Progressive Web Application Development For Use In The TCNJ Library Application
William Roche
Submitted in partial fulfillment of the requirements of CSC298 Mentored Research
5/3/2024

Abstract:
This mentored research project continued the development of a Progressive Web Application to augment the TCNJ Library mobile application, specifically focusing on refining study room motion sensors and implementing a printer and computer finder feature. The PWA continued to integrate real-time data communication with a backend database, as it had in the first semester, regarding study room availability as well as computer availability. Varying inaccuracies within the study room data were addressed, which were primarily caused by Raspberry Pi and motion sensor issues. After fixing these problems, we moved to the development of more features within the PWA, specifically implementing the computer and printer finder features. These features would allow for a more complete and practical experience with the library app, matching the features of the current, non PWA application. The languages used in these features include HTML, CSS, and Javascript. Appendices are attached below for context.

Problem Statement:
The goal of this mentored research project was to continue the development of a Progressive Web Application, or PWA, that integrates real-time data and communication with a backend database to enhance services used in the TCNJ Library mobile application for study room, printer, and computer availability. This platform-independent application used tools such as Capacitor, the Ionic framework, and Angular during development. Work with the Ionic framework includes using Javascript, CSS, and HTML. This semester also saw work with troubleshooting Raspberry Pi Zero 2w's paired with a motion sensor. This project seeked to complete the functionality of the TCNJ Library App PWA for student and faculty use, and also serves as an academic exploration of PWA technology.

Background:
The R. Barbara Gitenstein TCNJ Library Application was published on the app store 11 years ago. Since its creation, an expansive list of students has contributed to its development. The application was first developed by Art Malarcyzk and Jon Stoeber as a capstone experience at TCNJ. Updates were implemented in the Spring of 2013 by Frank Corradi, Brandon Gottlob, and Ken Whittaker. These updates were furthered by work done in the Fall of 2014 by Warren Seto and in the Fall of 2016 by Madison Martin. Dual-platform implementation was implemented during a capstone experience by David Shull, assisted by Sophia Goldberg in the Spring of 2017. Additionally, Sophia Goldberg implemented study room occupancy detection from Fall 2017 to Spring 2018, assisted by Randell Corrido and Victoria Swartz. From 2018 to 2019, Ryan Bogutz performed work on backend development and data filtering. During this time, Jenna Oak implemented accessibility, room scheduling algorithms, module design, and database enhancements, and Gabrielle Curcio implemented room scheduling integration. From 2019 to 2020, production rollout, server support, the accuracy of occupancy detection, and computer availability upgrade were completed by Mark Middleton, assisted by Allison Russell in 2019, Lisa Walker from 2019 to 2020, and Kenna Stiesi in 2020. Application WiFi stabilization, updated computer mappings, additional backend development including serviced computers and computer hostnames error reporting, study room searching, Mac occupancy detection, VPN information, and other minor frontend features were implemented by Payton Shaltis from Fall 2021 to Spring 2022, and Mila Manzano in Spring of 2022. Parvathi Krishnan and Will Roche began PWA Implementation with a study room availability feature in the Fall of 2023.

Methodology:
Having learned about the concept of Progressive Web Applications and how to implement one during the previous semester, the main goals of this semester were centered around the upkeep and maintenance of the Raspberry Pi motion sensors, as well as the implementation of a computer finder feature which mapped the positions and statuses of the computers in the library based off of real-time data. During the beginning months of the semester, our primary focus was to confirm that all Raspberry Pis and motion sensors were functioning as intended. Our project implemented the 'Raspberry Pi zero 2 w' coupled with a motion sensor. This is essentially a tiny, affordable, and versatile single-board computer. While higher-end motion sensors may have produced more accurate data, limited funding meant finding a more affordable solution. Essentially, every room has a Raspberry Pi and a motion sensor. These Pis run with programs from an SD card, which return the motion status in the room at any given time. After determining whether or not the sensor senses movement, the room status is returned as occupied or unoccupied and sent into the backend database, wherein each Pis status is displayed with individual IP addresses, Mac addresses, and room numbers. This real-time data is then used in the study room finder feature, so the Raspberry Pi must return accurate values. The second goal of this semester was to complete further implementations of the library app and printer and computer availability. The printer availability feature, when interacted with, will display a static map of all of the printers on the inputted floor of the library. For the computer availability feature, a dynamic map would display computers for the given floor based on the status of their use; for example, a computer not being used would display green, whereas a computer in use would display red. This was done using a canvas with rectangles drawn on top of it. These features employed standard PWA languages: HTML, Javascript, and CSS.

Implementation:
During the beginning weeks of the semester, one of my tasks was to compare the real-time data from the backend to the actual occupancy of the study rooms within the library. When faced with a study room that habitually displayed inaccurate data, it was placed on the list to be taken out for further analysis. The first step to fixing a malfunctioning pi was to plug it into a monitor using a micro HDMI cable, a micro USB cable for a computer or mouse input, and the power cable. (Appendix B) A rainbow or gray loading screen typically indicated a corrupted SD card, meaning one had to be reimaged. (Appendix C) This process involved downloading the Raspberry Pi imager application and either reimagining the existing card or imaging a fresh blank card with a saved GZ file consisting of the scripts needed for the Pi to display data accurately. (Appendix A) Following re-inserting the freshly imaged SD card, the IP address and Mac address would need to be checked on the pi using the command ifconfig in the pi terminal to ensure proper data storage (Appendix D). The motion sensor cables must also be properly connected to the Pi for proper output of the data.(Appendix E,F) If no rainbow or gray loading screen appeared, a reboot of the Pi typically sufficed for troubleshooting, followed by a 15-minute initial delay of the motion sensor. Then, the output is correctly displayed in the backend. (Appendix G) If neither produced a positive result, the following assessment was of the power cable and the soldering of the pins on the Pi itself. Assuming both were adequately connected to the Pi and all other steps had failed, it is to be assumed that the Pi was dead and would not function. Following proper testing and deployment of all Pis, the next goal of the semester was implementing the computer and printer finder features. The printer display feature

simply displayed a static map of the printers in the library on the given floor. (Appendixes N,O,P). As for the computer display feature, there had been a typescript version of the code in the antiquated app, which had to be converted to JavaScript for proper use in the PWA. After having issues with using the original code translated to JavaScript and wanting to learn how it worked, I started from scratch. I began by learning how to use the canvas feature in JavaScript, drawing rectangles on coordinates of a blank canvas, and adjusting its size and ratios. After this, I pulled the maps onto canvases and made cases for each floor. (Appendix I, I2) Having the coordinates of the computers on the maps from the prior implementation of the feature in the antiquated app, I first statically drew the computers onto the maps, which was successful. After reviewing the code for fetching php data from last semester with the study rooms, I reimplemented the feature to fetch and store the raw computer data in an array. (Appendix J) Using this array, I made cases for each floor. Since every floor had a unique computer name associated with it, I simply implemented methods for every floor to run a loop within the array that drew all computers on each respective floor. After this, I implemented if statements based on the status of the computers, drawing red if in use and green if open. (Appendix K) The computer finder feature was fully and correctly implemented after resizing the rectangles and the canvas to conform to the page size with proper ratios.

Results:

This mentored research project completed the Progressive Web Application (PWA) for the TCNJ Library mobile application, marking a significant achievement in our development journey. Key accomplishments include the implementation of the computer and printer finder features, as well as the successful maintenance of Raspberry Pi devices. The primary focus from January 23rd to March 26th consisted of the upkeep and maintenance of the Raspberry Pis. Following the successful deployment of all Raspberry Pis back into the library, from March 26th to the present, the focus shifted to printer and computer availability implementation. Currently, the Raspberry Pis are all functioning correctly, and the printer and computer availability are completed, as shown below. (Appendix P,Q, H) Despite not testing the finished product, the PWA is fully implemented with all the initially desired features. These past two semesters have successfully recreated the deprecated TCNJ Library App as a progressive web application, and any future work will consist of testing and deploying the completed version of the PWA.

Conclusions:

This mentored research project successfully recreated the TCNJ Library mobile application by creating a progressive web application that integrates real-time data, communicating seamlessly with a backend database Throughout this project, I was exposed to various new technologies, namely Raspberry Pis and motion sensors. I primarily focused on learning the concepts that affected our research: imaging SD cards and working inside a Pi. Additionally, I refined my skills with technologies I had been introduced to last semester, such as fetching PHP data and displaying content based on backend data. This marks the completion of the PWA version of the TCNJ library application and essentially concludes the project's developmental stage.

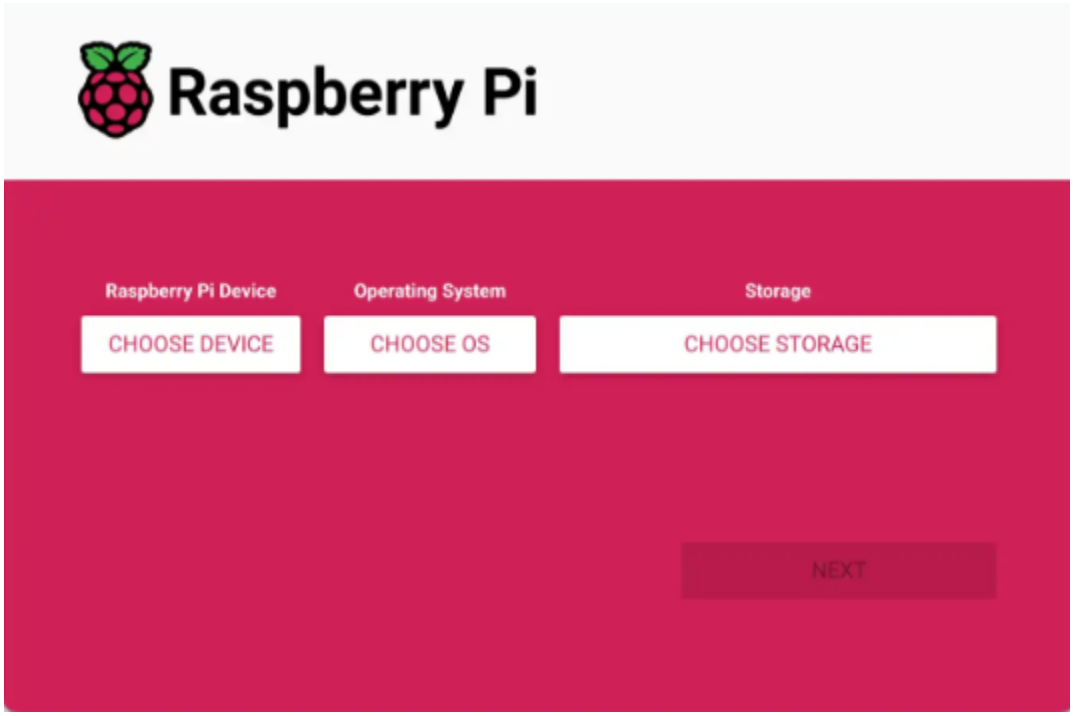
Future Work:
If the PWA version of the TCNJ library app continues development, any further work will likely consist of thorough testing of the various application features and a launch for public use. As of spring 2024, all basic features have been completed, and no systems are deprecated or nonfunctioning.

References:
Raspberry Pi Foundation. 2024. Setting up your Raspberry Pi.
https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2

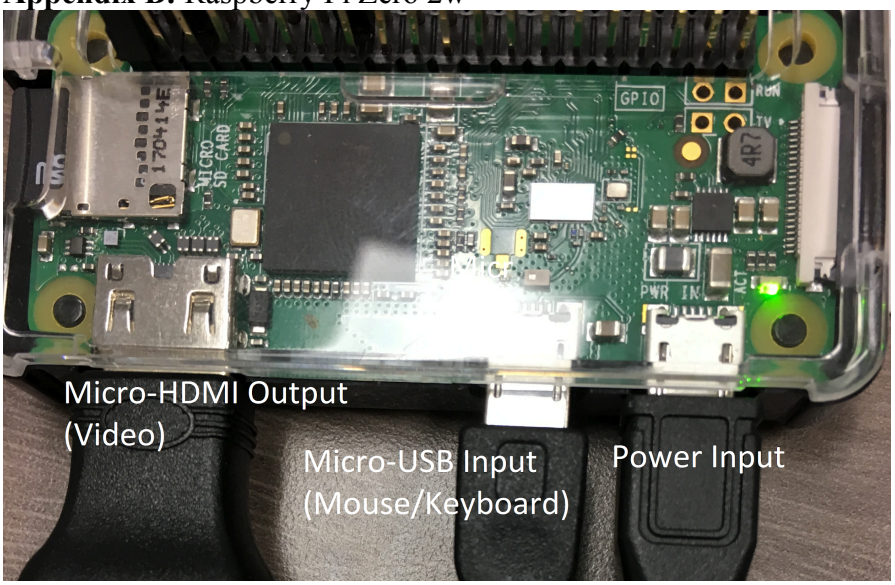
W3Schools. 2024. HTML5 Canvas Tutorial.
https://www.w3schools.com/html/html5_canvas.asp

Acknowledgments
Collaborators:
Parvathi Krishnan [Research Partner]
Deborah Knox [Research Mentor]
Previous students:
Art Malarcyzk, Jon Stoeber, Frank Corradi, Brandon Gottlob, Ken Whittaker, Warren Seto, Madison Martin, David Shull, Sophia Goldberg, Randell Corrido, Victoria Swartz, Ryan Bogutz, Jenna Oak, Gabrielle Curcio, Mark Meddleton, Allison Russell, Lisa Walker, Kenna Sties, Payton Shaltis, Mila Manzano

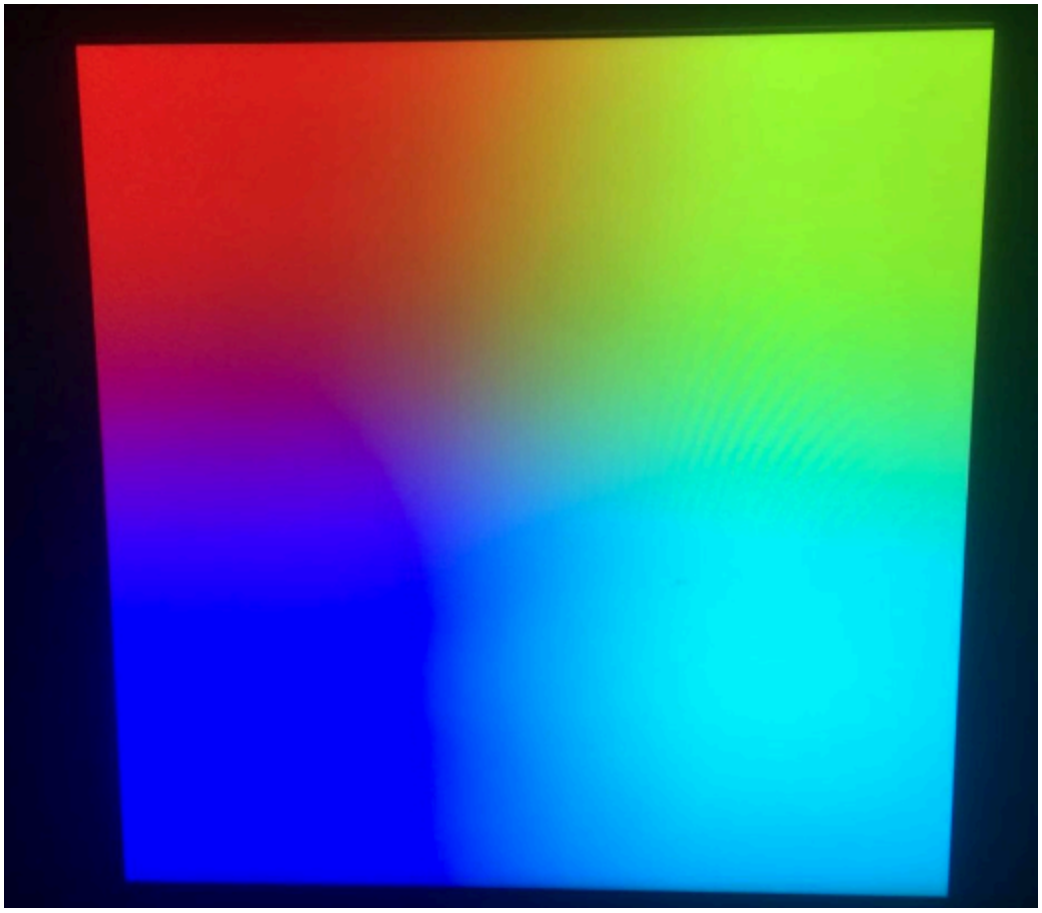
Appendix A: Raspberry Pi Imager



Appendix B: Raspberry Pi Zero 2w



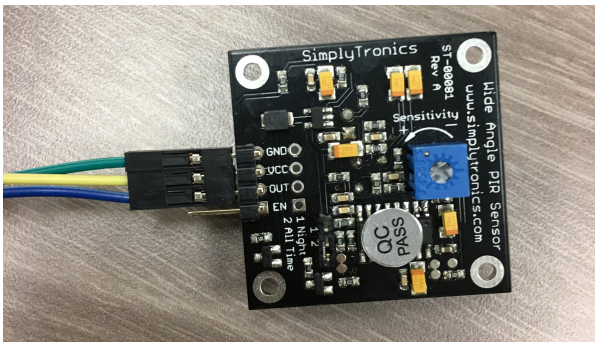
Appendix C: Pi Rainbow error screen



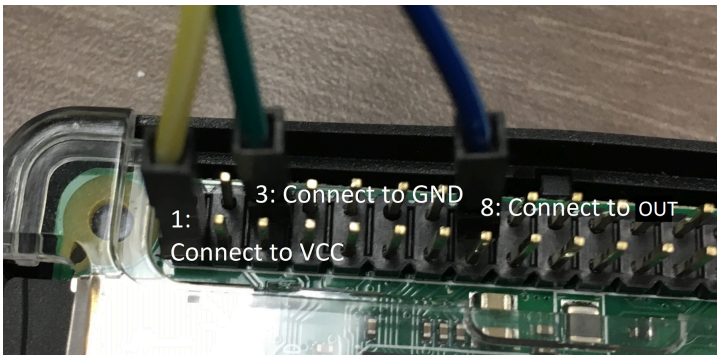
Appendix D: IP and Mac addresses

```
        inet 172.29.145.22  netmask 255.255.224.0  broadcast 172.29.159.255
        inet6 fe80::c91f:34bf:2856:4d2a  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:be:24:4f  txqueuelen 1000  (Ethernet)
        RX packets 7921  bytes 633794 (618.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9313  bytes 1035551 (1011.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Appendix E: Motion sensor



Appendix F: Sensor Connection to pi

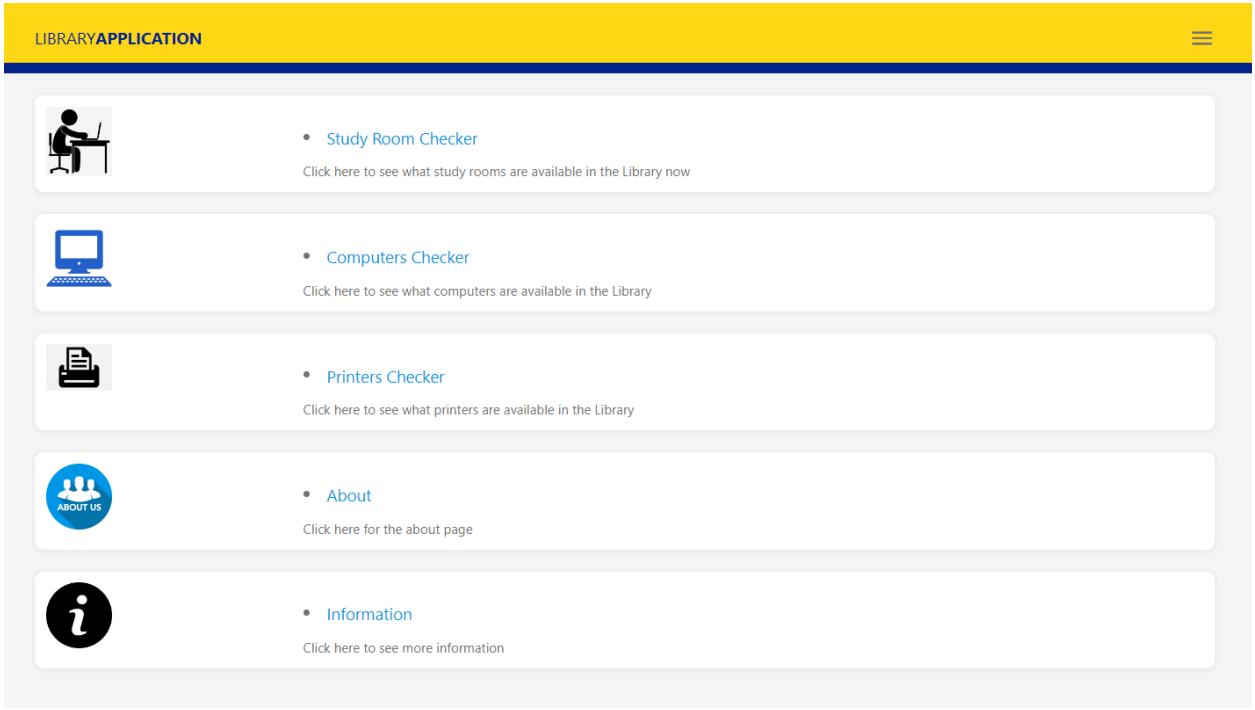


Appendix G: Backend data

```
mysql> select * from studyroomlist;
+-----------------+--------+----------------+-------------------+
| hostname        | status | ip_address     | mac_address       |
+-----------------+--------+----------------+-------------------+
| studyroom_109   |      1 | 172.29.136.52  | b8:27:eb:ea:ae:16 |
| studyroom_110   |      0 | 172.29.133.19  | b8:27:eb:b3:78:8b |
| studyroom_111   |      1 | 172.29.149.83  | b8:27:eb:41:a6:14 |
| studyroom_202   |      1 | 172.29.143.76  | b8:27:eb:39:22:f5 |
| studyroom_205   |      1 | 172.29.143.246 | b8:27:eb:90:c1:8e |
| studyroom_220   |      1 | 172.29.134.254 | b8:27:eb:9b:b9:ff |
| studyroom_220_1 |      1 | 172.29.133.109 | b8:27:eb:ed:9d:8e |
| studyroom_224   |      1 | 172.29.143.233 | b8:27:eb:b8:5c:d5 |
| studyroom_225   |      1 | 172.29.147.95  | b8:27:eb:94:19:12 |
| studyroom_226   |      1 | 172.29.138.151 | b8:27:eb:00:e4:01 |
| studyroom_228   |      1 | 172.29.149.253 | b8:27:eb:49:6e:c0 |
| studyroom_301   |      1 | 172.29.135.5   | b8:27:eb:4b:d6:21 |
| studyroom_308   |      0 | 172.29.151.244 | b8:27:eb:b1:38:2d |
| studyroom_309   |      1 | 172.29.136.113 | b8:27:eb:80:c4:2e |
| studyroom_310   |      1 | 172.29.143.241 | b8:27:eb:93:49:c5 |
| studyroom_311   |      1 | 172.29.144.72  | b8:27:eb:93:c3:90 |
| studyroom_315   |      1 | 172.29.144.193 | b8:27:eb:b6:90:bf |
| studyroom_316   |      1 | 172.29.143.201 | b8:27:eb:95:af:30 |
| studyroom_317   |      1 | 172.29.143.238 | b8:27:eb:be:0d:ac |
| studyroom_319   |      1 | 172.29.137.140 | b8:27:eb:d6:64:b8 |
| studyroom_404   |      1 | 172.29.139.206 | b8:27:eb:a2:5f:92 |
| studyroom_406   |      1 | 172.29.148.213 | b8:27:eb:10:81:6e |
| studyroom_411   |      1 | 172.29.148.134 | b8:27:eb:ba:d6:e8 |
| studyroom_412   |      1 | 172.29.136.88  | b8:27:eb:dc:a0:aa |
| studyroom_413   |      1 | 172.29.144.234 | b8:27:eb:c1:36:0e |
| studyroom_413_1 |      1 | 172.29.143.232 | b8:27:eb:54:d1:88 |
| studyroom_414   |      1 | 172.29.141.108 | b8:27:eb:bb:7a:56 |
| studyroom_415   |      1 | 172.29.148.193 | b8:27:eb:a3:ba:34 |
| studyroom_lab   |      1 | 172.29.151.246 | b8:27:eb:6f:cf:22 |
+-----------------+--------+----------------+-------------------+
29 rows in set (0.00 sec)

mysql>
```

Appendix H: Current application UI



Appendix I: DisplayMap function

```
function displayMap() {
    var floorSelect = document.getElementById("floorSelect");
    var selectedFloor = floorSelect.value;
    var mapCanvas = document.getElementById("canvas");
    var ctx = mapCanvas.getContext("2d");
    noPrintersMessage.style.display = "none"; // Hide the message


    // Clear the canvas before drawing
    ctx.clearRect(0, 0, 600, 387);

    var mapImage = new Image();
    //Setting image source based on selected floor
    switch (selectedFloor) {
        case "lower":
            drawLowerMap(ctx, mapImage);
            break;
        case "first":
            drawFirstMap(ctx, mapImage);
            break;
        case "second":
            drawSecondMap(ctx, mapImage);
            break;
        case "third":
            noPrintersMessage.style.display = "block";
            mapImage.src = "/img/computer/3.png";
            break;
        case "fourth":
            drawFourthMap(ctx, mapImage);
            break;
    }
}
```

Appendix I2: drawing map onto canvas and calling the draw function

```
function drawLowerMap(ctx, mapImage) {
    mapImage.onload = function() {
        // Draw the map image
        ctx.drawImage(mapImage, 0, 0,600, 387);

        // Draw computers on the lower map
        drawLowerComputers(ctx);
    };
    mapImage.src = "/img/computer/0.png";
}
```

Appendix J: loading php file with status data

```
function loadPHPFile() {
    const phpUrl = "http://knoxlablibrary.tcnj.edu/compstatus.php";

    fetch(phpUrl)
        .then(response => {
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            return response.text();
        })
        .then(data => {
            // Split the data into an array of strings separated by spaces

            phpDataArray = [];
            const parser = new DOMParser();
            const xmlDoc = parser.parseFromString(data, "text/xml");
            const computers = xmlDoc.getElementsByTagName("comp");
            for (let i = 0; i < computers.length; i++) {
                const name = computers[i].getElementsByTagName("name")[0].textContent;
                console.log("Computer Name:", name); // Log the computer names

                const status = computers[i].getElementsByTagName("status")[0].textContent;
                const service = computers[i].getElementsByTagName("service")[0].textContent;
                phpDataArray.push({ name, status, service });
            }
        })
        .catch(error => {
            console.error('There was a problem with the fetch operation:', error);
        });
}
```

Appendix K: drawing rectangles on canvas

```
function drawLowerComputers(ctx) {
    // Filter computers belonging to the lower level (LIB2 and LIB5)
    let lowerLevelComputers = phpDataArray.filter(computer => computer.name.startsWith('LIB2') || computer.name.startsWith('LIB5'));

    // Loop through lower level computers and draw them on the canvas in red
    for (let i = 0; i < lowerLevelComputers.length; i++) {
        let computer = lowerLevelComputers[i];
        let status = computer.status;
        let xCoord = [863, 830, 797, 764, 731, 698, 698, 731, 764, 797, 830, 863, 863, 830, 797, 764, 731, 698, 698, 731, 764, 797, 830, 863
        let yCoord = [930, 930, 930, 930, 930, 930, 897, 897, 897, 897, 897, 897, 831, 831, 831, 831, 831, 831, 798, 798, 798, 798, 798, 798

        let scaleFactor = 0.2555; // Adjust this value as needed
        for (let j = 0; j < xCoord.length; j++) {
            xCoord[j] *= scaleFactor;
            yCoord[j] *= scaleFactor;
        }
        // Draw the computer based on its status
        // Set color and size based on status
        if (status === '0') {
            ctx.strokeStyle = "#FF0000"; // Black for status 0
            ctx.strokeRect(yCoord[i], xCoord[i], 5, 5); // Larger rectangle for status 0
        } else if (status === '1') {
            ctx.strokeStyle = "#008000"; // Red for status 1
            ctx.strokeRect(yCoord[i], xCoord[i], 5, 5); // Smaller rectangle for status 1
        }
    }
}
```

Appendix L: canvas HTML

```
<img src="" id="floorMap" >
<p id="noPrintersMessage" style="display:none;">No computers available on this floor</p>
<canvas id="canvas" width="600" height="387"></canvas>
```

Appendix M: Display HTML

```
<main>
  <section id="contact">
      <h2><b><center>Select a floor to view the map  </center></b></h2>
      <br>
      <!-- Dropdown list-->
      <div class="form-group">
        <label for="floorSelect">Floor:</label>
        <select id="floorSelect" name="floorSelect" required>
            <option value="lower">Lower Level</option>
            <option value="first">First Level</option>
            <option value="second">Second Level</option>
            <option value="third">Third Level</option>
            <option value="fourth">Fourth Level</option>
        </select>
    </div>
      <button type="submit" onclick="displayMap()">Display Map</button>

  </section>
```

Appendix N: Static Printer Display

```
function displayMap() {
    var floorSelect = document.getElementById("floorSelect");
    var selectedFloor = floorSelect.value;
    var mapImage = document.getElementById("floorMap");

    //Setting image source based on selected floor
    switch (selectedFloor) {
        case "lower":
            mapImage.src = "/img/printer/lowerlevel.jpg";
            break;
        case "first":
            mapImage.src = "/img/printer/first.jpg";
            break;
        case "second":
            mapImage.src = "/img/printer/second.jpg";
            break;
        case "third":
            noPrintersMessage.style.display = "block";

            mapImage.src = "/img/printer/third.jpg";
            break;
        case "fourth":
            mapImage.src = "/img/printer/fourth.jpg";
            break;
    }
}
```

Appendix O: Printer HTML

```
<body>
  <header>
    <h1 class="centered-heading">Printers Available</h1>
  </header>

  <main>
    <section id="contact">
      <h2><b><center>Select a floor to view the map  </center></b></h2>
      <br>
      <!-- Dropdown list-->
      <div class="form-group">
        <label for="floorSelect">Floor:</label>
        <select id="floorSelect" name="floorSelect" required>
          <option value="lower">Lower Level</option>
          <option value="first">First Level</option>
          <option value="second">Second Level</option>
          <option value="third">Third Level</option>
          <option value="fourth">Fourth Level</option>
        </select>
      </div>
      <button type="submit" onclick="displayMap()">Display Map</button>
    </section>
    <div id="mapContainer">
      <img src="" id="floorMap" alt="Floor Map">
      <p id="noPrintersMessage" style="display:none;">No printers available on this floor</p>
    </div>
  </main>

  <br>
  <br>

<canvas id="canvas" width="500" height="500"></canvas>
```

Appendix P: Printer page

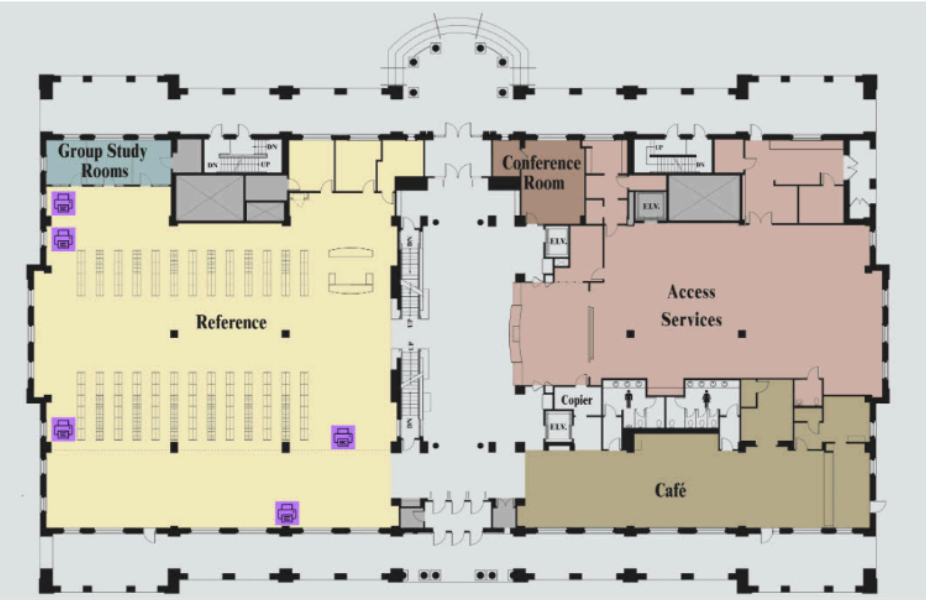
Printers Available

Select a floor to view the map

Floor:

First Level

Display Map



Appendix Q: Computer page