UMass Lost and Found

4/11/2025
Owen,Jacob,Roberto,Nathan

UMass-Lost-and-Found Project Vision

Problem: It is common for UMass students to misplace important belongings like UCards, water bottles, and academic utensils. There currently doesn't exist an organized place for students to communicate what items they may have lost or found.

Solution: UMass Lost and Found intends to be the first and best place for students to go to communicate with each other items they are looking for or have found. Our web app is organized and easy to use and provides users with all the tools and information they need to find lost items or share items they found.

UMass-Lost-and-Found Project Vision Continued

Key Features:

- Home page containing all found item posts
- Posting features such as images and GPS location to help users locate items
- Sorting, filtering, and searching options to narrow down posts
- Messaging system for communication between posters and users

Repository: https://github.com/realraft/UMass-Lost-and-Found

Milestone #5: https://github.com/realraft/UMass-Lost-and-Found/milestone/4

UMass-Lost-and-Found Builders

Owen Raftery
Role(s): Project-Manager
Role(s) Description: Will be leading the team during meetings and give instructions on what to code.
Issues from current milestone: Basic SPA implementation, navbar implementation, filtering on homepage, searching functionality, SPA navigation for homepages

Nathan Dennis
Role(s): Notetaker, Front-end developer
Role(s) Description: Will be taking down notes, and recording what my teammates do in regards to the website. Will also offer suggestions and implement them when necessary. Will mainly work on the front-end.
Issues from current milestone: SPA navigation for Admin Page, dynamically load reports from fake server to admin page, report modal for every post on homepage

UMass-Lost-and-Found Builders

Jacob Taylor
Role(s): Time-keeper
Role(s) Description: Will organize meetings, keep track of what's missing, and come up with deadlines to the issues.
Issues from current milestone: SPA navigation for PostedItemPage, dynamically load posts from fake server to home page, multi view ui for every post

Roberto Rubio
Role(s): Communicator
Role(s) Description: Will make sure the Backend and Frontend are syncronized and will code for both.
Issues from current milestone: BasePage implementation, base eventhub and events implementation, SPA navigation for MessagingPage, base implementation for messaging features

Historical Timeline For Milestone 5

- 4/3/25: Added fake server with data requirements
- 4/7/25: Eventhub, Events, and MessagingPage, dynamic loading of posts in HomePageSignedIn functionality added
- 4/8/25: Posting functionality, navbar, searching on HomePageSignedIn added
- 4/9/25: Added sorting on HomePageSignedIn
- 4/10/25: Implemented dynamic loading of messages in MessagingPage, BasePage implemented, SPA template implemented, changed HomePageSignedOut, HomePageSignedIn, and Navbar to load dynamically with JS
- 4/11/25: Fixed MessagingPage dynamic loading, implemented multi view for every listing

Jacob Taylor Work Summary

PR's: https://github.com/realraft/UMass-Lost-and-Found/pull/111 , https://github.com/realraft/UMass-Lost-and-Found/pull/102, https://github.com/realraft/UMass-Lost-and-Found/pull/84, https://github.com/realraft/UMass-Lost-and-Found/pull/64,

Tasks worked on: Initializing the fake server file with dummy post data, created methods to interact with the server(fetching the data), dynamic pages for clicking a post, and when a user creates a new post the new listing is shown on the homepage being dynamically generated.

Ongoing Issues: https://github.com/realraft/UMass-Lost-and-Found/issues/115 , this issue is ongoing as we are refactoring our code base, so its more modular and reduces code duplication.

.

Challengers and Future insights: A particular problem we faced was getting our individual components to work together, it required lots of communication and effort on our ends to make sure they linked correctly. A future issue is implementing our backend as again, we all need to be on the same page when it comes to designing it. For feature posts, we need to integrate a backend database for storing our posts instead of using fake_server.json. We also need to use an API for fetching geo data for a user to mark where he found a lost item. My biggest takeaway from working as a group is making sure we are all on the same page, as one person whos implementation differs forms others can break the whole website.

Feature Demonstration Jacob Taylor

Feature Name: Posted Item Details Page

Description: A dynamic page that displays comprehensive details of a lost or found item, including title, images, date found, location, description, and tags. The page provides an intuitive layout with a "Contact Finder" button that navigates users to a messaging interface. The implementation includes responsive content updates based on post data and proper DOM manipulation for dynamic content rendering.

Point Value: 3 points

Reason: This feature demonstrates several advanced techniques:

1. Dynamic Content Updates: The page dynamically updates its content based on the post data through the updatePost method
2. Multi-View UI: Implements a complex UI with multiple sections (images, details, tags, description)
3. Event-Driven Architecture: Uses an EventHub system for navigation and state management
4. Component-Based Structure: Built using a component-based architecture (extends BasePage)

Feature Name: Dynamic Post Creation

Description: Post creation interface that allows users to create lost and found item listings with multiple fields (title, description, date, tags, anonymous status). The implementation uses an event-driven architecture through the EventHub system to handle form submissions and navigation. When a user submits the form, the data is collected, validated, and published through the event system, which triggers updates across different components. The feature includes dynamic content updates through event propagation.

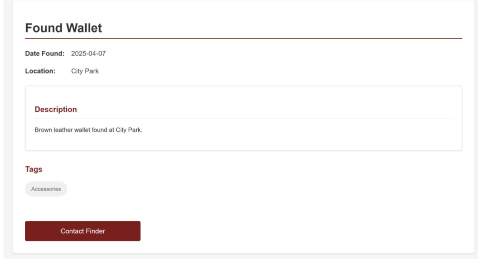
Point Value: 3 points

Reason: This feature demonstrates advanced implementation using three key techniques:

1. Event-Driven Architecture: Uses a EventHub system for managing post creation events and navigation.

2. Dynamic Content Updates: Handles real-time form validation and UI updates, with dynamic field processing

3. Multi-View UI: Implements a complex form interface with multiple interactive sections (image upload, date picker, tag input, description) and handles view transitions through the event system.


Found Wallet

Date Found: 2025-04-07

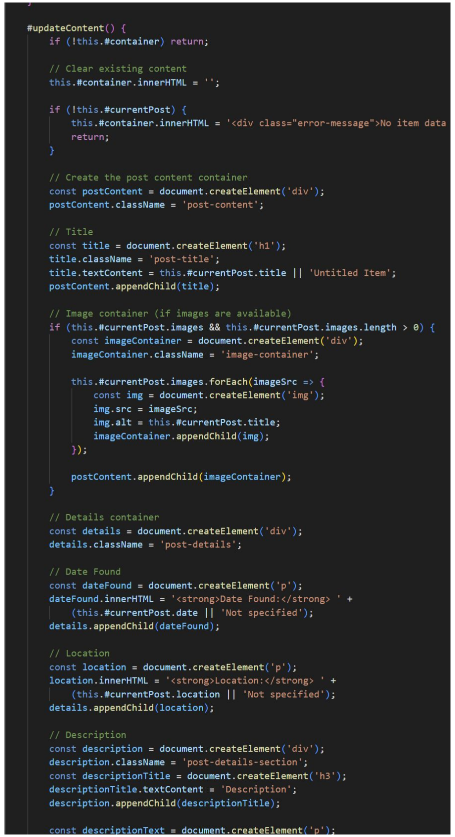
Location: City Park

Description

Brown leather wallet found at City Park.

Tags

Code Explanation Jacob Taylor



```
#updateContent() {
    if (!this.#container) return;

    // Clear existing content
    this.#container.innerHTML = '';

    if (!this.#currentPost) {
        this.#container.innerHTML = '<div class="error-message">No item data
        return;
    }

    // Create the post content container
    const postContent = document.createElement('div');
    postContent.className = 'post-content';

    // Title
    const title = document.createElement('h1');
    title.className = 'post-title';
    title.textContent = this.#currentPost.title || 'Untitled Item';
    postContent.appendChild(title);

    // Image container (if images are available)
    if (this.#currentPost.images && this.#currentPost.images.length > 0) {
        const imageContainer = document.createElement('div');
        imageContainer.className = 'image-container';

        this.#currentPost.images.forEach(imageSrc => {
            const img = document.createElement('img');
            img.src = imageSrc;
            img.alt = this.#currentPost.title;
            imageContainer.appendChild(img);
        });

        postContent.appendChild(imageContainer);
    }

    // Details container
    const details = document.createElement('div');
    details.className = 'post-details';

    // Date Found
    const dateFound = document.createElement('p');
    dateFound.innerHTML = '<strong>Date Found:</strong> ' +
        (this.#currentPost.date || 'Not specified');
    details.appendChild(dateFound);

    // Location
    const location = document.createElement('p');
    location.innerHTML = '<strong>Location:</strong> ' +
        (this.#currentPost.location || 'Not specified');
    details.appendChild(location);

    // Description
    const description = document.createElement('div');
    description.className = 'post-details-section';
    const descriptionTitle = document.createElement('h3');
    descriptionTitle.textContent = 'Description';
    description.appendChild(descriptionTitle);

    const descriptionText = document.createElement('p');
```

This code snippet is located in PostedItemPage/index.js, This file extends the basePage class and implements all the abstract methods. This particular method #updateContent, creates the post page when a user clicks on a listing and generates all the information that a aunique post has, including the description, Date, tags, and content. The impact of this feature is the user can see the posts in its own individual page without having to refresh the page. Ensuring a more smooth user experience. The biggest challenge when implementing this is trying not to break other features while implementing this new one.

Owen Raftery Work Summary

Details: I worked on the base SPA implementation with main.js, app.js, and index.html. I also converted multiple html files to dynamically load with this new implementation, these files are HomePageSignedOut, HomePageSignedIn, and postItemPage. I also implemented filtering and sorting and searching of listings on the home page. Lastly, I implemented the nav bar at the top of the web pages.

PRs:

- fixed searching from pages that arent HomePageSignedIn by realraft · Pull Request #114 · realraft/UMass-Lost-and-Found
- Simplify by realraft · Pull Request #119 · realraft/UMass-Lost-and-Found
- Navbar nav by realraft · Pull Request #109 · realraft/UMass-Lost-and-Found
- Finished most of convert to template by realraft · Pull Request #103 · realraft/UMass-Lost-and-Found
- 66 task bar by realraft · Pull Request #81 · realraft/UMass-Lost-and-Found

Owen Raftery Feature Demonstration

Searching and Filtering:

- Searching and filtering of listings on the home page
- Advanced Feature (ABCDF)
- All intended parts of the feature are complete and functional
- See branches 66-task-bar, convert-to-template, and search-fix

NavBar:

- Nav bar that sits at the top of pages for navigating the site and searching
- Advanced Feature (ABCD)
- All intended parts of the feature are complete and functional, navbar is waiting on the creation of one additional page
- See branches 66-task-bar, convert-to-template, and search-fix

Owen Raftery Code Explanation

This NavBar uses the EventHub to publish a navigation event which allows the user to navigate through the web page. When searching the same happens except a search parameter is added to the URL for HomePageSignedIn to interpret and use to display the users search. The largest struggle I faced while implementing this feature was implementing the search feature to work from pages other than the home page. I was unfamiliar with strategies around changing the URL and had to learn them through online resources.

```
import { BasePage } from
"../../pages/BasePage/BasePage.js";
import { EventHub } from "../../eventHub/EventHub.js";
import { Events } from "../../eventHub/Events.js";

export class NavBar extends BasePage {
  #container = null;

  constructor() {
    super();
    const pathPrefix = this.getPathPrefix();
    this.loadCSS(`${pathPrefix}components/navbar`,
"navbar");
  }

  getPathPrefix() {
    return window.location.pathname.includes('/pages/')
? '../../' : '';
  }

  render() {
    if
(window.location.href.includes('HomePageSignedOut')) {
      return null;
    }

    if (this.#container) {
      return this.#container;
    }

    this.#container = document.createElement('div');
    this.#setupContainerContent();
    this.#attachEventListeners();

    return this.#container;
  }

  #setupContainerContent() {
    if (!this.#container) return;

    this.#container.innerHTML = `
      <div class="navbar">
        <div class="navbar-brand">
          <a href="#" style="text-decoration: none;
color: white;" class="home-link">
            <h1>UMass Lost and Found</h1>
          </a>
        </div>
        <div class="navbar-search">
          <form class="search-form" action="/search"
method="get">
            <input type="search" name="search"
placeholder="Search…" aria-label="Search" />
            <button type="submit">🔍</button>
          </form>
        </div>
        <div class="navbar-actions">
          <a href="#" class="nav-link">
            <button class="post-button">Post</button>
          </a>
          <div class="dropdown">
            <button class="dropdown-button">☰</button>
            <div class="dropdown-content">
              <a href="#"
class="messaging-link">Messaging</a>
              <a href="#" class="disabled-link">Post
Manager</a>
              <a href="#" class="disabled-link">Admin
Page</a>
              <a href="#" class="logout-link">Log
Out</a>
            </div>
          </div>
        </div>
      </div>
```

```
  #setupDropdown() {
    const dropdownButton =
this.#container.querySelector('.dropdown-button');
    const dropdownContent =
this.#container.querySelector('.dropdown-content');

    if (dropdownButton && dropdownContent) {
      dropdownButton.addEventListener('click',
function(e) {
        e.stopPropagation();
        dropdownContent.classList.toggle('show');
      });

      document.addEventListener('click', function() {
        if (dropdownContent.classList.contains('show'))
{
          dropdownContent.classList.remove('show');
        }
      });

      dropdownContent.addEventListener('click',
function(e) {
        e.stopPropagation();
      });
    }
  }

  #setupSearch() {
    const searchForm =
this.#container.querySelector('.search-form');
    const hub = EventHub.getEventHubInstance();

    if (searchForm) {
      searchForm.addEventListener('submit', function(e)
{
        e.preventDefault();

        const searchInput =
this.querySelector('input[type="search"]');
        const searchQuery = searchInput.value.trim();

        if (searchQuery) {
          hub.publish(Events.NavigateTo,
`/HomePageSignedIn?search=${encodeURIComponent(searchQu
ery)}`);
          searchInput.value = '';
        }
      });
    }
  }

  #setupNav() {
    const hub = EventHub.getEventHubInstance();

    // Set up navigation links
    const navLinks = [
      { selector: '.home-link', path:
'/HomePageSignedIn' },
      { selector: '.post-button', path: '/PostItemPage'
},
      { selector: '.messaging-link', path:
'/MessagingPage' },
      { selector: '.logout-link', path:
'/HomePageSignedOut' }
    ];

    navLinks.forEach(link => {
      const element =
this.#container.querySelector(link.selector);
      if (element) {
        element.addEventListener('click', (event) => {
          event.preventDefault();
          hub.publish(Events.NavigateTo, link.path);
        });
```

Owen Raftery Challenges and Insights

The main reflection on this milestone is the importance of reading and following directions. To add to that, it is crucial to truly understand the directions you think you are following. Our team did not prepare properly for the shift to SPA implementation and were left scrambling to fix it last minute, which we will attempt to remedy by a more careful reading of guidelines and directions in the future. A takeaway from working in a team environment is that sometimes working on something is gonna break someone else's feature and conversations need to be had and planned for around that fact.

Owen Raftery Future Improvements and Next Steps

In the future, a post management page is required for users to delete and edit posts they have made in the past. Also, making CSS more consistent across pages will contribute to the overall professionalism of the app.

Implement post management page · Issue #76 · realraft/UMass-Lost-and-Found

Make CSS consistent across the web app · Issue #75 · realraft/UMass-Lost-and-Found

Roberto’s Work Summary

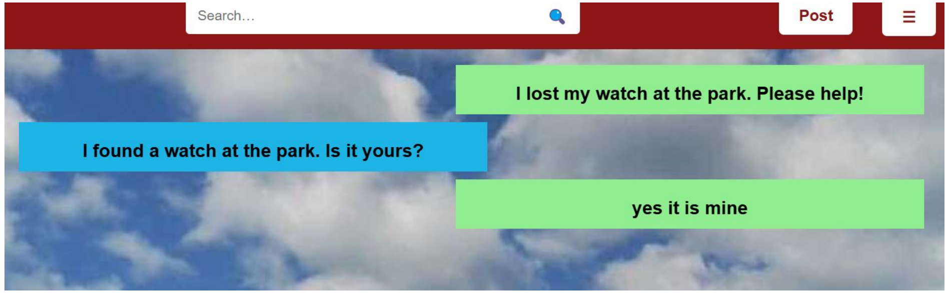
I worked on the following tasks: implemented Service, BasePage, and EventHub, all for the multiview architecture; implemented MessagingService to connect the messaging page to IndexedDB; used the fake server to store conversations between users; and made the Messaging Page dynamic, so that users can change between conversations, go to the posts page, and submit new messages.	https://github.com/realraft/UMass-Lost-and-Found/pull/121 https://github.com/realraft/UMass-Lost-and-Found/pull/113 https://github.com/realraft/UMass-Lost-and-Found/pull/96 https://github.com/realraft/UMass-Lost-and-Found/pull/95 https://github.com/realraft/UMass-Lost-and-Found/pull/78 https://github.com/realraft/UMass-Lost-and-Found/pull/61 https://github.com/realraft/UMass-Lost-and-Found/issues/50 https://github.com/realraft/UMass-Lost-and-Found/issues/47

Roberto’s Feature Demonstration

The feature I implemented for the messaging page contains the following front-end techniques: **A, C, D, E,** and **F.** The feature is Advanced because it does basic fetch() TO A server to retrieve information, it is capable of dynamically changing between chats and posts, it follows the multiview architecture, it handles a few user interactions. What’s missing is better storing of current messages on IndexedDB, however I did set up the service if it is ever needed for the messaging page, or other part of the website.

There are several branches I worked on for this feature, but the most important are: messagingpagefucntionality, messagingpagetesting, and messagingpagedynamicfeature

Roberto’s Code Explanation



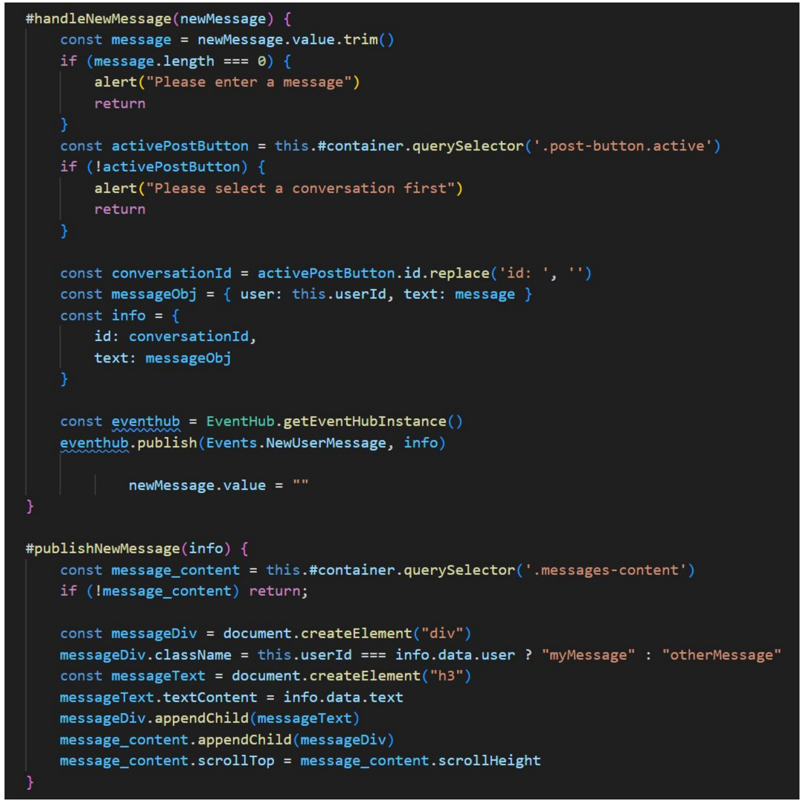
I lost my watch at the park. Please help!

I found a watch at the park. Is it yours?

yes it is mine

The code handles the creation of a new message, by rendering it in the correct conversation, calling eventhub for storage, and prompting the user to not send an empty message.

A challenge I faced was only creating html elements for the messages of a specific conversation between the user looking at the website and the user that posted the lost item.



```
#handleNewMessage(newMessage) {
    const message = newMessage.value.trim()
    if (message.length === 0) {
        alert("Please enter a message")
        return
    }
    const activePostButton = this.#container.querySelector('.post-button.active')
    if (!activePostButton) {
        alert("Please select a conversation first")
        return
    }

    const conversationId = activePostButton.id.replace('id: ', '')
    const messageObj = { user: this.userId, text: message }
    const info = {
        id: conversationId,
        text: messageObj
    }

    const eventhub = EventHub.getEventHubInstance()
    eventhub.publish(Events.NewUserMessage, info)

            newMessage.value = ""
}

#publishNewMessage(info) {
    const message_content = this.#container.querySelector('.messages-content')
    if (!message_content) return;

    const messageDiv = document.createElement("div")
    messageDiv.className = this.userId === info.data.user ? "myMessage" : "otherMessage"
    const messageText = document.createElement("h3")
    messageText.textContent = info.data.text
    messageDiv.appendChild(messageText)
    message_content.appendChild(messageDiv)
    message_content.scrollTop = message_content.scrollHeight
}
```

Roberto’s Challenges and Insights

I learned that it is not easy to create a functional website. My feature took a very long time to code, but with help of some of my teammates I was able to add a meaningful section.

I faced many challenges, but the most important where: getting the fake server to work, implementing IndexedDB, and changing between message pages for different posts.

As a team, the hardest problem we had to overcome was changing to multiview architecture. We had to get rid of so many files and change all html to .js injections. Also we had to connect the pages with eventhub, and the messaging page wasn’t being properly loaded because of some problem with the instantiation of the page object.

Roberto’s Future Improvements and Next Steps

In later milestones we should add, a working server containing posts, messages, images, reports, and any other objects needed. We should also add better communication between pages with the use of custom events and eventhub.

Some aspects that could be improved are: the use of IndexedDB for current messages and user preferences.

For back-end, I will keep working on the messaging page and will have to develop features that connect it to the server and APIs.

https://github.com/realraft/UMass-Lost-and-Found/issues/123

https://github.com/realraft/UMass-Lost-and-Found/issues/124

https://github.com/realraft/UMass-Lost-and-Found/issues/125

Nathan’s Work Summary

https://github.com/realraft/UMass-Lost-and-Found/issues/126
https://github.com/realraft/UMass-Lost-and-Found/issues/127
https://github.com/realraft/UMass-Lost-and-Found/issues/129
https://github.com/realraft/UMass-Lost-and-Found/issues/128
https://github.com/realraft/UMass-Lost-and-Found/issues/130

I worked on the following:

-Allowed access to page using nav bar

-Generated posts and reports to be generated in the Administration page

-Ability to keep and delete posts from the Admin Page, using a modal (will be done to the Home Page listings when backend is implemented due to necessity of actual server)

-Comments on each modal to be stored for each user (not currently working, but will be imple,ented in the backend)

Nathan’s Feature Demonstration

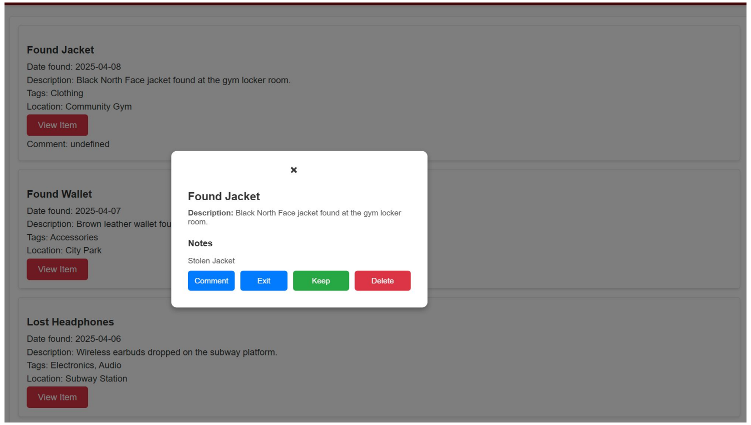
- Asynchronous reading of server for two separate objects: Using both Posts and Reports for an object of the same ID

-Multiple UI interfaces

-Dynamic content updates, for keeping or deleting a post, after it was read from the server.

- Attempted usage of IndexedDB to generate a local post that is not bound to a server in the modals. It will be retained for a future milestone, utilizing the back-end for implementation

Nathan’s Code Explanation (Screenshots of site)

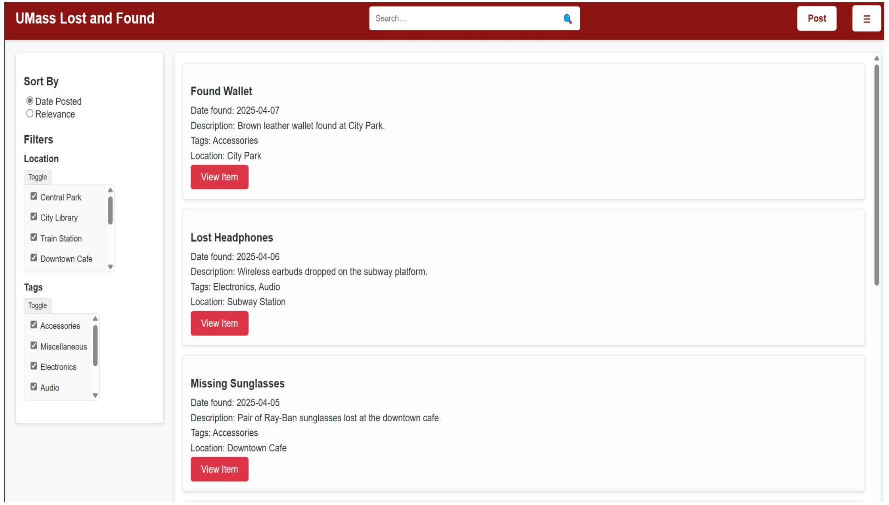


Found Jacket
Date found: 2025-04-08
Description: Black North Face jacket found at the gym locker room.
Tags: Clothing
Location: Community Gym
Comment: undefined

Found Wallet
Date found: 2025-04-07
Description: Brown leather wallet fou
Tags: Accessories
Location: City Park

Lost Headphones
Date found: 2025-04-06
Description: Wireless earbuds dropped on the subway platform.
Tags: Electronics, Audio
Location: Subway Station

Found Jacket
Description: Black North Face jacket found at the gym locker room.
Notes
Stolen Jacket



UMass Lost and Found

Sort By
(●) Date Posted
( ) Relevance

Filters
Location
☑ Central Park
☑ City Library
☑ Train Station
☑ Downtown Cafe

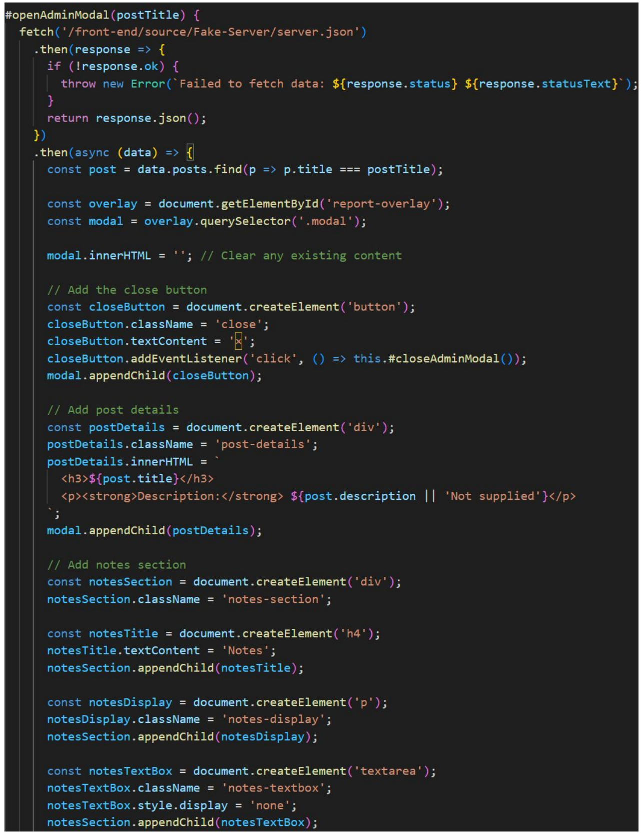
Tags
☑ Accessories
☑ Miscellaneous
☑ Electronics
☑ Audio

Found Wallet
Date found: 2025-04-07
Description: Brown leather wallet found at City Park.
Tags: Accessories
Location: City Park

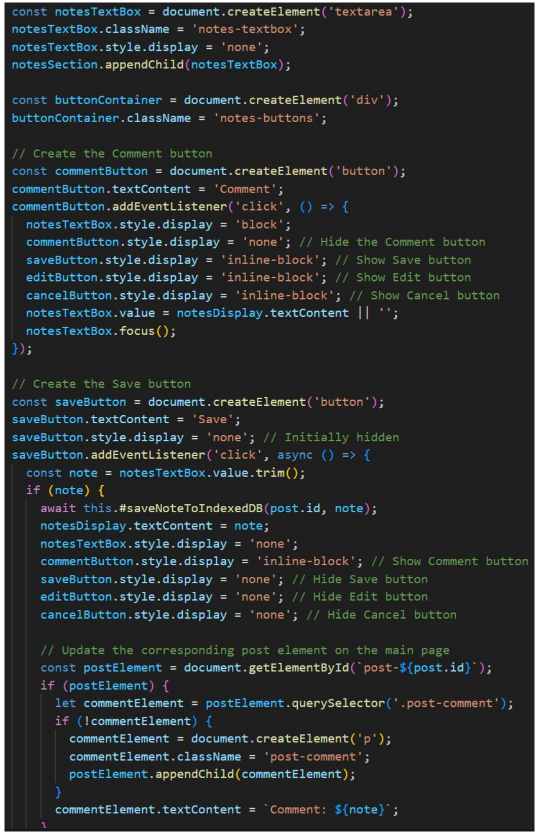
Lost Headphones
Date found: 2025-04-06
Description: Wireless earbuds dropped on the subway platform.
Tags: Electronics, Audio
Location: Subway Station

Missing Sunglasses
Date found: 2025-04-05
Description: Pair of Ray-Ban sunglasses lost at the downtown cafe.
Tags: Accessories
Location: Downtown Cafe

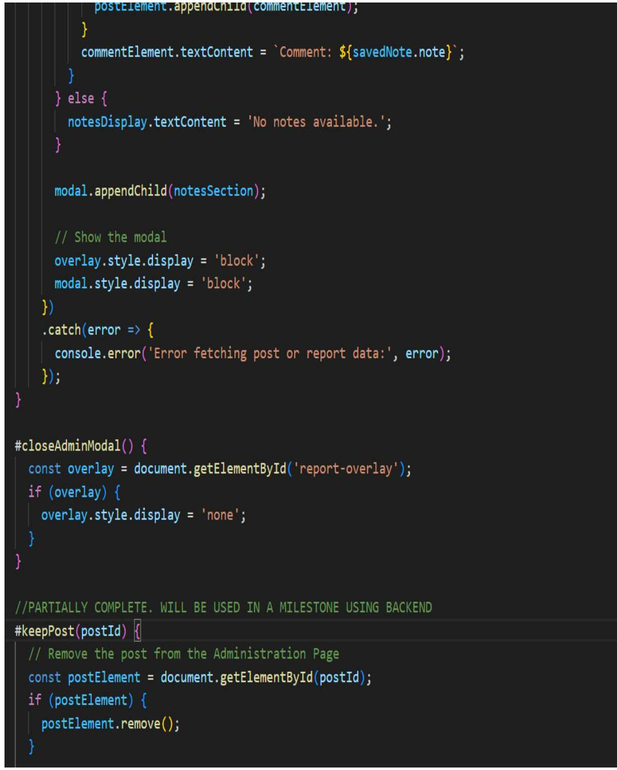
Nathan Code Explanation (Code Screenshots)



```
#openAdminModal(postTitle) {
  fetch('/front-end/source/Fake-Server/server.json')
    .then(response => {
      if (!response.ok) {
        throw new Error(`Failed to fetch data: ${response.status} ${response.statusText}`);
      }
      return response.json();
    })
    .then(async (data) => {
      const post = data.posts.find(p => p.title === postTitle);

      const overlay = document.getElementById('report-overlay');
      const modal = overlay.querySelector('.modal');

      modal.innerHTML = ''; // Clear any existing content

      // Add the close button
      const closeButton = document.createElement('button');
      closeButton.className = 'close';
      closeButton.textContent = '×';
      closeButton.addEventListener('click', () => this.#closeAdminModal());
      modal.appendChild(closeButton);

      // Add post details
      const postDetails = document.createElement('div');
      postDetails.className = 'post-details';
      postDetails.innerHTML = `
        <h3>${post.title}</h3>
        <p><strong>Description:</strong> ${post.description || 'Not supplied'}</p>
      `;
      modal.appendChild(postDetails);

      // Add notes section
      const notesSection = document.createElement('div');
      notesSection.className = 'notes-section';

      const notesTitle = document.createElement('h4');
      notesTitle.textContent = 'Notes';
      notesSection.appendChild(notesTitle);

      const notesDisplay = document.createElement('p');
      notesDisplay.className = 'notes-display';
      notesSection.appendChild(notesDisplay);

      const notesTextBox = document.createElement('textarea');
      notesTextBox.className = 'notes-textbox';
      notesTextBox.style.display = 'none';
      notesSection.appendChild(notesTextBox);
```



```
const notesTextBox = document.createElement('textarea');
notesTextBox.className = 'notes-textbox';
notesTextBox.style.display = 'none';
notesSection.appendChild(notesTextBox);

const buttonContainer = document.createElement('div');
buttonContainer.className = 'notes-buttons';

// Create the Comment button
const commentButton = document.createElement('button');
commentButton.textContent = 'Comment';
commentButton.addEventListener('click', () => {
  notesTextBox.style.display = 'block';
  commentButton.style.display = 'none'; // Hide the Comment button
  saveButton.style.display = 'inline-block'; // Show Save button
  editButton.style.display = 'inline-block'; // Show Edit button
  cancelButton.style.display = 'inline-block'; // Show Cancel button
  notesTextBox.value = notesDisplay.textContent || '';
  notesTextBox.focus();
});

// Create the Save button
const saveButton = document.createElement('button');
saveButton.textContent = 'Save';
saveButton.style.display = 'none'; // Initially hidden
saveButton.addEventListener('click', async () => {
  const note = notesTextBox.value.trim();
  if (note) {
    await this.#saveNoteToIndexedDB(post.id, note);
    notesDisplay.textContent = note;
    notesTextBox.style.display = 'none';
    commentButton.style.display = 'inline-block'; // Show Comment button
    saveButton.style.display = 'none'; // Hide Save button
    editButton.style.display = 'none'; // Hide Edit button
    cancelButton.style.display = 'none'; // Hide Cancel button

    // Update the corresponding post element on the main page
    const postElement = document.getElementById(`post-${post.id}`);
    if (postElement) {
      let commentElement = postElement.querySelector('.post-comment');
      if (!commentElement) {
        commentElement = document.createElement('p');
        commentElement.className = 'post-comment';
        postElement.appendChild(commentElement);
      }
      commentElement.textContent = `Comment: ${note}`;
```



```
          postElement.appendChild(commentElement);
        }
        commentElement.textContent = `Comment: ${savedNote.note}`;
      }
    } else {
      notesDisplay.textContent = 'No notes available.';
    }

    modal.appendChild(notesSection);

    // Show the modal
    overlay.style.display = 'block';
    modal.style.display = 'block';
  })
  .catch(error => {
    console.error('Error fetching post or report data:', error);
  });
}

#closeAdminModal() {
  const overlay = document.getElementById('report-overlay');
  if (overlay) {
    overlay.style.display = 'none';
  }
}

//PARTIALLY COMPLETE. WILL BE USED IN A MILESTONE USING BACKEND
#keepPost(postId) {
  // Remove the post from the Administration Page
  const postElement = document.getElementById(postId);
  if (postElement) {
    postElement.remove();
  }
```

Nathan’s Code Explanation (Comments)

This code is what draws a modal to the screen, whenever the View Item button is pressed. Each modal has the item name, an explanation for a report, and the option for the administrator to add a note (with the buttons appearing or disappearing depending on the text box), and having the note be appended to the modal with the text box disappearing (intended to use IndexedDB for this feature, but it doesn’t keep retention possibly due to the nature of dynamic posts in general). There is the option to keep and delete the posts, which will be implemented within the next two milestones. When being press “Keep” or “Delete”, would remove the post from the Administrator side, but will be utilized to affect whether a post is kept or deleted from the listings page.

Nathan’s Challenges and Insights

Not so much with this code itself, but rather organization skills, as we barely had enough time to finish our project today, before the due-date, because our team had a misunderstanding about how to do the project to begin with. For me, personally, still trying to get to grips on how to use Github with VSCode, due to its very (and perhaps unnecessarily) difficult setup, especially for both used in tandem.

Nathan’s Future Improvements and Insights

https://github.com/realraft/UMass-Lost-and-Found/issues/128

https://github.com/realraft/UMass-Lost-and-Found/issues/130

These two are only partially implemented, due to limitations and time constraints. Will be altering the dynamics of a copied post from another page (to keep or delete), from the Administrator’s side (which requires a back-end), and will likely use a server to keep comments created in each post, instead of trying to do it via IndexedDB. Will also utilize images, perhaps GPS coordinates, and, if it’s manageable, customized events for scenarios not thought of.