AppalLOCATE: A Lost and Found Solution

A Thesis
by
DEREK CLARK WILSON

Submitted to the Graduate School
Appalachian State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

May 2023

Department of Computer Science

AppalLOCATE: A Lost and Found Solution

A Thesis
by
DEREK CLARK WILSON
May 2023

APPROVED BY:

James B. Fenwick, Jr., Ph.D.
Chairperson, Thesis Committee

Cindy Norris, Ph.D.
Member, Thesis Committee

Mark Hills, Ph.D.
Member, Thesis Committee

Rahman Tashakkori, Ph.D.
Chairperson, Department of Computer Science

Marie Hoepfl, Ed.D.
Interim Dean, Cratis D. Williams School of Graduate Studies

Abstract

AppalLOCATE: A Lost and Found Solution

Derek Clark Wilson
B.S., Appalachian State University
M.S., Appalachian State University
Chairperson: James B. Fenwick, Jr., Ph.D.

AppalLOCATE is a lost and found application currently in beta release to select offices at Appalachian State University. AppalLOCATE comprises three interconnected components of a mobile application, a web application, and a cloud-based storage container. Faculty, staff, and students can employ a user interface to report found items or search items found and reported by others. The application includes a contextualized search function, and incorporates a proximity algorithm to aid in item collection. Reported items can be tagged for easy discovery, GPS location is used to monitor how close a user is to an item, and mobile users can snap a picture of an item they found. This thesis describes the design and development decisions, and explains several expected use cases. Results from a small focus group are also presented.

Acknowledgements

I am so grateful for everyone in my life that has supported me during this journey. Working with Dr. Fenwick over the past two years has been a true pleasure, and I truly believe that this thesis serves as a culmination of all of the effort we have put forth in that time. Thank you for all of the brainstorming sessions, suggestions, and patience, and I hope this project fulfilled your initial expectations from when it was just an idea on your whiteboard.

I am also extremely grateful to my friends and family for being there for me during the writing process. Anytime I got stressed or discouraged, they were right there to help pick me back up. I would not have been able to accomplish this without you all, so I hope you can take pride in this accomplishment with me. Thank you from the bottom of my heart.

Table of Contents

List of Figures

List of Tables

I. INTRODUCTION

At Appalachian State University, there is no standard procedure for reporting or recovering lost items. Expensive "found items" typically are taken to the police department office or some other university office. Inexpensive items seem to collect in university offices, classroom podiums, and lobby window sills, to name a few locations. Most university department offices have a dedicated cardboard box to store such found items, but no reporting or recovery mechanism is available. People looking for a lost item have to rummage through these *de facto* collection points.

A functional prototype of a web application version of AppalLOCATE was presented in partial fulfillment of an undergraduate Honors thesis [15]. AppalLOCATE is a reactive web application built in Vue.js that seeks to solve the prevalent lost and found issue at Appalachian State University. The application allows users to report found items, reporting their location and other item information. That research included a survey of the community to gauge interest in such an application, and the results showed that community members would be likely to use the application to help others.

AppalLOCATE was conceived initially from the perspective of an individual losing or finding an item, however, the requirements gathering and design activities described in [15] shed light on other perspectives and a more holistic, campus deployment. In particular, Appalachian Police Department Chief Stephensen expressed a desire for a campus policy regarding lost and found item management and for several modifications to AppalLOCATE. This thesis expands on the prior work.

Chapter II describes background information. Chapter III describes the design and implementation of the mobile application, including all of the features of the application with a focus on new, novel features. Ways to use AppalLOCATE at Appalachian State University and other locations are discussed in Chapter IV, including a user manual on how the web and mobile versions work. Chapter V details the feedback received about the new versions of AppalLOCATE. Finally, Chapter VI concludes the thesis and discusses future work that could be done to improve AppalLOCATE.

II. BACKGROUND

Research was performed to determine technologies that would be useful to improve AppalLOCATE. Three notable improvements were identified that would better AppalLOCATE users' experiences. Proximity tracing was identified as a way to enable users to find their items faster. A search engine was deemed necessary to allow users to search AppalLOCATE's inventory. Lastly, cross-platform development was explored to create a mobile version of AppalLOCATE. These technologies are explored in this chapter.

A. Proximity Tracing

Proximity tracing is a technology that is used to track the close contacts of individuals who have tested positive for a communicable disease, such as COVID-19. Decentralized implementations of proximity tracing aim to address concerns around data privacy and centralization of data in traditional proximity tracing systems. A protocol that aimed to accomplish this decentralization of data was the Decentralized Privacy-Preserving Proximity Tracing (DP-3T) protocol [14], which performed digital contact tracing (DCT) using smartphones.

Apps using DCT technology were able to alert the user when they came in contact with another person who had been exposed to COVID-19 within a certain time frame, and it did so anonymously. The research and work to create DP-3T inspired Apple and Google to create a shared framework named Google and Apple Exposure Notification (GAEN). An example of an application that used this technology was SlowCOVIDNC, an app released on September 22, 2020, during the COVID-19 pandemic in North Carolina [11]. These efforts allowed for an extra layer of safety during the pandemic that allowed people to gradually begin leaving their homes again to return to work and school.

If two users were to both have the SlowCOVIDNC app downloaded on their devices, both of them would be given what DP-3T calls an ephemeral ID (EphID). This EphID is generated with a hash function and allows for the users to be referred to anonymously by SlowCOVIDNC. When the devices of the two individuals (and presumably the individuals themselves) are near each other, the application will record the proximity of the two. If they are a certain distance from each other for a certain amount of time that is specified by the protocol, their EphIDs will be stored on each other's devices. If later that day, one of the individuals were to report a positive COVID test to health services, their EphID would be tagged with a positive test result and propagate to other devices that they were near. The other individual would receive a notification that they were potentially in contact with someone with COVID-19 and to take the necessary precautions.

This proximity tracing research is relevant due to inspiring a similar system with lost items on campus that is incorporated into AppalLOCATE. Using the longitude and latitude of the input items, which are gathered upon submission of a found item, the precise location of the item can be stored. Using proximity tracing, the location of the device running AppalLOCATE can be compared to the location of the items, allowing the app to take action depending on how close the item is to the device. This DCT process has been named AppalLOCATION. One example of using AppalLOCATION is sending the user a notification to quickly check if an item is still in its reported found location as they walk by. The creation of AppalLOCATION in AppalLOCATE is discussed further in subsection 2.

B. Search Engines

After thinking about AppalLOCATE as an inventory management system, it became apparent that AppalLOCATE would need a robust system for storing and searching for items. This system would need to key on multiple parts of the item such as the name, part of the description, and even tags on the item. Research was conducted to find an efficient and appropriate search engine, but also one that could be extended for use at other institutions.

1) Integrated Storage & Search: The easiest solution is to use built-in indexing from a cloud storage system. Google's Firebase was the initial cloud storage system for AppalLOCATE. Unfortunately, Firebase does not have this indexing feature integrated with the realtime database component [5]. Replacing Firebase with an indexing storage service would require significant changes with the AppalLOCATE codebase. Other integrated solutions have a significant financial cost associated with them. Switching to a different cloud storage system for native indexing capabilities is discussed further in chapter VI.

2) Separated Storage & Search: Another approach is to augment the storage service with a separate indexing service. Elasticsearch is a popular open-source search engine that is known for its scalability, reliability, and efficiency [4]. It uses an inverted index, allowing for fast search results, even with large data sets. Additionally, Elasticsearch supports a variety of search features, such as keyword matching, full-text search, and faceted search. The challenge lies in the implementation to join the search service with the storage service.

Algolia is a popular search engine that is known for its speed and accuracy [1]. The engine uses a proprietary ranking algorithm that prioritizes the most relevant results, allowing users to find what they're looking for quickly and efficiently. One of the key features of Algolia is its instant search capabilities, which allow for results to appear as the user types their query. Additionally, Algolia offers a range of customization options, including the ability to fine-tune relevance, use synonyms, and implement faceted search. However, the use of Algolia in AppalLOCATE is not feasible due to its cost, as it can quickly become prohibitively expensive.

An open-source, free option is Typesense [10]. Typesense is described on their website as the "open source alternative to Algolia." [10] Typesense can be erected on a server, and as long as the server has a public URL, it can be accessed and queried for items on the server. Some of the features of Typesense include support for fuzzy search, typo tolerance, and synonym suggestions. It also offers advanced query capabilities, such as filtering and sorting. Additionally, Typesense has a clean and easy-to-use API that makes it simple to integrate into the application. This is currently how AppalLOCATE's search engine is implemented and will be discussed further in chapter III.

C. Cross-Platform Development

It was apparent early on that AppalLOCATE would serve most users best as a mobile application. However, a web-based application was also necessary to facilitate more intricate functions such as bulk additions and inventory management. This section describes the issues involved in building applications with mobile and web app UI options.

1) Separate Native Applications: While the obvious solution was to create both a web and mobile version of the app from scratch, this is more complicated than it sounds. Web development is often accomplished by combining HTML, CSS, and JavaScript to create web pages that users can interact with. However, this process can be long and cumbersome, as it involves creating code in all three languages to create a website.

Additionally, a mobile app would still have to be created completely separately from the web application. The development of native apps for each platform presented a further obstacle given the varying programming languages used by Apple’s iOS (Swift) and Google’s Android (Kotlin/Java). This brings the number of languages needed to at least five, which can get complicated when trying to implement the same features across all platforms.

2) Reactive Websites: Reactive web frameworks are an attempt at solving the problem of having to use HTML, CSS, and JavaScript for web projects by bringing them under a common framework. While reactive websites still use these three languages, they organize them in a single codebase that is more easily maintained by developers. Some examples of reactive web frameworks are Vue.js and Angular.

Another problem these reactive websites try to solve is displaying content on different-sized screens. Since websites can be accessed from a variety of different devices with varying screen sizes, they need to be able to render well on multiple screens. Websites designed with just HTML, CSS, and JavaScript often do not render well on smaller devices, but reactive websites will shrink the content dynamically to fit smaller screens. This allows for smaller, mobile devices to access the same sites that computers with large monitors can.

However, a reactive website is an inadequate substitute for a mobile version, which is able to take advantage of features embedded in mobile operating systems, such as Bluetooth. Therefore, AppalLOCATE needs to be a mobile app running on the specific mobile device's operating system.

3) Progressive Web Apps: While reactive frameworks are designed from a web application perspective, progressive web apps (PWAs) are designed more from a mobile application perspective. PWAs are meant to be designed in a single web codebase and then "installed" onto a device through a web browser. Ionic, Polymer, and React.js fall into this category due to being focused on creating these cross-platform applications.

PWAs are the newest of the technologies discussed, and therefore do not completely support some operations. Notifications are a big hurdle for PWAs, as iOS does not support being able to send a notification from these applications. The installation of PWAs also does not work on iOS, as it does not support creating a home screen shortcut that will open the application. While PWAs will eventually have these features and be able to be used to create applications that can instantly run independent of the operating system, the technology is currently not in a state to create a production-level application like AppalLOCATE.

4) Cross-Platform Native Apps: Creating cross-platform native apps is similar to creating applications in their native programming language, but instead exports code written in one framework out to native languages. The code for the app is written in a specific web language or framework and then uses an extra tool to convert that code into the equivalent instructions in the mobile native language. For AppalLOCATE, this looks like converting Vue.js code into Swift code for iOS and Native Java/Kotlin code for Android. There are several options for tools that do this translation.

Apache Cordova [2] is a free and open-source framework created in 2009 that allows you to create hybrid mobile applications easily using HTML, CSS, and JavaScript. Cordova offers plenty of pre-built plugins that give developers access to native device functionalities, such as the camera or accelerometer, in a web-based application. It can create a cross-platform mobile app that can be deployed on any operating system, including iOS, Android, and the web, using one programming language. Cordova is also highly versatile and integrates with Vue.js, making it a potentially good choice for AppalLOCATE.

While on paper, Cordova would be a good fit for use in AppalLOCATE, many of its APIs have been deprecated as of 2021. The current version of Cordova was released in December 2021, and any other updates have and will continue to be bug fixes for the versions of iOS and Android from 2021. Although using a deprecated system is possible for new products, it is highly discouraged in most scenarios. This is because many of the technologies used in the deprecated system can be old and sometimes broken, which would make it more difficult to use in production. For this reason, another framework was desired.

The recommended successor to Cordova is a tool named Capacitor. Capacitor [8] is another free and open-source tool used to create cross-platform web and mobile applications. It allows developers to build out the application as a universal application that can then be ported to iOS, Android, and the web, very similarly to Cordova. The difference between the tools comes in their intent. Cordova is more focused on creating mobile applications, whereas Capacitor describes itself as having a “web first” approach [8]. It also integrates mobile-friendly APIs directly in the framework, allowing developers to call these APIs on the web version and have them converted to their respective codebase for mobile platforms.

These features allow for Capacitor to be used in AppalLOCATE’s development, as it allows for a focus on a web version of the application and an easy export into mobile versions. How Capacitor is specifically used in AppalLOCATE is discussed more in chapter III.

III. DESIGN & IMPLEMENTATION

AppalLOCATE is separated into a client-side that encapsulates both the web and mobile versions of the application and a server-side that handles managing the data that users submit. While some work was done on both in [15], substantial work has been done to improve them and extend their functionality for this thesis. This chapter will discuss the design changes that have been made to the client-side to accommodate the mobile version of AppalLOCATE and the development of a search engine and proximity tracing for the server-side.

A. Website UI Design Changes

While the prototype of the web version of AppalLOCATE was completed in August 2022, more work needed to be done to polish the application. This spurred some design changes to better accommodate the application's users. Some of these changes included updating AppalLOCATE's logo, changing the item list page, and adding a search bar to the top of the page.

1) Logo Enhancements: Before doing any substantial work on AppalLOCATE, a meeting was conducted with Appalachian State University's Intellectual Property (IP) council. It was determined after the meeting that AppalLOCATE would continue to be completely owned by the author and could be licensed out as necessary. That decision caused discussions about licensing the software to other universities or institutions in the future, which necessitated a change in branding AppalLOCATE.

The first change that users will notice when accessing AppalLOCATE is the change in the logo. This was necessary for the situation that AppalLOCATE is used on other campuses in the future. Fig. 2 shows what AppalLOCATE's logo used to look like, compared to the new logo in Fig. 1.

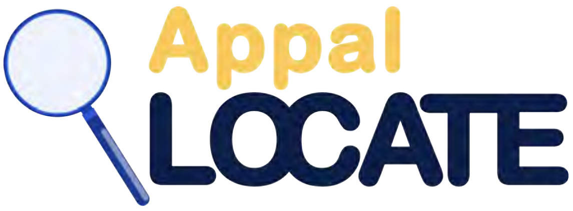


Fig. 1. New AppalLOCATE logo.

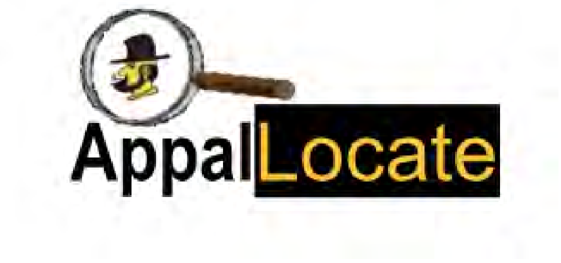


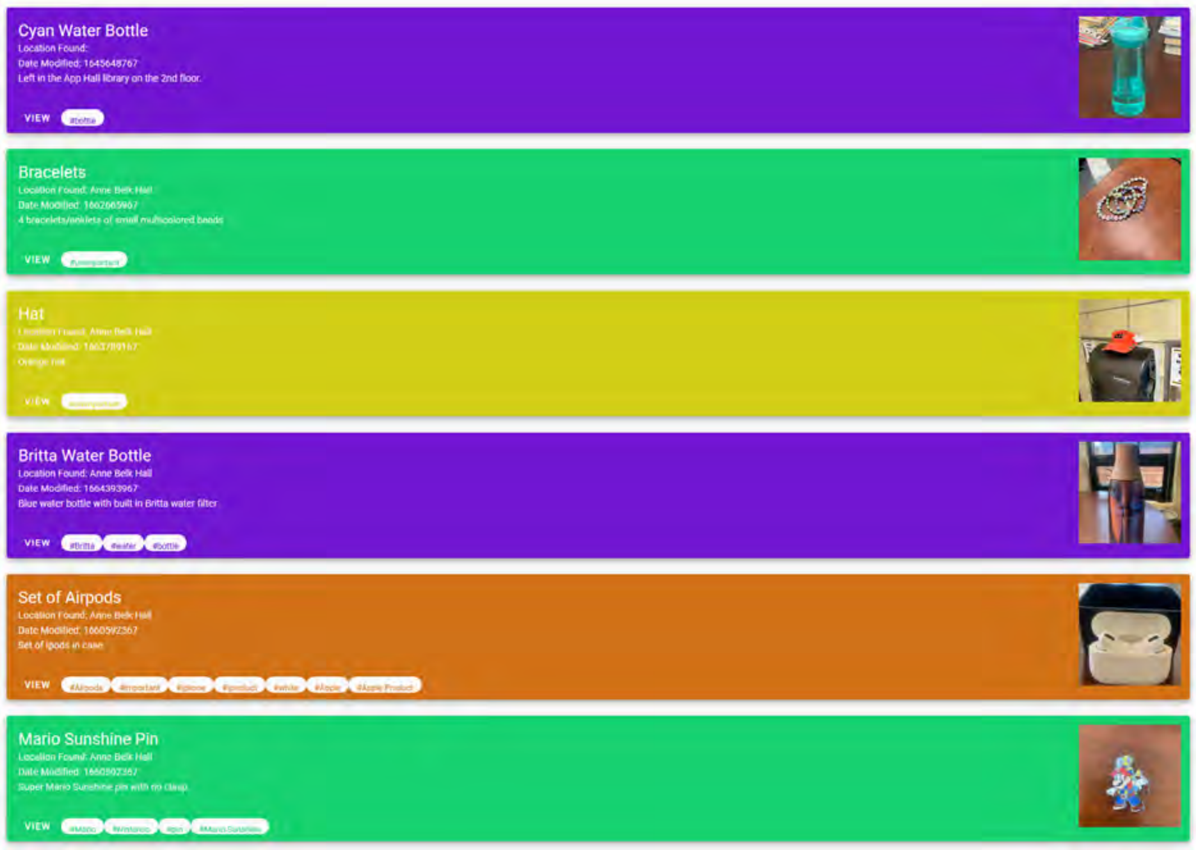
Fig. 2. Old AppalLOCATE logo.

While the magnifying glass and the "LOCATE" part of the name are not meant to change, "Appal" can change to suit any institution that ends up using the technology. Since "Appal" refers to Appalachian State University, it makes sense that others would not want to use this naming scheme. As such, the "Appal" can be switched out for a similar naming scheme and the colors can be changed to reflect the institution.

The magnifying glass can also be used as a simple icon to refer to the application (Fig. 3), such as in the favicon (the icon that shows up in the browser tab) of the website.

Fig. 3. AppalLOCATE icon.

2) Item List Edits: Another change with the design of the application is the item lists. In the previous implementation, the item list was displayed as a cascading list of items stretched horizontally across the screen, as seen in Fig. 4.


Cyan Water Bottle
Location Found:
Date Modified: 1645648767
Left in the App Hall library on the 2nd floor.

Bracelets
Location Found: Anne Belk Hall
Date Modified: 1662665967
4 bracelets/anklets of small multicolored beads

Hat
Location Found: Anne Belk Hall
Date Modified: [illegible]
Orange hat

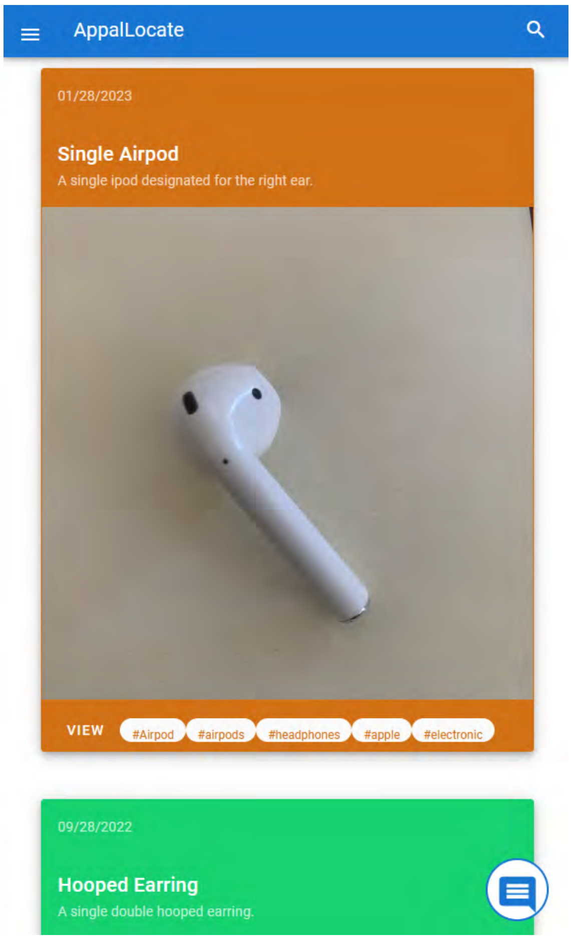
Britta Water Bottle
Location Found: Anne Belk Hall
Date Modified: 1664393967
Blue water bottle with built in Britta water filter

Set of Airpods
Location Found: Anne Belk Hall
Date Modified: 1660592367
Set of Ipods in case

Mario Sunshine Pin
Location Found: Anne Belk Hall
Date Modified: 1660592367
Super Mario Sunshine pin with no clasp

Fig. 4. Old list design.

Now, the item list is displayed as a list of cards that change orientation based on the width of the device's screen, as demonstrated by Fig. 5 and Fig. 6. This allows for the space in the application to be better allocated to show more items to the user, as well as allowing for a cleaner pagination system.



01/28/2023

Single Airpod

A single ipod designated for the right ear.

09/28/2022

Hooped Earring

A single double hooped earring.

Fig. 5. New list design on a narrow screen.

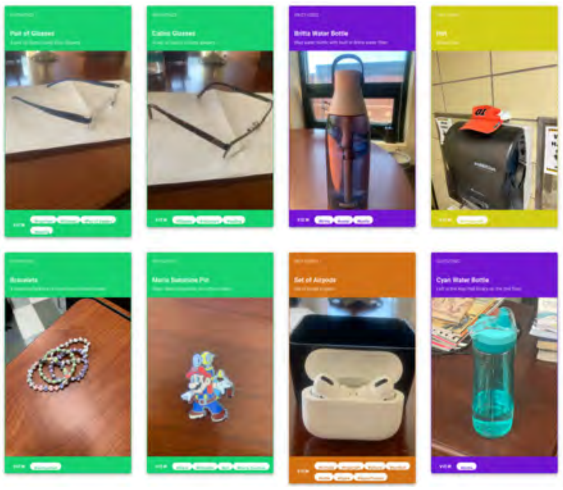


Fig. 6. New list design on a wide screen.

Items now are limited to twelve per page, with users being able to select the page number in the pagination bar at the bottom of the list to decide which page of results they would like to see (Fig. 7).

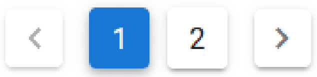


Fig. 7. Pagination bar.

The information that a user is able to input about an item has also changed. This is influenced by the fields that are stored in Firebase, whose realtime database option is being used as AppalLOCATE's backend to house lost, found, and recovered item data. Since AppalLOCATE actually keeps track of lost items, items that have been found and are recoverable, and recovered items, it has three different "buckets" that these items go into. All three hold the same information about an item, which can be seen in Table I.

TABLE I
Final Firebase Item Data

Article Type	Date Submitted	Date Returned
Item Name	Item Description	Item Notes
Item Images	Current Location	Found Location
Listing's Last Modified Date	Listing Tags	Item Value
Who Found the Item	Make	Model
Unique ID of Listing	Unique ID of User that Submitted Item	Recovery Identification Information
Location Coordinates		

This information can be seen on each found item listing in AppalLOCATE for which the user has the proper authorization. More potential Firebase fields are discussed in chapter VI, and authenticating users to view all fields in AppalLOCATE is covered in section E.

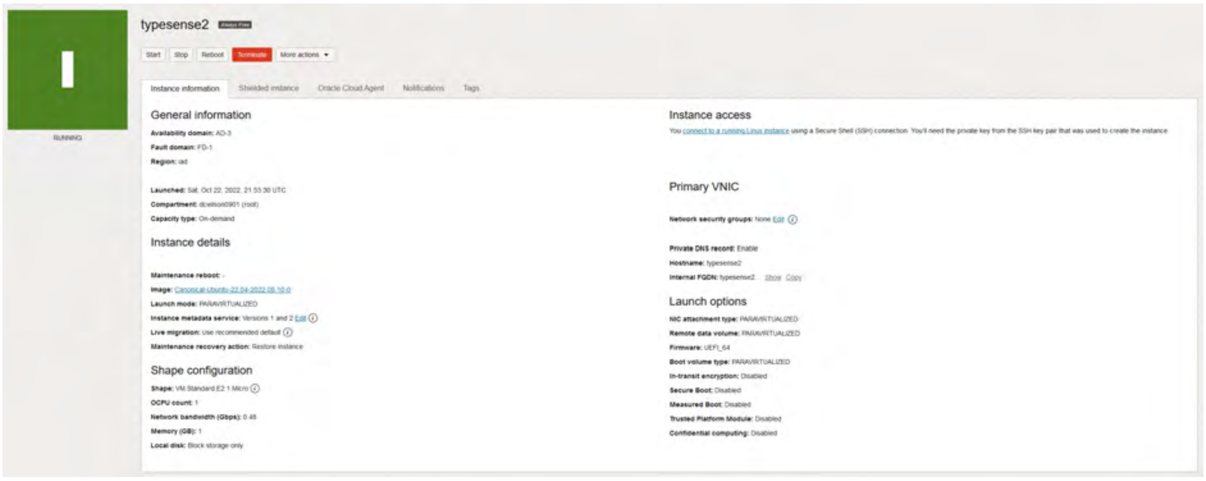
3) Search Bar: The item list pages also coordinate with the new searching feature, which can be accessed by clicking the magnifying glass at the top right of the page (Fig. 8). The implementation of the searching and pagination system uses the Typesense search service.

Fig. 8. Search bar.

B. Typesense & Searching

The searching and pagination systems are made possible by Typesense, an open-source search engine [10]. Typesense has an online pay-as-you-go option called Typesense Cloud that allows developers to index items in their database. However, it also has a free version that can be run and accessed from a developer-hosted server. Oracle's Cloud Infrastructure [7] was used to build this server.

Oracle Cloud has a free tier that developers can use to create virtual machines with a limited amount of resources. The configuration of the server for AppalLOCATE is shown in Fig. 9 and includes a standard Ubuntu Linux distribution, 1 CPU, 1 GB of memory, and a nominal 60 MB network connection. Since this server only runs Typesense, these are sufficient resources for its usage in AppalLOCATE. However, it is not difficult to reprovision and add additional resources later if necessary.



typesense2

General information

Availability domain: AD-3
Fault domain: FD-1
Region: iad

Launched: Sat, Oct 22, 2022, 21:53:30 UTC
Compartment: dcwilson0901 (root)
Capacity type: On-demand

Instance details

Maintenance reboot: -
Image: Canonical-Ubuntu-22.04-2022.08.10-0
Launch mode: PARAVIRTUALIZED
Instance metadata service: Versions 1 and 2 Edit
Live migration: Use recommended default
Maintenance recovery action: Restore instance

Shape configuration

Shape: VM.Standard.E2.1.Micro
OCPU count: 1
Network bandwidth (Gbps): 0.48
Memory (GB): 1
Local disk: Block storage only

Instance access

You connect to a running Linux instance using a Secure Shell (SSH) connection. You'll need the private key from the SSH key pair that was used to create the instance.

Primary VNIC

Network security groups: None Edit

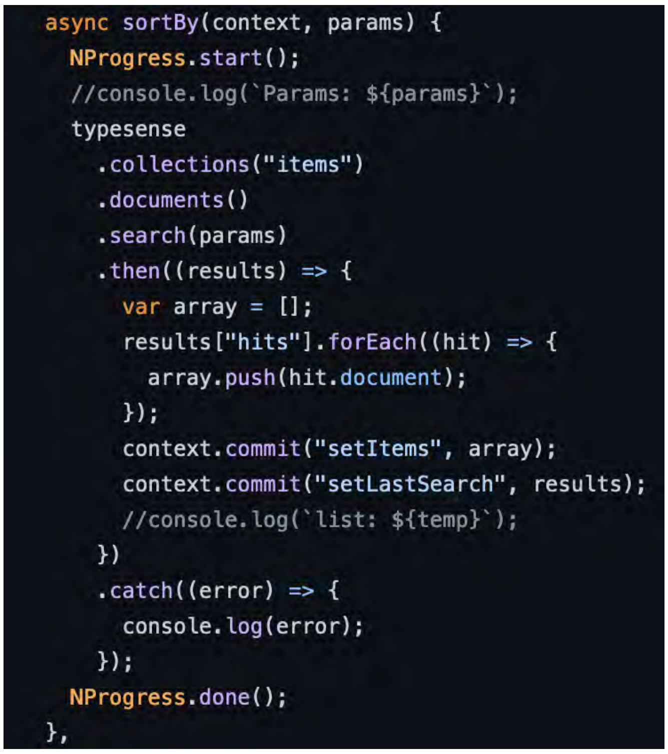
Private DNS record: Enable
Hostname: typesense2
Internal FQDN: typesense2... Show Copy

Launch options

NIC attachment type: PARAVIRTUALIZED
Remote data volume: PARAVIRTUALIZED
Firmware: UEFI_64
Boot volume type: PARAVIRTUALIZED
In-transit encryption: Disabled
Secure Boot: Disabled
Measured Boot: Disabled
Trusted Platform Module: Disabled
Confidential computing: Disabled

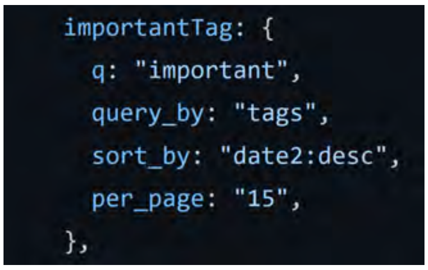
Fig. 9. Typesense server configuration.

Typesense is accessible through an API endpoint. This allows AppalLOCATE to make HTTP requests to this Typesense endpoint whenever an item needs to be indexed, removed, or searched. Items are indexed and removed from Typesense simultaneously with the Firebase storage service, but the search function is separated and shown in Fig. 10.



```
async sortBy(context, params) {
  NProgress.start();
  //console.log(`Params: ${params}`);
  typesense
    .collections("items")
    .documents()
    .search(params)
    .then((results) => {
      var array = [];
      results["hits"].forEach((hit) => {
        array.push(hit.document);
      });
      context.commit("setItems", array);
      context.commit("setLastSearch", results);
      //console.log(`list: ${temp}`);
    })
    .catch((error) => {
      console.log(error);
    });
  NProgress.done();
},
```

Fig. 10. Search function.



```
importantTag: {
  q: "important",
  query_by: "tags",
  sort_by: "date2:desc",
  per_page: "15",
},
```

Fig. 11. Example Typesense query.

This function uses the Typesense library to access the “document” holding the items collection, and then searches that collection for the user’s search terms, which are shared in the $params$ variable. These queries are formatted in a specific way as required by Typesense and usually specify the field that is being searched against (item name, date found, tags, etc.) and what is being put into the search bar. An example query is shown in Fig. 11, which shows how to search for all items that have the “important” tag. This query would be passed in the $params$ variable to the $sortBy$ function and then sent to the Typesense endpoint to search for important items. When the Typesense search function finishes, the results are sent to a lambda function to be processed as a dictionary. The function saves all of the “hits” into an array that will contain all of the important items, and then sends this array to the $setItems$ function to update the local list of items shown to the user. The search results are also sent to the $setLastSearch$ function to be stored for paginating through the results.

C. Web Configuration

Although the prototype of AppalLOCATE was set up using a combination of GitHub and Netlify, more work had to be done to bring the application to a production state. Previously, the application had a randomly assigned hostname that Netlify assigned, but it became evident that the application would need a custom hostname for a unique identity to avoid issues with Cross-Origin Resource Sharing's (CORS) same-origin security policy. This same-origin security policy requires that modern browsers perform all API requests from the same origin, which is the hostname of the application.

1) Domain Setup: The domain name chosen had to be able to host both the main AppalLOCATE site and all of the API endpoints that it uses since JavaScript expects the hostname of a website and an API to be the same. This resulted in the acquisition of appallocate.com, the current hostname for AppalLOCATE.

Port numbers and subdomains separate the different services running on this host. For example, the host to access both the main site and Typesense are the same (appallocate.com), but they both are accessible via different subdomains (www vs search). This allows for the application to not have "cross-origins" in order to obtain the data that is necessary to run (populating item lists, indexing items, etc.). It is easy to accomplish in Netlify, the service that is building out and hosting the production version of AppalLOCATE.

2) Netlify Usage: Netlify is a hosting service used to host web services directly from Git [3]. It also allows developers to manage the subdomains and ports of their applications easily. In AppalLOCATE's case, this is the service that manages the different subdomains that the application needs. Although the Typesense and FingerprintJS (discussed in section E) servers are not running on Netlify like the main site is, it is possible to redirect their respective IPs to refer to specific appallocate.com subdomains. Netlify also makes it easy to obtain an SSL certificate for the application so that it can run over HTTPS instead of HTTP, which increases the security of the site and is mandatory for making API requests.

Netlify also has a feature that allows developers to make branch deployments. These branches directly correlate to branches in GitHub, which are a means of encapsulating different groups of features during development. This feature allows AppalLOCATE to have a separate development version that is assigned to a specific subdomain (dev) that has beta features that have not yet been tested in a production environment. Once these features are thoroughly tested on dev.appallocate.com, the changes on the dev branch can be merged into the production (prod) branch, which tells Netlify to rebuild the version of AppalLOCATE hosted at appallocate.com. Netlify also has analytical features that are not yet utilized in the current version of AppalLOCATE, and are discussed further in chapter VI.

D. Mobile Development

As stated previously, Capacitor was used to export the web version of AppalLOCATE to both an iOS and Android version of the app. While using Capacitor is straight-forward, a few design changes were needed in order to make AppalLOCATE display and operate correctly in a mobile environment. Further improvements on the mobile version are discussed in chapter VI, such as Bluetooth and GPS.

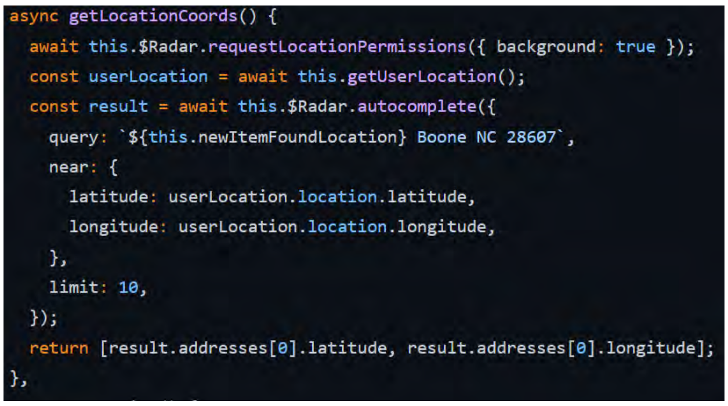
1) Capacitor Usage: Building out the mobile version of the application is easy to do with Capacitor. All that is required is running a few commands that specify which mobile operating system to build out to:

```
npx capacitor add [ios|android]

npx capacitor sync [ios|android]

npx capacitor open [ios|android]
```

First, the necessary dependencies are installed for the respective operating system with the *capacitor add* command. This step only has to be done once, as after it has those dependencies, it can now export out to the required operating system. This export is accomplished by running the *capacitor sync* command, which begins the process of exporting all of the Vue.js code in the project out to Swift and Kotlin code. If any updates are made to the web version, the Capacitor sync must be executed again to synchronize the web and mobile versions. After the builds are complete, the respective projects can be opened with the *capacitor open* command. This will open the project in either Xcode or Android Studio depending on the operating system specified. At this point, there is an application that can now be configured to deploy to the operating system's respective app stores. Additionally, any changes that can only be made in the operating system's native programming language are now also possible. Capacitor provides specific files that can still be modified to add in native code that will not be overwritten on a subsequent sync.

2) AppalLOCATION: AppalLOCATION is a digital tracing protocol (DCP) that is provided by AppalLOCATE to assist users with finding their items faster. AppalLOCATION currently uses Radar [12] to grab the latitude and longitude of users and items. Radar is used because it interfaces with Capacitor, allowing for just one API to be used in both the web and mobile versions of AppalLOCATE. Since Radar needs location service permissions in order to operate, users are prompted to enable location services whenever AppalLOCATION is enabled. Since location services are not required for every operation in AppalLOCATE though, these prompts are limited to only these situations when AppalLOCATION is used.

Radar is used to grab the latitude and longitude of an item when a user submits a found item, as shown in Fig. 12.



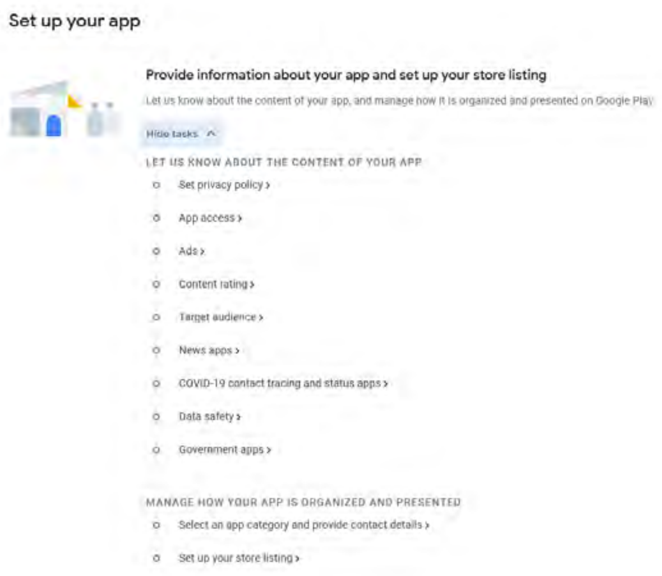
```
async getLocationCoords() {
  await this.$Radar.requestLocationPermissions({ background: true });
  const userLocation = await this.getUserLocation();
  const result = await this.$Radar.autocomplete({
    query: `${this.newItemFoundLocation} Boone NC 28607`,
    near: {
      latitude: userLocation.location.latitude,
      longitude: userLocation.location.longitude,
    },
    limit: 10,
  });
  return [result.addresses[0].latitude, result.addresses[0].longitude];
},
```

Fig. 12. Get location coords function.

The API is sent the location specified by the user when creating the item listing and the user's current location to obtain the coordinates for the item. If the user is indoors, the item listing just uses the geolocation of the building, since GPS systems can be inaccurate indoors. In this case, the user's location (who is assumed to be in Boone) is used to grab the closest address's coordinates, which helps Radar be as precise as possible. Otherwise, if the user is outside, the item will take the user's current coordinates to be more accurate. Typesense's "geopoint" datatype can then be used to store these coordinates, which allows for location-based queries to be done on the item. This is later used by AppalLOCATION to tell a user what items are in their immediate vicinity. How users can use this feature is discussed in section A.

3) App Deployment: The mobile application implementation is made publicly available for users by submitting the finished apps to their respective app stores, which are Apple's App Store and the Google Play Store. Deploying AppalLOCATE to the Play Store is completed and is described in this section. Deployment to the App Store is discussed as future work in chapter VI.

Deploying an application to the Google Play Store is made possible using the Google Play Console [6]. A small, one-time fee has to be paid to access the console, but then all of the resources to release the application are provided to developers. The Google Play Console provides a list of criteria that must be met to submit the first release of an application. These criteria are shown in Fig. 13.


Set up your app

Provide information about your app and set up your store listing

Let us know about the content of your app, and manage how it is organized and presented on Google Play

Hide tasks

LET US KNOW ABOUT THE CONTENT OF YOUR APP
- Set privacy policy ›
- App access ›
- Ads ›
- Content rating ›
- Target audience ›
- News apps ›
- COVID-19 contact tracing and status apps ›
- Data safety ›
- Government apps ›

MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED
- Select an app category and provide contact details ›
- Set up your store listing ›

Fig. 13. Google Play Console checklist.

Most of the requirements are straightforward, like specifying if the application is going to be showing users ads, providing news to users, or is going to be used as a COVID-19 proximity tracing app, none of which AppalLOCATE is doing. The more applicable requirements are setting a privacy policy, content rating, and disclosing data safety information.

A privacy policy is a legal form that specifies how an application uses user data. This is necessary to prevent malicious applications from using user data inappropriately without the user's knowledge and adds an extra layer of security. Creating a privacy policy is rather easy given the correct tools, such as TermsFeed [13]. AppalLOCATE uses TermsFeed to create a privacy policy, which is being hosted at appallocate.com/privacy. Providing the URL to the privacy policy is required by Google so that they can consider it during their verification process.

Content rating is a way to determine what age range of users should be using an application. For apps and games, the content rating given is usually determined by the International Age Rating Coalition (IARC), which will assign the appropriate age restrictions to these applications internationally.

Google determines content rating by requiring developers to complete a questionnaire that will assign the application a rating based on what kind of content is present in it. AppalLOCATE has been determined to be rated "E for Everyone" in the Americas (Fig. 14), as nothing in the application is harmful to younger demographics, although it is not specifically designed for them.

Fig. 14. AppalLOCATE's content rating.

Google requires that all submitted apps disclose what data they collect on a user's device upfront so that they can consider it during their review. This can be information like the user's name, email address, photos, or even physical address. This is to prevent any malicious applications from being approved on the Google Play Store. If Google determines a mismatch between the information provided by the developer and what is actually in the application it will not be allowed on the store. Since AppalLOCATE collects the location and unique identifier of a user's device when they submit an item and allows users to log into the application with their email address and password, this has to be disclosed to Google.

Since AppalLOCATE has undergone review and been approved, it can now be listed on the Google Play Store. If a user searches for "AppalLOCATE" on the Google Play Store, they will now see the public listing for the app and be able to install it on their Android device, as shown in Fig. 15. This allows any member of the Appalachian State University community to use the application and take advantage of the mobile features that have been implemented. Use cases for the web and mobile versions of the application are discussed further in chapter IV.

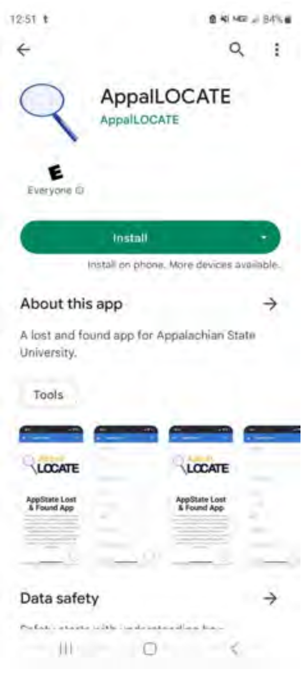

12:51
AppalLOCATE
AppalLOCATE
Everyone
Install
Install on phone. More devices available.
About this app
A lost and found app for Appalachian State University.
Tools
LOCATE
AppState Lost & Found App
LOCATE
AppState Lost & Found App
Data safety

Fig. 15. Google Play Store listing.

E. Security & Authentication

While AppalLOCATE is public for anyone in the Appalachian State University community to use, not everyone should have access to all of the information in the application. If all of the information about items were public knowledge, it would be easy for someone to fraudulently look at the item list and claim that any item present on the list is theirs. This fraud concern was discovered during requirements gathering with the Appalachian State Police Department (APD), and influenced the implementation of an authentication service into AppalLOCATE.

For recovered items, standard AppalLOCATE users should only be able to see the name of an item, where it is being kept, and when it was found. Office users can instead look at the full extended information about an item. This will allow these standard users to go to the location where the items are being held and describe their lost items to office personnel, who will be able to determine if the information the user is giving them matches an item's full description.

The authentication system also needs to be able to be generalized so that any institution that uses AppalLOCATE in the future can also authenticate their users. At Appalachian State University, SAML 2.0 is used to authenticate users through our single sign-on service, Shibboleth. To get a user's information from Shibboleth, AppalLOCATE needed to be able to send SAML 2.0 requests. SAML 2.0 requires that a personal and private key be generated for the application that will be using it, and the identity service will then check against the personal key to make sure that the application is authorized to obtain the user's information. Once the information is acquired for a user, AppalLOCATE can determine if the user is in an Active Directory (AD) group named CS-AppalLocate-Admins that is allowed to view extended item information.

This group currently includes department office personnel, APD office workers, and authorized student union workers. This group can be added to and removed without impacting AppalLOCATE, which will allow any new users to then be able to access the extended information. From the user's perspective, they are logging into the service with their Appalachian State University credentials and authorizing the request with Duo Mobile, much like any other service at the University. This process is shown in section B.

Whenever they are finished viewing the information, the user can choose to log out of the instance of the application they are using, otherwise, they will be logged out after a determined amount of time.

As previously mentioned, this authentication system has been generalized so that any institution can take advantage of it. A developer would just need to reroute the keys in the application to their own Shibboleth equivalent, which will then allow them to authenticate users in a similar way.

Sometimes, there is also a need to be able to determine which users have submitted what items, such as allowing the user submitting an item to modify the listing. Since AppalLOCATE is primarily designed to not need to be logged into by standard users, using Shibboleth to identify these users is undesirable. This inspired the usage of FingerprintJS [9] to give users a hashed, unique user ID that is generated based on instance information (serial number, browser, etc.). Since this user ID is created on the device and then sent up with an item submission, AppalLOCATE has no way to trace the ID back to a user. These IDs persist and allow users to be identified as the publisher of an item listing without having to log into AppalLOCATE, so long as the user uses the same device.

IV. APP USES & PROCEDURES

Since some features of the application are exclusive to office personnel, AppalLOCATE has two different use cases for two different kinds of users. These two types of users in AppalLOCATE are *standard* and *office* users. Standard users are any students, custodians, or visitors to Appalachian State University's campus that may download the application to report their lost items or see if they have been found and reported to an office space. These users can only see basic information about items held in offices to mitigate false item claims. Office users are the office personnel who will have full access to held items' information, including images and full descriptions. This chapter will discuss both of these types of users and how it is recommended that they use AppalLOCATE.

A. Standard Users

This section will describe the steps for standard users to accomplish certain goals with AppalLOCATE. It is expected that standard users use the mobile version of AppalLOCATE, but these steps will apply to the web version as well. The basic actions that standard users can perform are creating listings for their lost items and reporting found items.

To create a bulletin board listing for a lost item, a standard user will open AppalLOCATE and navigate to the "Submit Lost Item" page from the navigation bar (Fig. 16). The user can then fill out the information about their item as shown in Fig. 17, including a name for the item, where and when it was last seen on campus, a short description of the item, and the user's name. The only information that is required to post the listing is the name of the item and where and when the user last had the item. If "Submit another" is checked on the page, some of the information about the previous item will persist to make it easier to submit multiple items in succession.

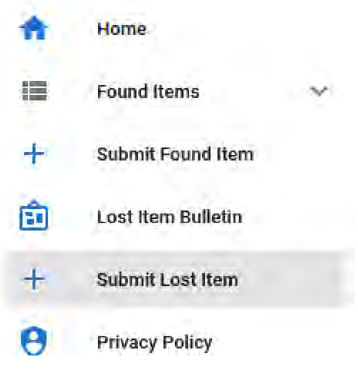


Fig. 16. Submit lost item listing.

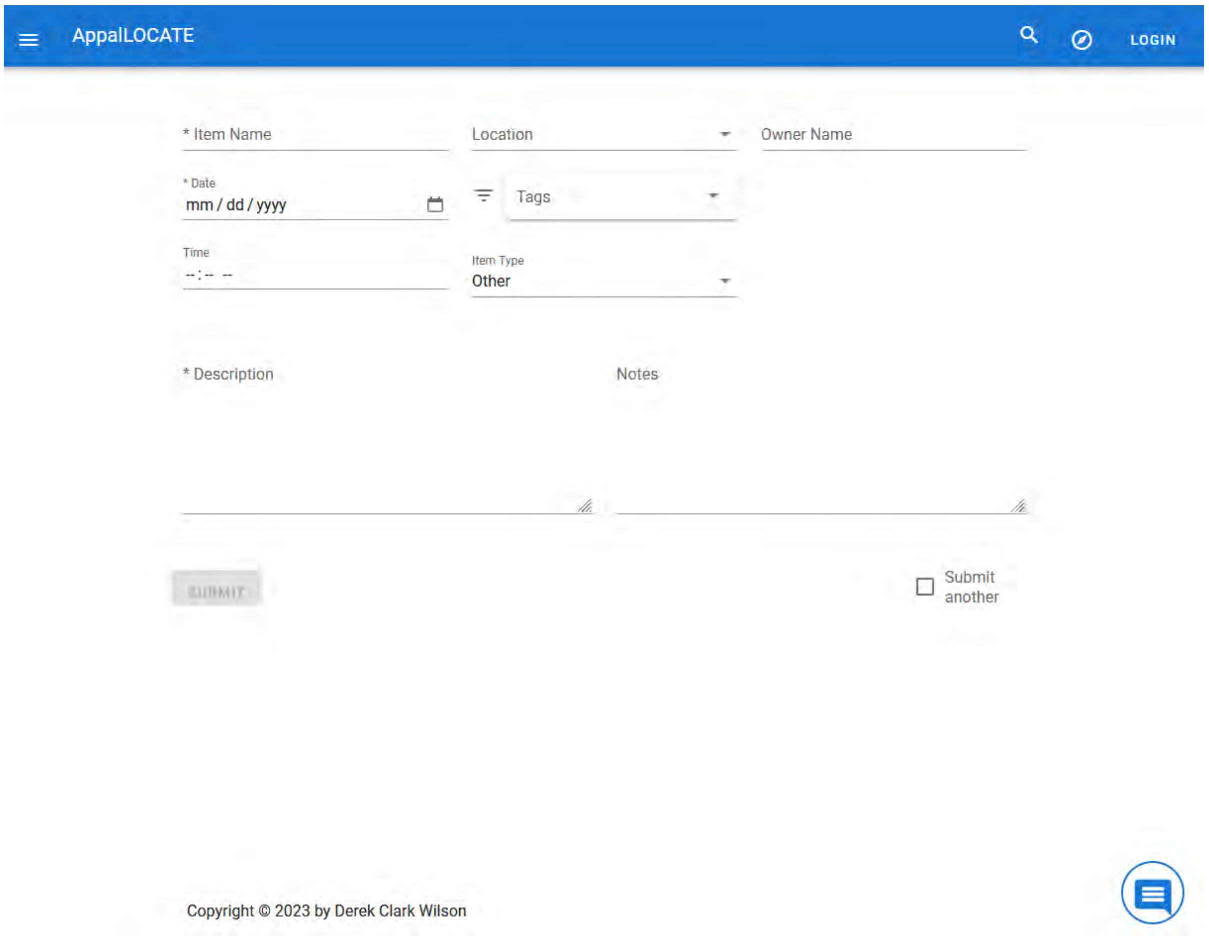


Fig. 17. Submit lost item page.

The items on the bulletin board can also be viewed from the “Lost Item Bulletin” page (Fig. 18). This will show all of the items that have been reported missing, as shown in Fig. 19. This allows users to view this list and keep their eyes open around campus for any of these items, which can then be submitted as found items.

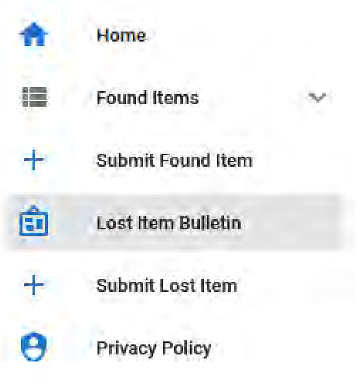
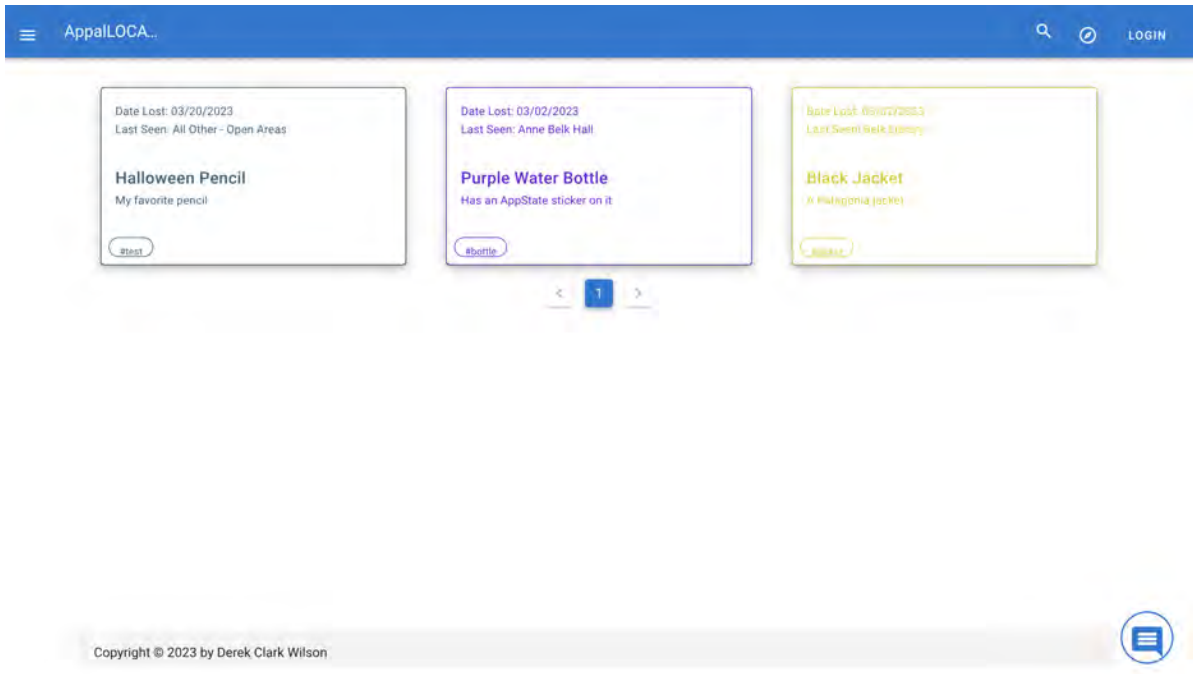


Fig. 18. Lost item bulletin listing.



Date Lost: 03/20/2023
Last Seen: All Other - Open Areas
Halloween Pencil
My favorite pencil
#test

Date Lost: 03/02/2023
Last Seen: Anne Belk Hall
Purple Water Bottle
Has an AppState sticker on it
#bottle

Date Lost: [illegible]
Last Seen: Belk Library
Black Jacket
A Patagonia jacket

Fig. 19. Lost item bulletin page.

To create a listing for a found item, a standard user can navigate to the “Submit Found Item” page from the navigation bar (Fig. 20). The user can then take a picture of the item that they found and enter the required information about it. This item will then be available from the “Found Items” page, shown in Fig. 21, which can also be accessed from the navigation bar.

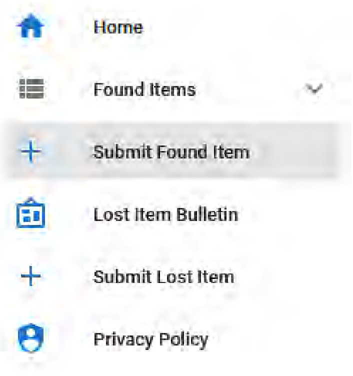


Fig. 20. Submit found item listing.

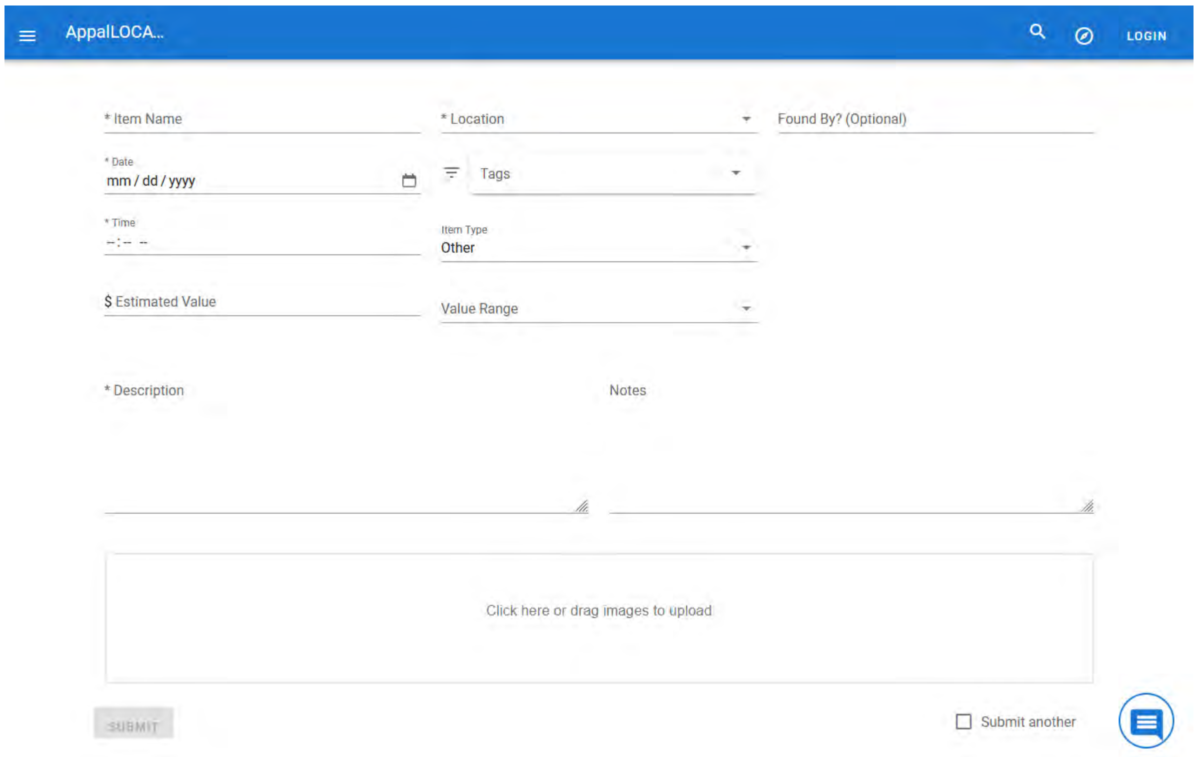


Fig. 21. Submit found item page.

The “Found Items” page shows all items that have been confirmed to be seen on campus (Fig. 22). A standard user can see the name of the listing, the date it was found, and where it currently is if that location is at a confirmed office location, which is shown in Fig. 23. This allows the user to identify if any of their items have been submitted and go to the appropriate office to give a detailed description of their item to retrieve it.

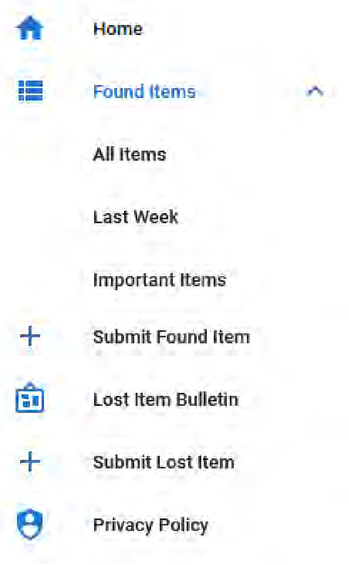
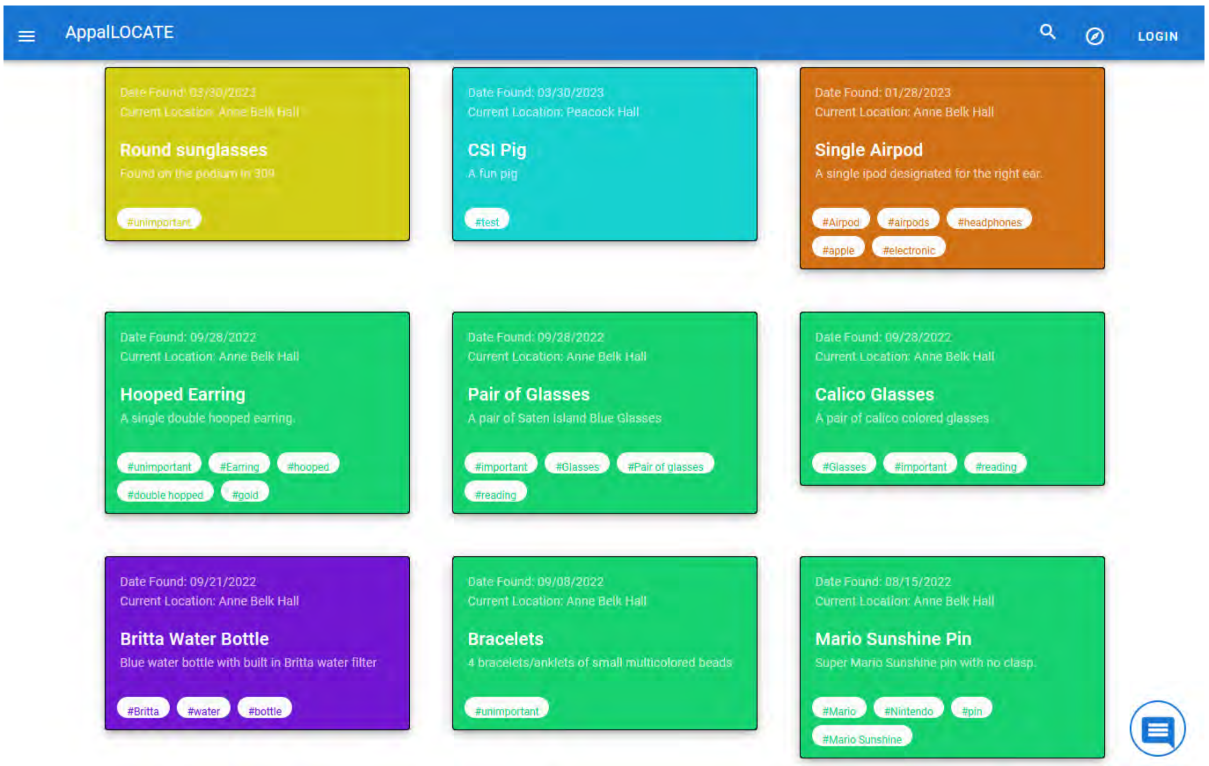


Fig. 22. Standard user’s found items listing.



Date Found: 03/30/2023
Current Location: Anne Belk Hall
Round sunglasses
Found on the podium in 309
#unimportant

Date Found: 03/30/2023
Current Location: Peacock Hall
CSI Pig
A fun pig
#test

Date Found: 01/28/2023
Current Location: Anne Belk Hall
Single Airpod
A single ipod designated for the right ear.
#Airpod #airpods #headphones #apple #electronic

Date Found: 09/28/2022
Current Location: Anne Belk Hall
Hooped Earring
A single double hooped earring.
#unimportant #Earring #hooped #double hopped #gold

Date Found: 09/28/2022
Current Location: Anne Belk Hall
Pair of Glasses
A pair of Saten Island Blue Glasses
#important #Glasses #Pair of glasses #reading

Date Found: 09/28/2022
Current Location: Anne Belk Hall
Calico Glasses
A pair of calico colored glasses
#Glasses #important #reading

Date Found: 09/21/2022
Current Location: Anne Belk Hall
Britta Water Bottle
Blue water bottle with built in Britta water filter
#Britta #water #bottle

Date Found: 09/08/2022
Current Location: Anne Belk Hall
Bracelets
4 bracelets/anklets of small multicolored beads
#unimportant

Date Found: 08/15/2022
Current Location: Anne Belk Hall
Mario Sunshine Pin
Super Mario Sunshine pin with no clasp.
#Mario #Nintendo #pin #Mario Sunshine

Fig. 23. Standard user’s found items page.

AppalLOCATION is also available to be used by standard users. AppalLOCATION can be utilized by the user by clicking the compass icon at the top right corner of AppalLOCATE, as shown in Fig. 24.

This will immediately prompt the user to enable location services for the application, causing a search to be performed for every item within a 200-foot radius. This allows users to just look for items in their immediate vicinity, in case there was a found item reported in the area. Eventually, users will be able to get push notifications to look out for these found items in their location, which is discussed in chapter VI.

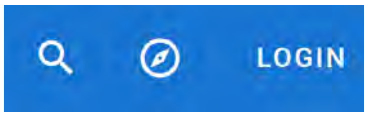


Fig. 24. AppalLOCATION button.

If a user is not sure where to return an item that has been deemed important, they can navigate to the "Item Info" page (Fig. 25). This page (Fig. 26) provides some basic information about where to return an important/expensive item so that it can remain safe.

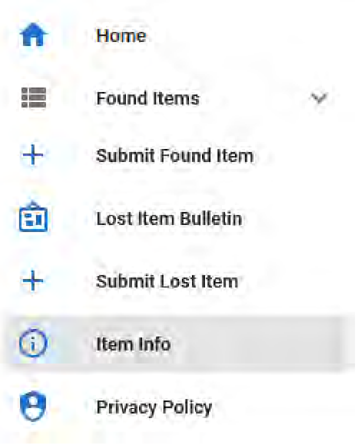
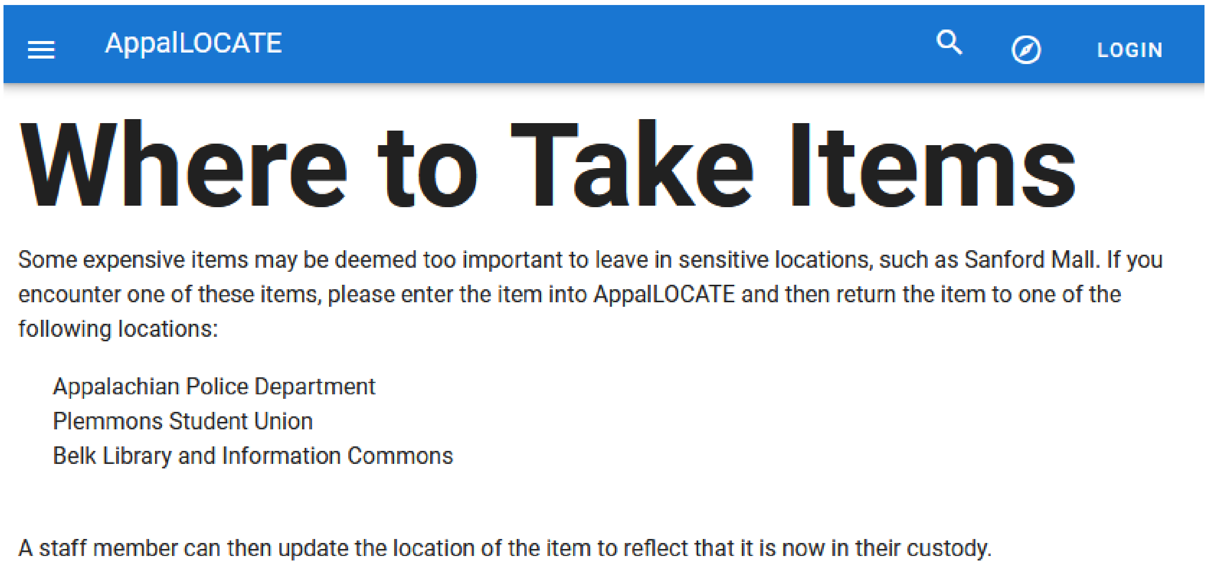


Fig. 25. Item info listing.



Where to Take Items

Some expensive items may be deemed too important to leave in sensitive locations, such as Sanford Mall. If you encounter one of these items, please enter the item into AppalLOCATE and then return the item to one of the following locations:

Appalachian Police Department
Plemmons Student Union
Belk Library and Information Commons

A staff member can then update the location of the item to reflect that it is now in their custody.

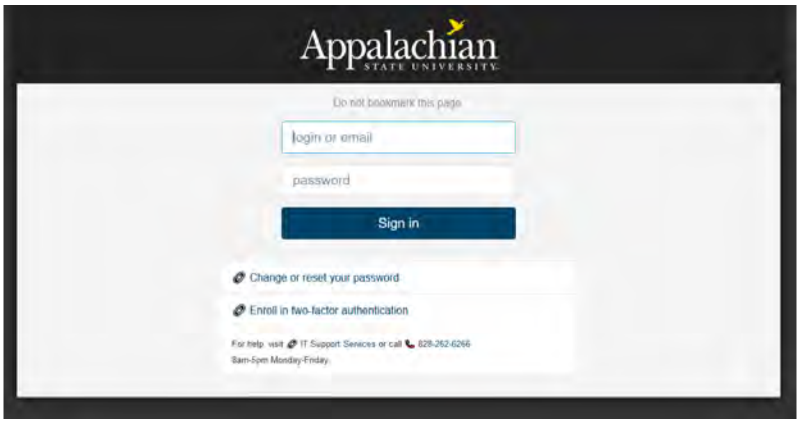
Fig. 26. Item info page.

B. Office Users

This section will describe the steps for office users to accomplish certain goals with AppalLOCATE. It is assumed that office users use the web version of AppalLOCATE, but these steps apply to the mobile version as well. The basic actions that office users can perform are creating listings for lost items, reporting found items, viewing detailed information about submitted items, and moving found items to a recovered state when they are returned to their owner. Office users can perform all of the actions of standard users, so shared actions have been consolidated in the previous standard user section.

Office users must first log in by clicking the “Login” button at the top right of the page (Fig. 27). Users can log in with their AppState username and password (Fig. 28), and the application will confirm that the user is allowed to use office features. After logging in, the user will be redirected back to the AppalLOCATE main page, where office users can now navigate the full application.

Fig. 27. Login button.


Appalachian STATE UNIVERSITY

Do not bookmark this page

login or email

password

Sign in

Change or reset your password

Enroll in two-factor authentication

For help visit IT Support Services or call 828-262-6266
8am-5pm Monday-Friday

Fig. 28. Login page.

To view the full information for a lost item, an office user can open AppalLOCATE and navigate to the "Found Items" page from the navigation bar (Fig. 29). The user will be presented with a different view of the items from a standard user, which includes pictures of the items, shown in Fig. 30. To see more information and edit a listing a user can click on the "View" button on the item, which will bring up a popup full of all of the information entered for the item.

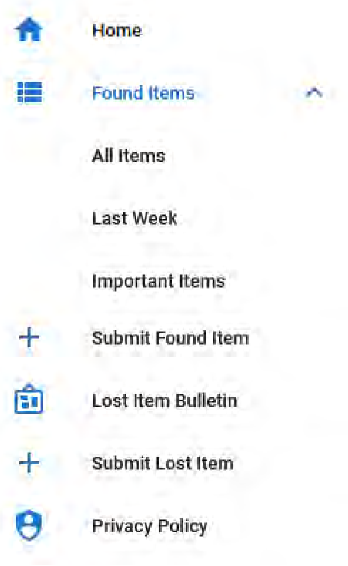


Fig. 29. Office user's found items listing.

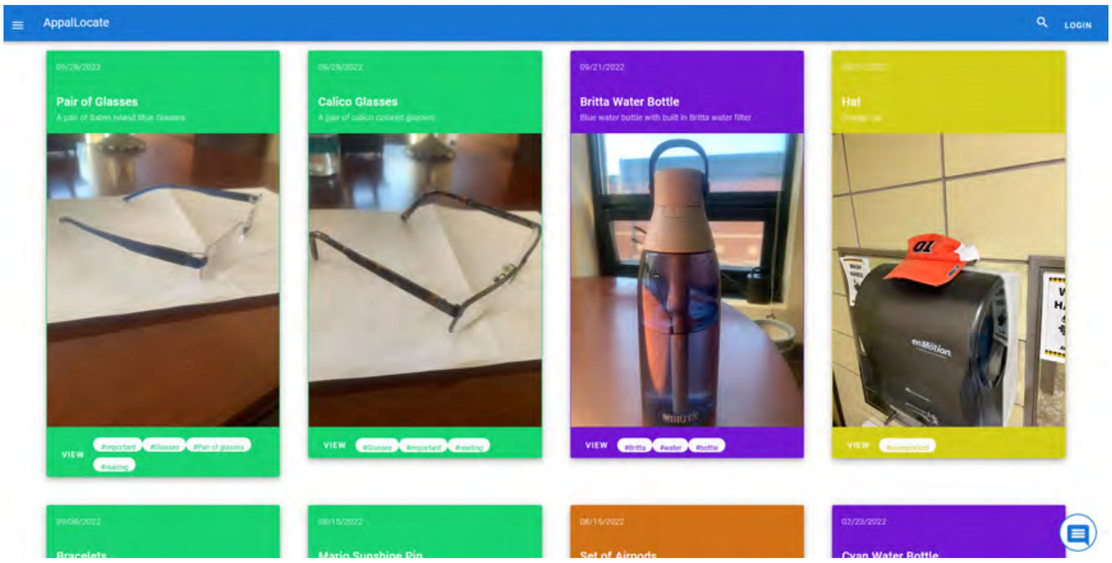
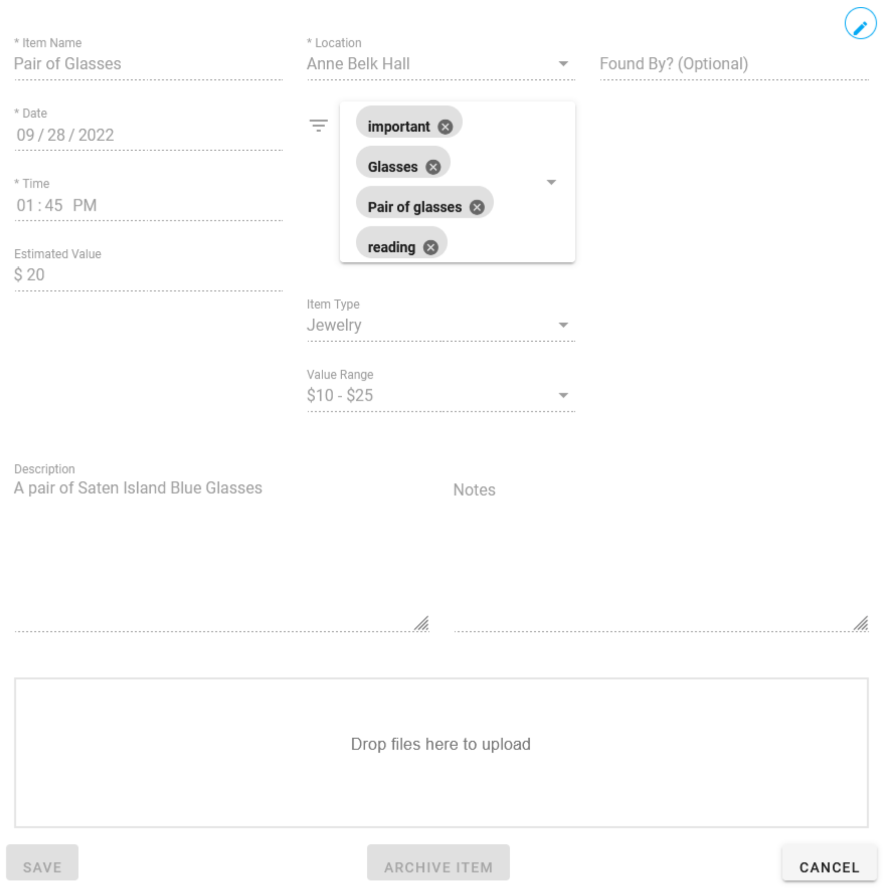


Fig. 30. Office user's found items page.

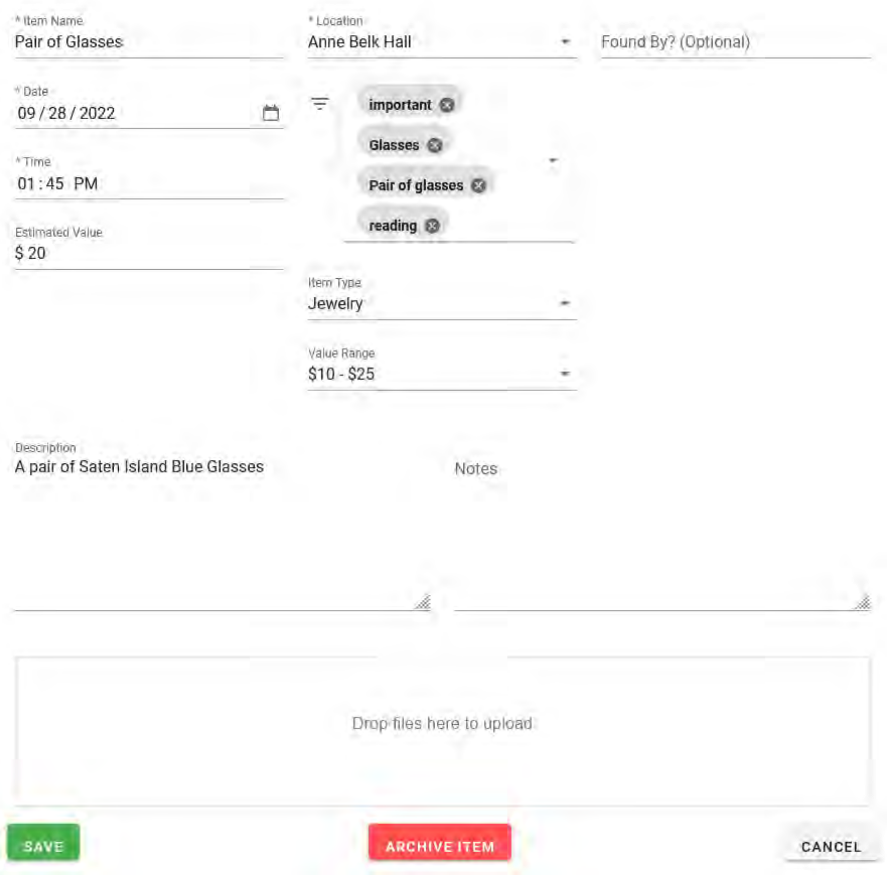
This information can also be edited by clicking the pencil at the top right of the popup (Fig. 31), as this will enable each field to be changed and then saved with the "Save" button (Fig. 32). If the office user wants to mark the item as having been recovered by its owner, the "Archive" button can be used to do so.



* Item Name: Pair of Glasses	* Location: Anne Belk Hall	Found By? (Optional)
* Date: 09 / 28 / 2022	important, Glasses, Pair of glasses, reading	
* Time: 01 : 45 PM		
Estimated Value: $ 20	Item Type: Jewelry	
	Value Range: $10 - $25	

Description: A pair of Saten Island Blue Glasses

Fig. 31. Found item's extended view.


* Item Name
Pair of Glasses

* Location
Anne Belk Hall

Found By? (Optional)

* Date
09 / 28 / 2022

important
Glasses
Pair of glasses
reading

* Time
01 : 45 PM

Estimated Value
$ 20

Item Type
Jewelry

Value Range
$10 - $25

Description
A pair of Saten Island Blue Glasses

Notes

Drop files here to upload

Fig. 32. Found item's edit page.

Historic items that have been marked as recovered can be viewed by navigating to the "Recovered Items" page (Fig. 33). This page (Fig. 34) looks similar to the "Found Items" page, but individual item information can no longer be modified on the items. For record-keeping purposes, the displayed information for the office user has been updated to include the recipient of the item.

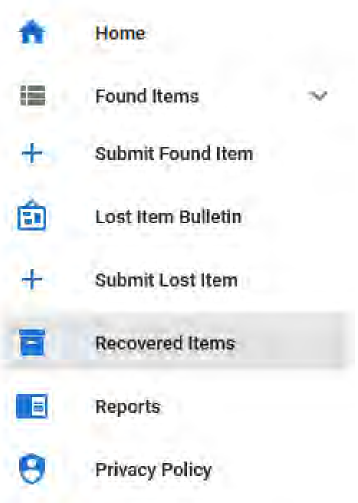


Fig. 33. Recovered items listing.

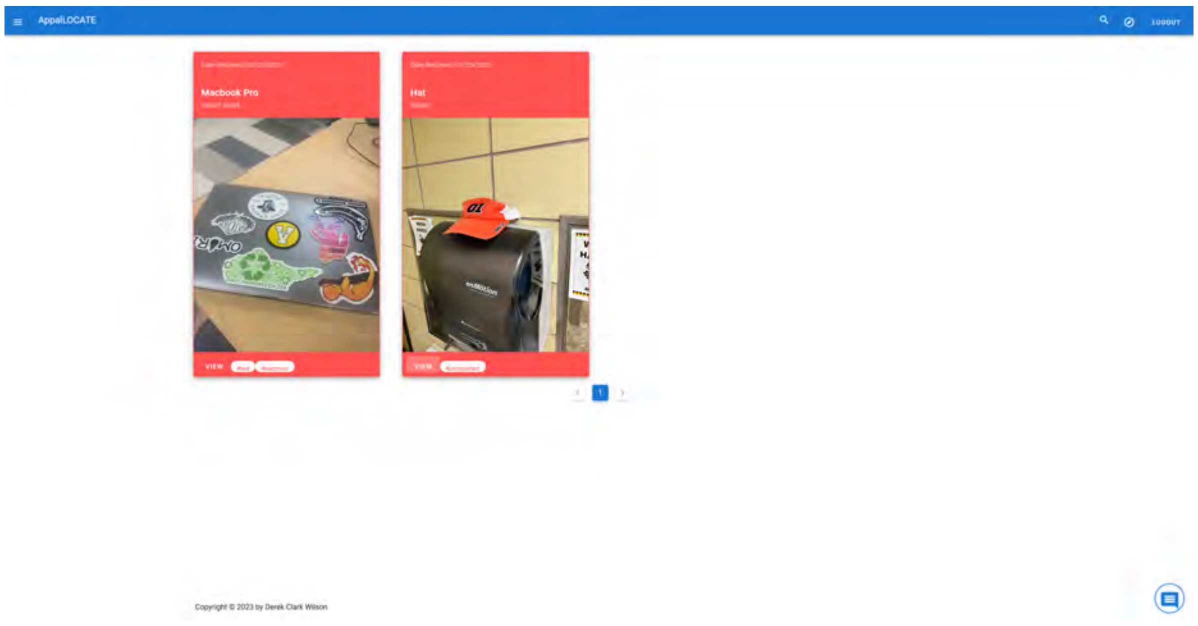


Fig. 34. Recovered items page.

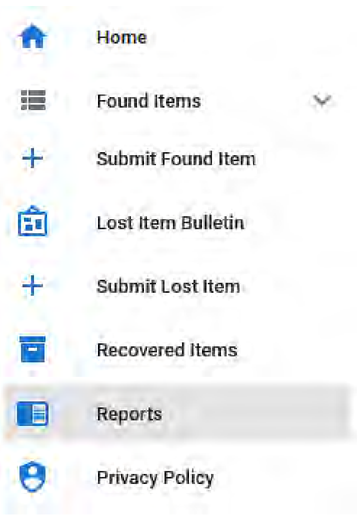
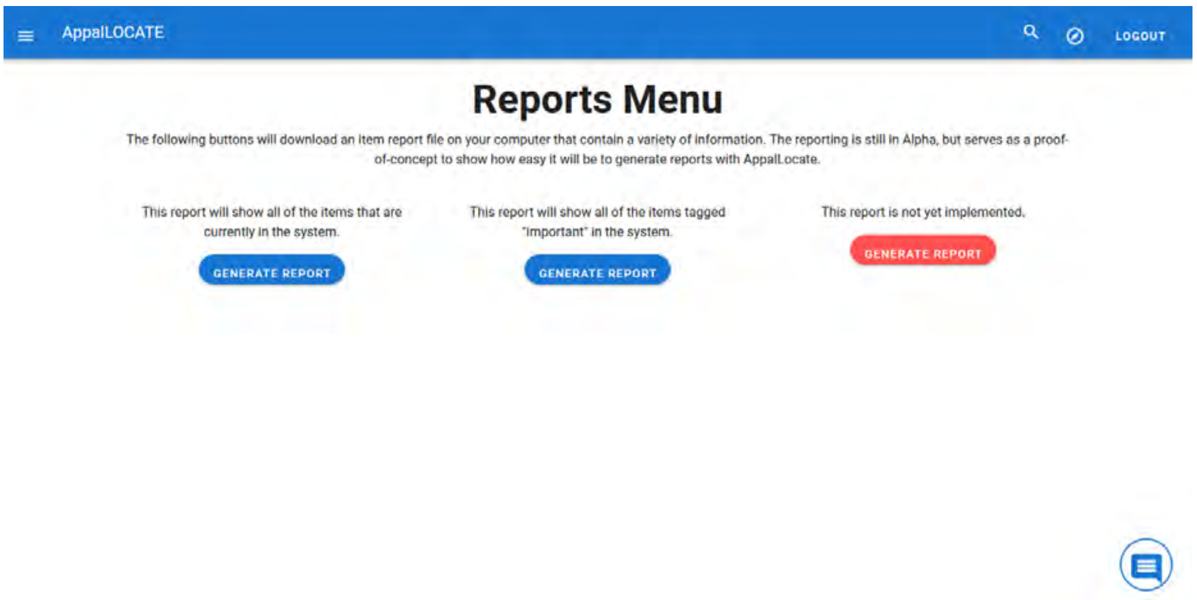


Fig. 35. Reports listing.

AppalLOCATE also allows for office users to generate reports about the stored items from the "Reports" page (Fig. 35). This page (Fig. 36) currently has two types of reports defined, but can be iterated upon once new types of reports are desired. The reports that can be generated are about all items and items with the "important" tag. Regardless, the two reports have the same format, consisting of showing summarized information about a list of the items, the total number of items, and their total estimated cost. An example report can be seen in Fig. 37.



Reports Menu

The following buttons will download an item report file on your computer that contain a variety of information. The reporting is still in Alpha, but serves as a proof-of-concept to show how easy it will be to generate reports with AppalLocate.

This report will show all of the items that are currently in the system.

GENERATE REPORT

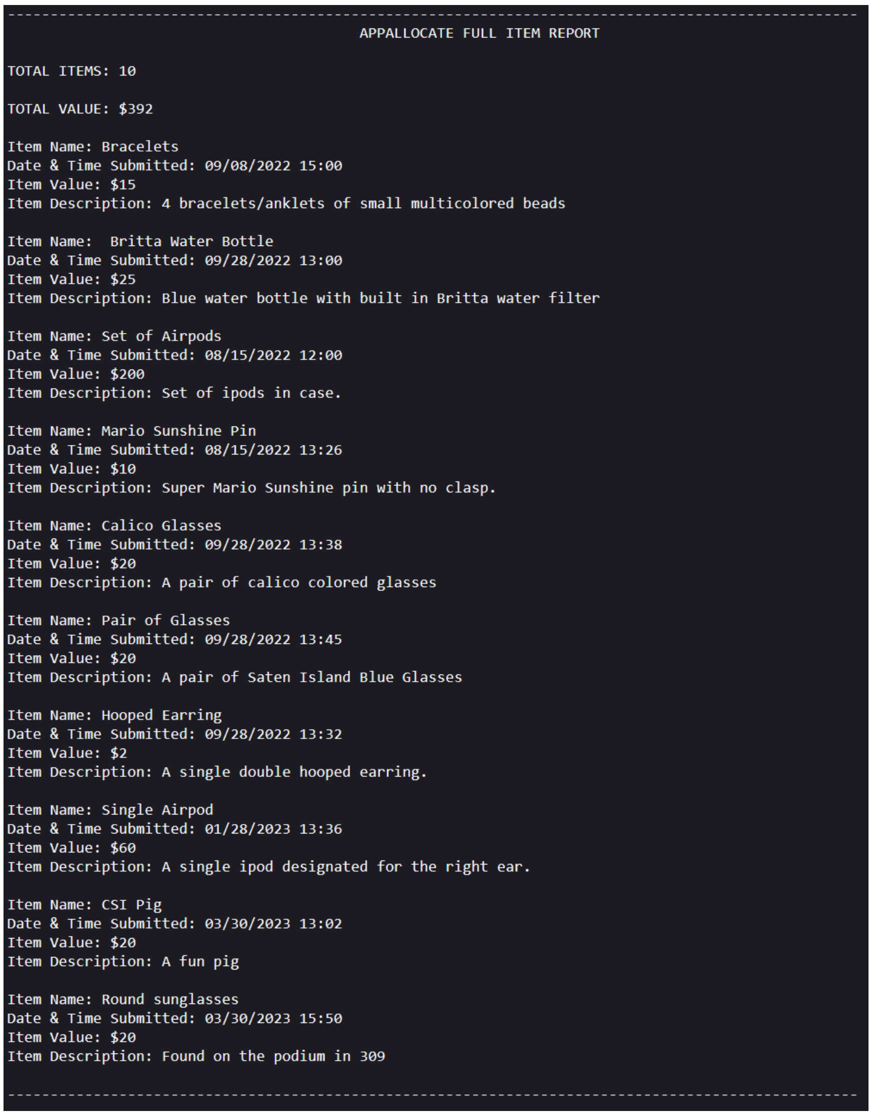
This report will show all of the items tagged "important" in the system.

GENERATE REPORT

This report is not yet implemented.

GENERATE REPORT

Fig. 36. Reports page.



```
-----------------------------------------------------------------------------------------------
                                    APPALLOCATE FULL ITEM REPORT

TOTAL ITEMS: 10

TOTAL VALUE: $392

Item Name: Bracelets
Date & Time Submitted: 09/08/2022 15:00
Item Value: $15
Item Description: 4 bracelets/anklets of small multicolored beads

Item Name:  Britta Water Bottle
Date & Time Submitted: 09/28/2022 13:00
Item Value: $25
Item Description: Blue water bottle with built in Britta water filter

Item Name: Set of Airpods
Date & Time Submitted: 08/15/2022 12:00
Item Value: $200
Item Description: Set of ipods in case.

Item Name: Mario Sunshine Pin
Date & Time Submitted: 08/15/2022 13:26
Item Value: $10
Item Description: Super Mario Sunshine pin with no clasp.

Item Name: Calico Glasses
Date & Time Submitted: 09/28/2022 13:38
Item Value: $20
Item Description: A pair of calico colored glasses

Item Name: Pair of Glasses
Date & Time Submitted: 09/28/2022 13:45
Item Value: $20
Item Description: A pair of Saten Island Blue Glasses

Item Name: Hooped Earring
Date & Time Submitted: 09/28/2022 13:32
Item Value: $2
Item Description: A single double hooped earring.

Item Name: Single Airpod
Date & Time Submitted: 01/28/2023 13:36
Item Value: $60
Item Description: A single ipod designated for the right ear.

Item Name: CSI Pig
Date & Time Submitted: 03/30/2023 13:02
Item Value: $20
Item Description: A fun pig

Item Name: Round sunglasses
Date & Time Submitted: 03/30/2023 15:50
Item Value: $20
Item Description: Found on the podium in 309

-----------------------------------------------------------------------------------------------
```

Fig. 37. Full item report.

V. USER FEEDBACK

A formal usability test was not conducted for AppalLOCATE, but four office users and standard users each were asked for feedback on the application. This chapter describes the feedback given for both standard and office use.

Standard users thought that the application was relatively easy to navigate and use. Standard users had a high likelihood of using the application often for looking for their lost items on campus. Standard users preferred using AppalLOCATE on mobile devices as opposed to the web version, which is to be expected due to convenience.

Office users found the layout and presentation of the application easy to understand and use. The amount that office users would use the application differed depending on office location, but all thought that the application would see moderate if not frequent use. Surprisingly, office users also preferred the mobile version of AppalLOCATE over the web version. This was because of the increased ease of being able to take pictures of items with mobile phones, but for other lost and found inventory management tasks, the web version was preferred.

VI. CONCLUSION

This thesis has described two fully developed versions of AppalLOCATE that can be distributed on a wide variety of operating systems. Users can report both items they have lost and items they have found to the application. This will allow the Appalachian State University community to help each other and increase the number of items that are returned to their owners.

AppalLOCATE is also designed to be a tool used by offices to keep track of their lost and found inventory. The Appalachian Police Department has begun using AppalLOCATE to monitor its inventory, which coupled with the ability to create reports on the items recovered, will allow them to more accurately and efficiently take action on these items.

A. Future Work

While AppalLOCATE is now in production, some features could still be added to the application. These additional features will make AppalLOCATE easier to use for all types of users and help prolong the development of the product. Other features could also be discovered upon more widespread usage of the application.

Some of the improvements that have been considered are discussed further below:

- **Load balancing on busy servers**
  While AppalLOCATE is currently not being used by thousands of users across campus, it has been developed with this possibility in mind. One consideration for this is to load balance the application. Load balancing is the process of redirecting users to different

instances of the application on different servers to lessen the amount of stress on a single server. This can help increase responsiveness for the end users and make using the application a better experience. Load balancing is possible through both Netlify and Oracle Cloud, so both AppalLOCATE and the Typesense server have been prepared to be load balanced, but it has not been activated.

- **Proper searches for all item lists**
  Currently, the search bar in AppalLOCATE only controls the main found items list. This is due to the bulletin board and recovered items list not needing to be searched quite as thoroughly. However, it may be beneficial in the future for the search bar to work with both of these lists as well once more items have been submitted so that users can search for any particular item(s).

- **Automatic indexing**
  Since the Typesense server and Firebase backend are not directly connected, items have to be sent to each individual with two API calls. It may be beneficial in the future to create a process that will index relevant items into Typesense when a search is made. This will lessen the number of items that have to be indexed at once and not waste space on items that are never searched for.

- **Populating tags list with relevant suggestions**
  While currently the tags recommended to users are hardcoded into the application, it would make sense for an algorithm to eventually suggest relevant tags to a user based on the item they are submitting. For example, if a user has input information about a blue book but has not yet entered tags for the item, it could be beneficial to suggest the tags “blue” and “book” to the user.

- **Linking items from the bulletin board and found inventory**
  Since we have records of all of the found items when users are adding lost items to the bulletin board, AppalLOCATE could give possible suggested links between a lost item a user is about to submit and an item already present in the system. This will prevent the user from submitting an item that might already be present somewhere in the system as

well as point the user in the right direction on where their item is.

- **Make login session persist**

  If an office user logs into AppalLOCATE, they will only be logged into that particular instance of the application. If they refresh or close the application, when it is next opened it will not know who was logged in previously. Cookies are a way to store small pieces of session information about websites in browsers and could be used to store users' login information temporarily so they do not have to log in as often.

- **Make images editable after submission**

  A flaw with the way that images are currently uploaded to items is that they cannot be directly edited after the item has been submitted. Instead, if a user wants to modify the images on an item at all, they will have to upload new images and completely remove the previous images from the item. Ideally, users will be able to individually remove existing images on items and add more images independently of these existing images.

- **Proximity notifications**

  AppalLOCATION currently allows users to search for items within a 200 ft radius of them, but it can be taken a step further by sending push notifications to users' mobile devices when they are passing by a reported item's found location. This will give a user a chance to confirm whether an item is still present in the location when they are already there, lessening the burden on them and helping to keep our location records up-to-date.

- **Utilize Netlify analytics**

  Netlify has built-in analytic tools that can be used to monitor website traffic, location metrics, and other data. This could be useful to utilize in AppalLOCATE to measure how many total users and unique users are using the application and graph these metrics over time.

- **More in-depth user tests**

  Although informal feedback was gathered from users about the usability of AppalLOCATE, more in-depth user tests and feedback would be required to shape the application into the best that it can be. These tests would also quantify the usefulness of certain features that have already been implemented and potentially identify alternative features that would be better suited for the application.

- **Apple App Store deployment**

  While the deployment to the Google Play Store is completed, deployment to Apple's App Store is more difficult. Specifically, Apple has an annual $99 for their Apple Developer Program, which makes it difficult to justify uploading it to the App Store. If funding is provided to develop AppalLOCATE further, this fee could be paid to distribute AppalLOCATE to iOS users.

BIBLIOGRAPHY

[1] Algolia, *Site Search & Discovery powered by AI*, 2012 (en).

[2] Apache Software Foundation, *Apache Cordova*, 2009.

[3] Christian Bach, *Develop and deploy websites and apps in record time*, 2014 (en).

[4] Elastic NV, *Elasticsearch: The Official Distributed Search & Analytics Engine*, 2010 (en-us).

[5] Google, *Firebase*, 2011 (en).

[6] ———, *Google Play Console | Google Play Console*, 2012 (en).

[7] Michał Tomasz Jakóbczyk, *Practical Oracle Cloud Infrastructure: Infrastructure as a Service, Autonomous Database, Managed Kubernetes, and Serverless*, Apress, Berkeley, CA, 2020 (en).

[8] Max Lynch, *ionic-team/capacitor*, Ionic, 2022. original-date: 2017-11-18T21:38:09Z.

[9] Sergey M., *fingerprintjs/fingerprintjs*, Fingerprint, 2023. original-date: 2015-02-11T08:49:54Z.

[10] Kishore Nallan, *typesense/typesense*, Typesense Inc., 2022. original-date: 2017-01-18T08:01:47Z.

[11] NCDHHS, *NCDHHS Launches SlowCOVIDNC Exposure Notification App; Available for Download Today*, 2020 (en).

[12] Nick Patrick, *radarlabs/capacitor-radar*, Radar, 2023. original-date: 2019-08-11T16:14:37Z.

[13] TermsFeed, *TermsFeed Privacy Policy Generator*, 2012 (en).

[14] Carmela Troncoso, Dan Bogdanov, Edouard Bugnion, Sylvain Chatel, Cas Cremers, Seda Gürses, Jean-Pierre Hubaux, Dennis Jackson, James R. Larus, Wouter Lueks, Rui Oliveira, Mathias Payer, Bart Preneel, Apostolos Pyrgelis, Marcel Salathé, Theresa Stadler, and Michael Veale, *Deploying decentralized, privacy-preserving proximity tracing*, Communications of the ACM **65** (August 2022), no. 9, 48–57.

[15] Derek Clark Wilson, *LOST AND FOUND APPLICATION: APPALLOCATE* (July 2022), 66 (en).

VITA

Derek Clark Wilson was born in Wilson, North Carolina on September 1, 2001. They grew up on computers, making pursuing a Computer Science degree seem natural. They entered Appalachian State University in August 2019 after graduating from high school earlier in the year. They graduated with a B.S. in Computer Science from Appalachian State University in 2022 and with an M.S. in Computer Science in 2023. They worked in IT services on campus through both degrees and aspire to continue within the technology field in the future.