State University of New York at New Paltz

Team ID: 1

Noah Franklin (c-s21-08)

Yitzhak Alvarez (c-s21-03)

Alexandra Maceda (c-s21-14)

Project Type: On-Campus Project

“Smart Library”

FINAL PROJECT REPORT

LINK: cs.newpaltz.edu/p/s21-01/smart-library

LOGIN for Dynamic Map Admin Panel:

Username: a

Password: a

GITHUB: github.com/noah-franklin/smart-library

Computer Science Projects

Spring 2021

(Prof. Hanh Pham)

TABLE OF CONTENTS

1. Problem Description

1.1 Business Context and Goals

General Description:

The initial problem that began this project was that if a student or library worker wanted a book, they would need to look up the call number and go through the shelves manually and find the book. However, this can be an inconvenience because if the book is misplaced, it can be hard to find. Returning the book can also become an issue if the book isn't put in the correct location.

These inconveniences can be solved when creating a virtual library for SUNY New Paltz, which accurately reflects book locations without adding new tags. Users can also browse library floors (Main and Concourse), and bookshelves. The goal is to closely mimic a virtual experience while keeping the application simple and easy to use.

In previous semesters, students did not work much on the website but focused more on setting up and training CLARA and ANN so that it can then be used on the website. Things the previous group focused on were, expanding CLARA to compare accuracies of the book labels, set up ANN on the A server, created the ability to access and run images in the localizedChar directory. They began developing the dynamic mapping that would allow you to click on a shelf and take you to a bookcase. However, this would not work properly because of the data following different formats and being unsorted. Previous groups also worked on adding javascript popovers, creating a search system that works for all three floors. They wanted to create the script that will create HTML files for the shelves but again because of the data, this made it not work properly.

Coming into the semester and seeing what previous groups have done, we knew that one of the issues we needed to focus on was the reorganization of book images

in the A server. This is where most of the problems from previous semesters would come from. In order for the folders to be used and easily accessible via the website, we need to reorganize them into a consistent structure.

Additionally, we also needed to focus on creating the script that will create HTML files for when you click on a shelf. This was to be done after the reorganization because we wanted to avoid the problems previous groups went through. Lastly, the last thing we need to focus on was the design of the website. This would be done as we are working on the other tasks.

Users of the software: Sojourner Truth Library

When & Where can the users use this: Users can use this software via their preferred web browser. Web applications can be accessed via Desktop or Mobile.

Software Capabilities:

- Locate Book
- See if Book is Available
- Updating Current Book Locations

+ Analyze the obtained information and define the “BUSINESS” PROBLEM (in the context of the given business situations):

Sojourner Truth Library wants an easier way to find exact book locations, even if the book is not in its designated location.

Summarize things the software can do (transactions, operations, interactions … (be aware of Input, Processing, Output)

- User can login

  Input = User credential

  Output = Access Granted (if credentials match) or Access Denied (if credentials don’t match

- User can locate book

Input = Book Call Number

Output = Book Location (Floor>> Shelf>>Side>>Section>>Row>>Book)

- User can Browse Shelfs

Input = Click Floors/Shelves/Sides/Sections/Rows/Books

Output = Brings up next layer until arriving at books

1.2 Technical Requirements

ENVIRONMENT: Web App can run a desktop or mobile browser:

- CONNECTIONS: Centralized but connects to the A server => must have Internet access
- MEAN OF INTERACTION with user (keyboard, touch screen, …):

Mouse to click around the screen and explore the library

SYSTEM COMPONENTS:

- User Interface:
  - Basic computer-window display
- Processing:
  - Login mechanism:
    - Ø Input = Enter Credentials (User name & Password)
    - Ø Process = compare imputed credentials with the original in the DATABASE
    - Ø Output =

(i) fail when it's not matched => back to LOGIN

(ii) Access Granted when it's matched

- Browse Shelves:
  - Ø Input = Mouse input
  - Ø Process = Note what level user is at and also note the location of the click
  - Ø Output =
  - (i) Enter that next layer in the library
  - (ii) if It is a book layer, then info about the book will pop up

- Data:
  - Format of Data
  - Photos are saved as JPG images
  - Location of Data
  - "A" Server
- Hardware:
  - No additional hardware needed

1.3 Your Responsibilities

Restructuring Folders:

Before the start of the semester, each team member who took and uploaded the pictures to the A server did not follow a consistent folder structure. Our goal was to make sure that each bookshelf follows the same structure which is:

Shelf>> Side >> Section>> Row>> Book Images

By creating a consistent structure, it will be easier to design and implement the “Browsing Shelves” functionality in the website. We need to make sure that each image is in the correct folder so that we can use a python script to create static web pages with those images and link them to the website when the user clicks on it. After a few weeks and the use of a script, we managed to reorganize all folders and create a consistent structure among them all.

Updating Website:

This new website has a new login feature and the ability to click through images to browse floors and shelves. When you open up a shelf it takes you to an HTML file, the same thing when you want to open up a specific section. The website now visually displays the shelves and sections giving it that virtual library experience.

In addition to working on the website, a few group members and I need to sort through about 70k character images that would be used to train the neural network by other groups in the future.

2. Technologies

2.1 Related Technologies

Frontend/UI

- HTML - Used to structure the web pages.
- CSS - Used to style.
- Bootstrap - CSS framework for responsiveness and styling.
- JavaScript - Used to make the web pages interactive.
- jQuery - Much easier use of JavaScript on our website.

Backend

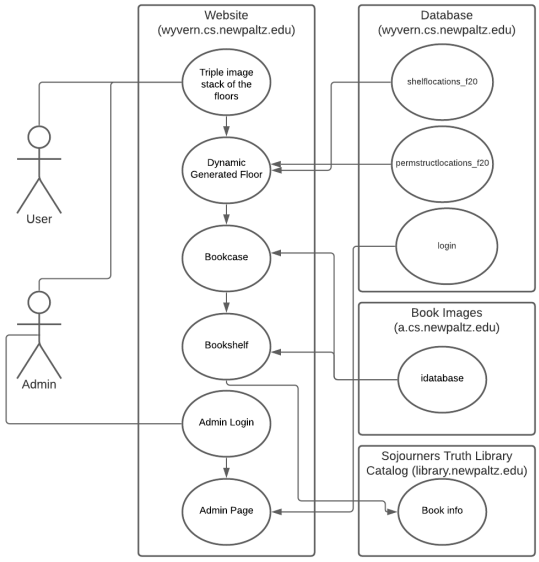
- PHP - Interact with the database to display on the front end.
- MySQL - Create tables and queries for the database.
- Python - Used to make shelfgenerator.py which goes thru every organized image and creates HTML files for bookcases & bookshelves.

2.2 Newly Learned Skills/Technologies

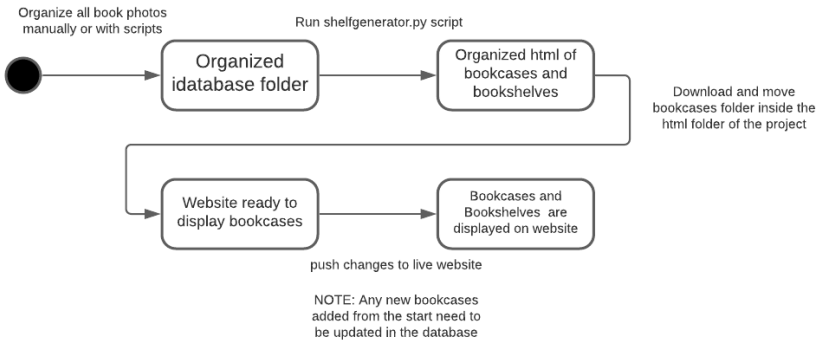
- Figma - Great design software tool for us to prototype our website.
- FileZilla - Great for file transferring and renaming folders.
- PHP - Good for backend development.
- jQuery - Simple but advanced JavaScript.
- Python - Great for making scripts.

3. Design

3.1 System Architecture



The website is hosted on the wyvern.cs.newpaltz.edu web server where the user will be interacting when connected on their browsers or mobile device. The user can navigate the site to browse bookcases located on a dynamically rendered map of the floor which is pulled from database tables also located on the wyvern.cs.newpaltz.edu server. When the user is viewing book images on a bookcase or bookshelf, the image links will point to image files hosted on the a.cs.newpaltz.edu server. In order to see these images, the user will need to be running the New Paltz VPN as the a.cs.newpaltz.edu server is not open to the public. Then to see more info about a book, the user can click on a book image which will bring them to a more info page on the library.newpaltz.edu website. Admins in addition to everything the user can do above can log in to the admin panel to edit the dynamic rendered map.



Since we are not using a web framework, we need to find an easy way to get all the HTML needed to display every bookcase and bookshelf in the idatabase folder where the images are stored and organized. That way when images are moved or updated, you can simply run the script on the image server, download and move the generated HTML to the website repository, and the bookcases and bookshelves will be automatically linked to the dynamic rendered map. However, if any new bookcases are added to the idatabase folder, the BookCase attribute in the shelflocations_f20 table of that bookcase will need to be added.

3.2 Components

File Organization	Bookcase/Bookshelf Generation

File Organization:

Our first task was to organize all the book images of the library taken from previous semesters. The folders and images were organized in different ways (if at all) because different students were responsible for different sections of the library. We started by using FileZilla and the VPN to remotely move and rename folders to follow a consistent naming format so we could easily display this on the website later. The following format was provided by the Professor...

Format:

vlib/idatabase/number-name/shelfnumber/side/sectionnumber/rownumber/imagefiles

Example:

vlib/idatabase/3-tahir/c46/a/section1/row1/IMG_0967

After manually organizing all the folders up to the side level, we decided to make some shell scripts to help with creating, making, and moving files and folders for the sections, rows, and images. The scripts we used can be found in the scripts directory of this repository.

Bookcase/Bookshelf Generation:

Now that we fully organized the idatabase folder, we now needed to make another script that will generate bookcase and bookshelf HTML files for the entire library in an organized fashion. We first started by designing what the bookcase and bookshelf page would look like which can be found inside the HTML directory of this repo under bookCase.html and bookShelf.html respectively. Then we used python to make a program that will loop through every folder/file in the idatabase folder with the use of recursion. Then by making the program create HTML files and write the appropriate HTML code as it loops through all the files and folders, we now have all the HTML we need for the library. This python script can be found in the scripts directory of this repository.

Clickable Books:

For future use when the CLARA/ANN system is more developed, the Professor wanted us to demonstrate a way for the image to have clickable sections which will link to the sojourner truth library page to show more info about the book. This search also has to be done with a book call number as that will be provided later on when CLARA processes an image. We started by creating a map of clickable columns in an image. These columns when clicked will run a function to open a new page on the sojourner's truth website to show more info about the book. The link is created by taking a book call number variable and adding it to the search URL. This book call number variable is currently hardcoded in but it can easily be changed to query a database to get all the

book call numbers for an image. All of this was added to the `shelfgenerator.py` script
Search URL:
https://suny-new.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma990002702100204844&context=L&vid=01SUNY_NEW:01SUNY_NEW&lang=en&search_scope=MyInstitution&adaptor=Local%20Search%20Engine&tab=LibraryCatalog&query=any,contains,<bookcallnumber>&offset=0

Where `<bookcallnumber>` is the variable containing the book call number

Image Stack Home Page:

The project proposal and Professor wanted the home page of the library to display an image stack of the three library floors. The floors consist of the Main, Concourse, and Ground floor. With some weird CSS, we managed to get it to look very similar to how it looked in the documentation while still being clean and responsive. Currently mousing over each floor will highlight it with the appropriate color and display the text of the floor name overhead.

f20-Dynamic-Mapping:

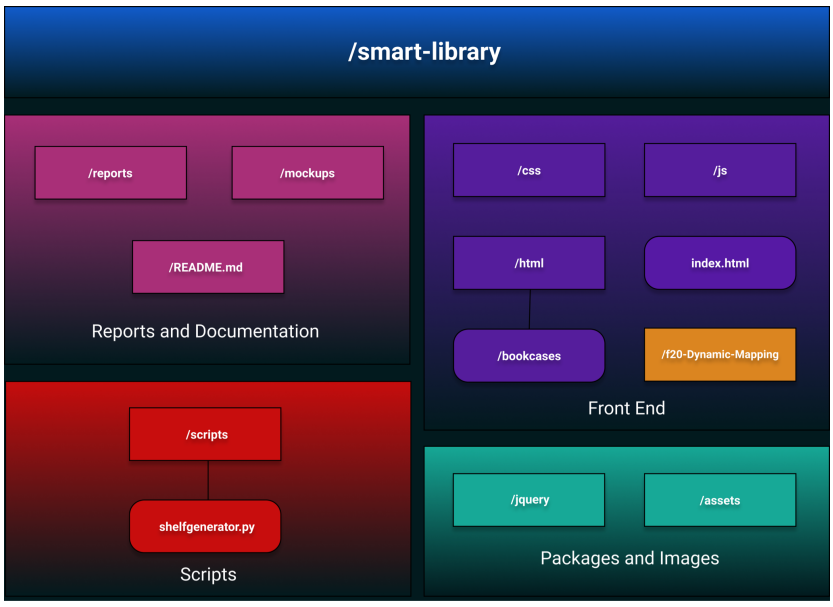
This part of the project was already completed by previous semesters and the documentation for it is inside the f20-Dynamic-Mapping folder. We took the part of the code responsible for the dynamic mapping to make our **(main, concourse, and ground.html)** files. We then had to connect this map to our generated bookcase HTML. We did this by copying the database from previous semesters and adding a new column called BookCase. We then manually added the bookcase number for each shelf that we have in the idatabase folder. Then we finished it by editing the PHP files that generate the black rectangular shelves located in f20-Dynamic-Mapping **(mainFloorDisplayer.php, concourseDisplayer.php, groundDisplayer.php)**. By removing the href and adding an onclick method to load the correct HTML file from the /bookcases folder.

Admin Functionality:

This is starred because it is not fully implemented yet. Currently, login info can be checked but upon successful login, the admin page **(/f20-Dynamic-Mapping/Admin/adminPanel.html)** will not open because browsers by default block pop-ups not generated by user action or event. Since the admin page load is triggered inside the PHP **(/f20-Dynamic-Mapping/Admin/login.php)** to check for the right login, this is considered not a user action and the page is blocked. This can be worked around if you allow all pop-ups from the website, but a better solution would probably be to add a login session and have an admin button appear after you log in. This way the admin button will be a user action and the admin page should display in a new tab. Another issue with the admin part is that if you have the direct link, any user could view the page. A solution for this would be to implement a route guard for the admin page.

4. Software/ System Description

4.1 Map of Files



Front End

- /CSS – CSS files for styling the website
- /js – javascript files to make the website dynamic
- /HTML – HTML files to structure the website and is inserted into the website with jquery when we need it
- /bookcases – organized HTML files that display the bookcases and bookshelves for the library, this is produced by the shelfgenerator.py script
- index.html – the main HTML file where the website is brought together, this is loaded by default when viewing the website
- /f20-Dynamic-Mapping – project from previous semesters to dynamically display shelves

Packages and Images

- /jquery – jquery package to let use easy and powerful javascript
- /assets - images files used throughout the website

4.2 DATA Format & Description

We imported the database used from previous semesters in order to take advantage of the dynamic rendered map and login functionality. The only thing we added was the BookCase column to the shelflocations_f20 so we could link our HTML bookcases to the map. Our database login info as well as an exported SQL file of the entire database will be available on the web server.

4.3 Developer's Guide

Setting up a dev environment with the project is really simple since we didn't use a framework. As long as you have the project by cloning with git or getting it elsewhere, you can simply start the website with a local web server using the VSCode extension Live Server or XAMPP.

Prerequisites

Local

- Git (to clone this repository and push changes to a web server)
- Live Share (VSCode Extension to run a local webserver to view the website)

Live

- All Local Prerequisites
- Cisco AnyConnect Secure Mobility Client (VPN needed to view the images on the live website and to connect to the "a" server)

Local

1. Install the prerequisites
2. Clone this repository or get it from the professor
3. Right-click index.html and "Open with Live Server"

Live (wyvern)

NOTE: Make sure a file called "connect.php" is located inside f20-Dynamic-Mapping/Admin folder with the following code and replace `<DATABASE USERNAME>`, `<DATABASE PASSWORD>`, and `<DATABASE NAME>` with the correct info. This file is responsible for establishing a connection to the database for the dynamic rendered map and login features. I was not able to get this to work in a local environment as I'm not sure how to access the database on the username and password-protected wyvern server. The web server hosting this website should have connect.php with our database info.

Instructions for `shelfgenerator.py`

1. Use FileZilla to create a bookcases directory inside the idatabase folder (if this is already done make sure to delete everything inside the bookcases directory before running the script)
2. Move the `shelfgenerator.py` script into the idatabase folder
3. SSH into the "a" server to run the script file (./shelfgenerator.py)

4. Wait for the script to finish and all the generated HTML should be located in the bookcases folder

Output

The script output format is as follows:

/bookcases/shelfnumber/side/sectionnumber/rownumber.html

The bookcase HTML is located inside each shelfnumber under a.html and b.html for side a and side b respectively of a bookcase.

The bookshelf HTML is located inside the a and b folders of the shelfnumber and organized per section and row.

Further explanation with an example:

- inside bookcases, the c01 folder is the bookcase
- inside the c01 folder is a.html and b.html which is the “a” side and “b” side of the c01 bookcase (This is what the bookcase page will look like)
- inside the c01 folder is a and b folders containing the sections of the bookcase for the respective side
- inside the section folders are row HTML files that display the appropriate row of the bookcase (This is what the bookshelf page will look like)

From there they can just move the bookcases folder inside the HTML folder of the website repository. If any new bookcases are added, you will need to go into the database and update the appropriate shelf with the correct bookcase number. You can do this as follows...

1. Find the ID of the shelf by right-clicking and inspecting element on the correct rectangular shelf on the map and finding the id number in the <rect> HTML

Example:

<rect class="1-55 1-56" id="255" width="7.5" height="87.7" x="442.1" y="220" style="cursor:pointer;"></rect>

Where 255 is the id of this shelf

2. Log in to the database using PHPMyAdmin on the New Paltz CS web server (cs.newpaltz.edu/phpmyadmin)

NOTE: Login info, as well as a database export SQL file, will be available on the wyvern web server where the website is hosted (/var/www/projects/s21-01)

3. Open the p_s21_db database and go into the shelflocations_f20 table

4. increase the number of rows you can see to max then search the id number from step 1

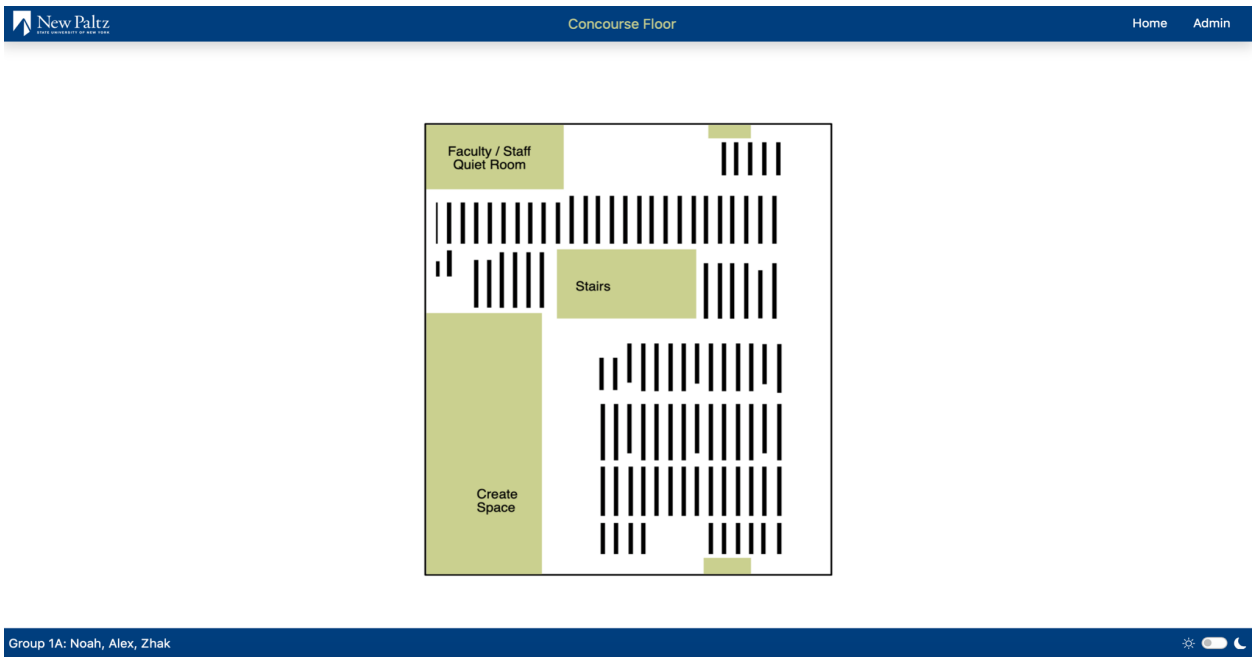
5. Edit the table entry containing the id of the bookshelf and add the correct bookshelf number to the BookCase attribute that should be blank. In this case, 255 corresponds to bookshelf c12

4.4 User's Guide

Our website starts from the index.html page where four `<div>` tags are located. These `<div>` tags are populated using jquery **(index.js)** to display the right HTML files. So initially the navbar **(navbar.html)**, homepage of the image stack **(home.html)**, and footer **(footer.html)** are loaded. Clicking the "Home" text in the navbar will bring you back to this initial state and clicking the "Admin" text will display a modal **(loginModal.html)** to login and if successful open the admin page.

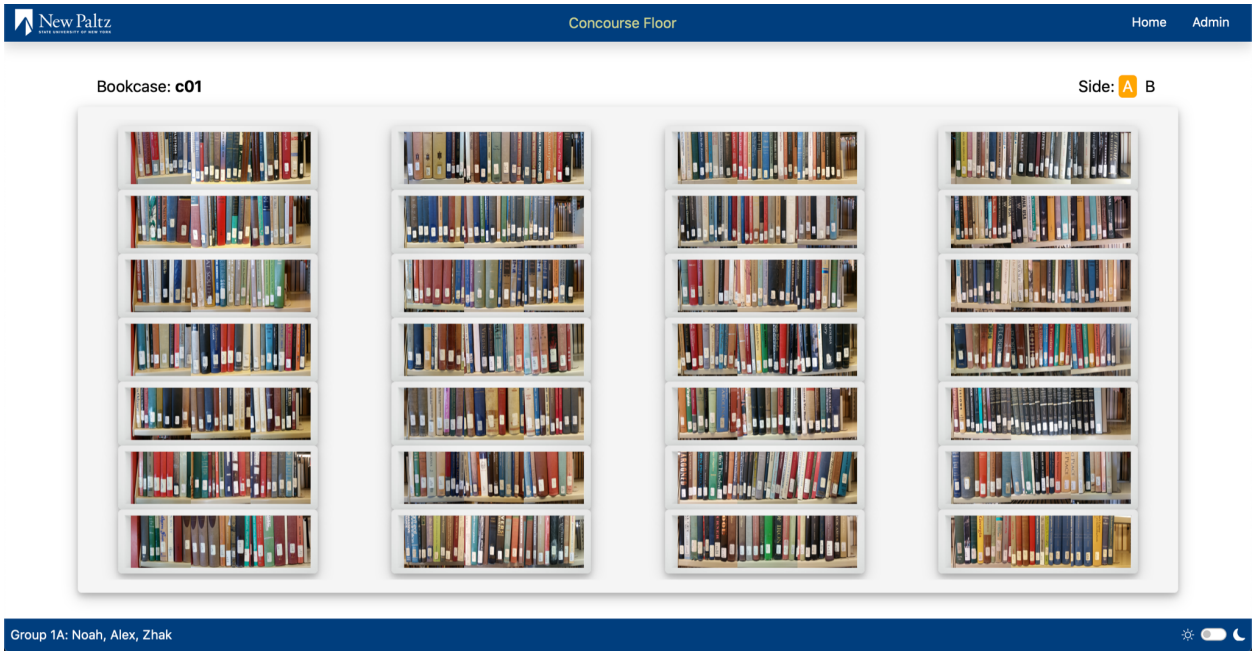
home page

Clicking any of the floors will replace the homepage image stack with the correct dynamically rendered floor **(main,concourse,ground.html)**.



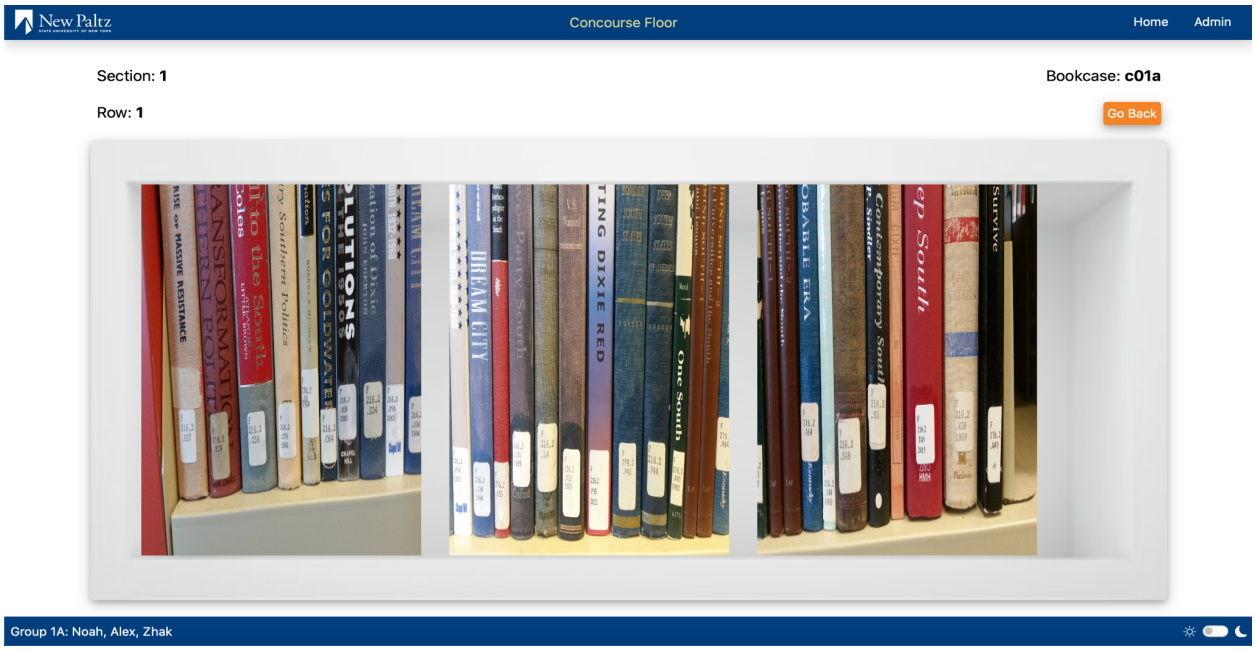
dynamic map

From there you can click any bookcase (black rectangular shapes) and if there is a bookcase page for it, the dynamically rendered floor will be replaced by the bookcase page (For Example: /bookcases/c01/a.html is loaded when you click on the c01 bookcase). The user will then see the bookcase page populated with all the images of the bookshelf (this can take some time) and switch the bookcase side by click on A for side a or B for side B.



bookcase page

They can then click on a bookshelf to replace the bookcase page with the correct bookshelf page (For Example: /bookcases/c01/a/section1/row1.html is loaded when you click on the first bookshelf (top left) on the c01 bookcase).



bookshelf page

To finish it, you can then click a column on the book image to open a new page displaying more info about the book. The Bookcase and Bookshelf pages will display info about the bookcase number, section, and row accordingly and buttons to go back or the home button can be used to navigate.

5. Test Results/ Observations

5.1 Test Function 1/Transaction #1 (Performance)

From a performance standpoint, the website functions really well. This is mainly because the website only uses the supplied jquery package and Bootstrap, everything else is just standard HTML, CSS, and javascript. The only part of the website that lacks performance is the bookcase page. Loading 50+ high res images from the “a” server cause the website to lag until it finishes loading the images. The only reasonable fix for this would be to make a smaller copy of all the book images so the website could load them faster.

5.2 Test Function 2/Transaction #2 (Quality)

This website has really come a long way since the midterm. The website design has seen many changes after implementing the mockup design as we thought it wasn't that visually appealing. Now the website has a very clean and simple look and it is for the most part mobile responsive. The triple image stack of the floors came out great and the bookcase/bookshelves pages look pretty similar to how they would in person. All the information the user might need is easily accessible. Admin functionality is there although not complete as well as the dark mode.

6. Professional and Career Benefits

The challenges faced in this project illustrate the current computer science job market, which requires strong proficiency in a variety of technology and languages to succeed. Web Development skills such as database management and server management are essential to maintain projects moving along without becoming slowed down by unexpected problems. The opportunity to collaborate and expand from someone else's program rather than starting from scratch is the most significant advantage gained from this initiative. It is a must to learn about the software through different documentations. It is an obligation to maintain the documents up to date so that any creator may pick up where you left off without sifting through information. This training will act as a springboard for my professional career, whether in Web Development or some branch of computer science, such as database management. As a humbling experience, I will take the lessons learned from this project in my career.

7. Conclusions

Overall this project has come a long way, most of the issues that came about during previous semesters were due to the fact that the images in the A server were not organized properly. However, after working on that this semester we believe that it will be much easier for future groups to be able to handle that large amount of data and use it to push the project forward. Additionally, fixing the website allowed us to get a step closer to achieving the goal of the project. After sorting and creating a new data set for the neural network, we believe this will allow CLARA and ANN to work more efficiently and make it easier to look up books and it's info on the library website.

8. References

[1.]J. T. Mark Otto, “Popovers,” · Bootstrap. [Online]. Available: https://getbootstrap.com/docs/4.0/components/popovers/. [Accessed: 2021].

[2.]JavaScript Tutorial. [Online]. Available: https://www.w3schools.com/js/default.asp. [Accessed: 2021].

[3.]jQuery Tutorial. [Online]. Available: https://www.w3schools.com/jquery/default.asp. [Accessed: 2021].

[4.]“Learn Bootstrap,” Bootstrap W3 Schools. [Online]. Available: https://www.w3schools.com/bootstrap/bootstrap_ver.asp. [Accessed: 2020].

[5.]PHP Tutorial. [Online]. Available: https://www.w3schools.com/php/default.asp. [Accessed: 2021].

[6.]SQL Tutorial. [Online]. Available: https://www.w3schools.com/sql/default.asp. [Accessed: 2021].

[7.]Python Tutorial. [Online]. Available: https://www.w3schools.com/python/default.asp. [Accessed: 2021].