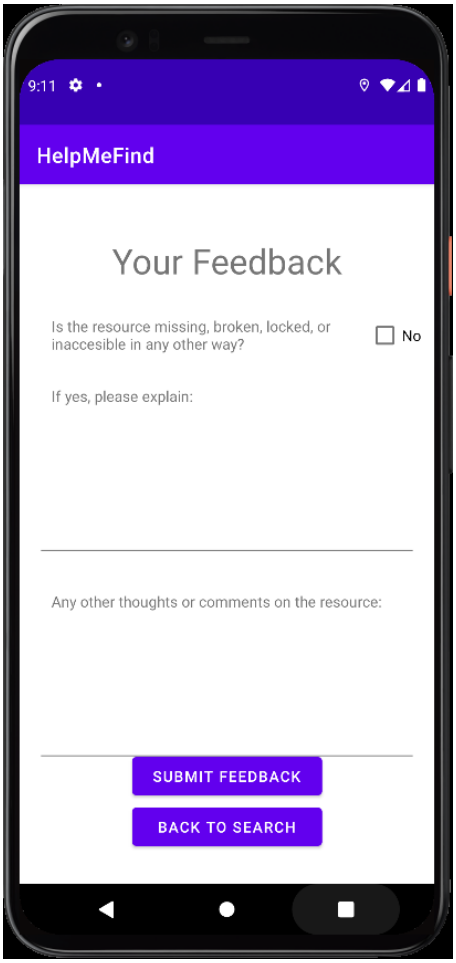


← *Wayfinder Activity*

In the Wayfinder Activity, an arrow points towards the selected resource whose name is displayed at the top of the user's screen. When GPS location is updated the distance from the resource is displayed below the navigation arrow. Once a user is within 65 feet of the given resource's GPS coordinates the distance text changes to "The resource should be in the building in front of you!" By selecting the "Arrived" icon in the bottom navigator a user is taken to the Feedback activity.

Feedback Activity →

In the Feedback Activity a user is asked to report whether the resource is inaccessible or unavailable, with space for explanation. This feedback can then be used by an administrator on the website to remove invalid resources from the database. The user can also add any additional feedback on the resource in the next text area. They have the option to either submit feedback or return to the FilterSearch activity.



Other Elements

Resource Class

This class was used to define the attributes a Resource should have. These include a type, a name, latitude and longitude, an address, and a list of strings to store user's comments on the given resource.

MapsFragment Fragment & MyMarker subclass

The MapsFragment is used to display the resources in a Google Maps view for the MapView activity. The MyMarker subclass is used to create custom map markers that make it clear to the user what types of resources are around them.

New User Comments

Address	Type	Comment	Date Submitted	Accept	Decline
728 State St, Madison, WI 53706	lactation	beautiful resource, very satisfied	Sun Dec 12 2021	Accept	Decline
728 State St, Madison, WI 53706	lactation	good resource!	Sun Dec 12 2021	Accept	Decline
600 N Park St, Madison, WI 53706	bathroom	3:30pm 12/11 follow-up comments	Sat Dec 11 2021	Accept	Decline

Resource Issues

Address	Type	Issue	Date Submitted	Resolve	Delete Resource
600 N Park St, Madison, WI 53706	bathroom	This is a test comment from Tyler Johnston. 3:30pm 12/11	Sat Dec 11 2021	Resolve	Delete Resource

Add a New Resource

Name: Address: Latitude: Longitude: Type:

Submit

Administrative Site

This is a web-based tool to allow administrators to interact with the Firestore database. Here an administrator can add a new resource, decline or accept new user comments, and resolve critical problems on resources reported by users. The administrator can also remove resources from the database.

Stuff we did not learn in class

Custom Markers with text/resource title

This application displays custom markers with the resource type in them. To put text into something that appears as a default marker on Google Maps, text needs to be embedded into an Icon bitmap and then sent to the marker.

```
private Marker addIcon(IconGenerator iconFactory, CharSequence text, LatLng position) {

    Bitmap myIcon  = iconFactory.makeIcon(text);

    MarkerOptions markerOptions = new MarkerOptions().
            icon(BitmapDescriptorFactory.fromBitmap(myIcon)).
            position(position).
            anchor(iconFactory.getAnchorU(), iconFactory.getAnchorV());

    Marker marker = myMap.addMarker(markerOptions);
    return marker;
}
}
```

Passing an object in an intent

To pass an object inside of an intent, we had our object implement the Serializable interface. This allows our activities to send our resource objects to each other and interact with the getter/setter functions in the Resource Class.

Firestore database and Web interface

The application stores the list of resources on Google's Firestore database which can be modified through an admin web interface, also hosted through Firebase. Both the mobile app and the website make extensive use of adding and querying documents from the database.

Bearing to destination and Image rotation

This application calculates the bearing (direction towards) a resource from the current location, using magnetic north, the azimuth (angle from magnetic north to "true north"), and the latitude and longitude of current location and the destination. These calculations are then used to rotate an ImageView using RotateAnimation.

Unique Mobile Features:

The features that make our application uniquely mobile are the use of GPS to determine current location and the use of the accelerometer and magnetometer. While GPS location is also available to desktop users, it is difficult to move around with a desktop or laptop for location-based applications. The accelerometer and magnetometer have unique behavior in a mobile phone as they allow an application to determine the direction a phone is pointing and moving in.

Features left out, Reflection, and Questions

Custom Icon Pictures and Cluster Markers (MapView Activity):

While we were able to create custom Icon Pictures for markers to match more closely with the type of resource, we found that text-based label markers appeared more readable/understandable in our MapView. For similar reasons, we also chose not to include Cluster Markers, (where several clustered resources are displayed as a single bubble with the number of resources as a label until a user zoom in further).

AR:

In our initial design for the application, once a resource was selected on the maps screen, a user would be taken to an AR activity and would walk towards an AR marker that appears to exist on the real location in the user's camera view. Initially, we planned to use Google's SceneForm to do this, but found Google has deprecated it. We also tried to use an open-source project called SceneForm Maintained but had issues running it in our emulator. We tried to display nearby Place objects in AR utilizing ARCore and Google's Places API, but this also failed to work. It seems possible to implement an AR application like our initial proposal given more time, but it may have been less usable for its purpose than the application we ended up creating.

.csv File Upload:

Due to time constraints, we decided against implementing an option for the administrator to upload csv files of resources to the database. Instead, the admin can add resources one at a time through a simple submission form right on the website.

Replacing our Mock Dataset:

Most of our resources in our data storage are mock resources at real locations. For a true deployment of this app, we would want to have an expansive dataset to work with.

What we might do this differently

The Wayfinder arrow could be replaced with another graphic when the current location is not available and when a destination is reached. We would still need to find another way to get the user to move around to acquire location, and we would need to re-add arrow if user moves away from location without selecting "Arrived," so we did not implement this.

Questions, answered!

Q) Why use this instead of Google Maps?

A) Following an arrow instead of a series of dots or text directions can be more pleasant for some users and can lead to finding some interesting shortcuts around campus.

Q) The arrow is "jumpy": How would you fix this?

A) We could do some smoothing while we interpret the data coming from the sensors, as we only need to know the "freshest" data and the most recent "smoothed" outcome. We did not have time to implement this but could do so in a future iteration.