Thriftara

Clothing Thrifting Websites similar to eBay and Craigslist

CISC 4900 Spring 2023

Group Members:

William Lin (VC1C) Email: William.lin48@bcmail.cuny.edu

Michael Newman (VC1A) Email: Michaelnewman690@gmail.com

Meggie Cheng (VC1A) Email: Meggie.cheng46@bcmail.cuny.edu

Chaocheng Zhu（VC1B) Email: Chaochengzhu@gmail.com

Technical Supervisor

Supervisor Responsibilities: Providing occasional feedback(monthly) with evaluation on group performance given to instructors on 4/7 and 5/10.

Basak Taylan

Professor, Brooklyn College

Email: Basak.taylan@brooklyn.cuny.edu

Table Of Contents

Abstract

This project focuses on creating a website for users to sell and buy used and no longer wanted items, similar to websites like eBay, Craigslist, Grailed, etc. Items can range from clothes to technology that is in good condition and reusable. But, unlike existing thrifting websites, we want to create a site centered around the user experience. This includes a better UI and item recommendations based on user preferences.

Changelog

1/27/23 - Project Idea Creation: Decided on the project to work on including elements and programs needed. We also worked on a timeline/deadline to follow for the semester and made edits to the project report.

1/30/23 - Completed Project Proposal.

2/3/23 - Create a layout for the website, this includes all web pages and elements on each page. Group also started working on wireframes.

2/11/23 - Completed wireframes and made a prototype and database for the website. We also decided on the applications we'll use for the website.

2/13/23 - Group worked on figuring out the React library and Spring Boot framework.

2/12/23 - The group started implementing the webpages and created the basic functionality in the backend.

3/20/23 - Most webpages have been created and the group has started to work on linking the back end with the front end.

4/19/23 - The group is the process of adding more features to every page.

5/12/23 - Complete most necessary elements on the webpage.

Project Details

Why - Instead of throwing away perfectly good items, users can sell the items to other individuals who are looking for those exact items to make a profit. By facilitating trade between users, we can also contribute to limiting waste and saving the environment.

What - Once consumers make an account, they are able to navigate the site, shop, and upload items for sale. This is a similar concept to eBay.

Who - Our target demographic are those looking to purchase second-hand items for a lower price.

Where - Our Project will be hosted online, where users around the world would have access to it.

How - Our website will use a format similar to every other shopping site to retain muscle memory for simple navigation. Where users will be able to add items they'd like to purchase to their cart and checkout the items.

When - The project is done when all the features, including the sign-in page, login page, payment page, upload page, etc, are up and running. Then, it'll be hosted online for users to access.

Technical overview of the project

We will create a website by using the React library for the front end, with the Spring Boot framework for the backend and MySQL for the database. Will connect the front end and back end with CRUD operations through Axios. If it's within the time constraint, we'll try to post the website online as well. To improve the UI of our website, we're going to follow Schneiderman's Eight Golden Rules while also building upon ideas from preexisting web pages. To create better item recommendations for users, we're hoping to create a personalized homepage depending on their interests when signing up for the Thriftara.

Experience

William - has experience with Html, CSS, Javascript, and Java. With limited experience with Bootstrap, and PostgreSQL.

Meggie - has experience using HTML, CSS, react, bootstrap, ExpressJS, Firebase Firestore, and creating REST APIs.

Michael - has experience with Html/CSS, Javascript, PostgreSQL, and limited experience with Springboot and creating REST APIs

Chaocheng - has experience with Html, CSS, javascript, java, Mysql, Maven, Mybatis, and Spring.

Project Objectives and Expected Outcomes

Objectives:

Allow the user to create a profile/account

Allow the user to upload items for sale

Allow the user to search for items(Search Bar and Search Page)

Allow the user to add and remove items in the cart

Allow users to purchase items from the cart

Create a front page for the user to view

Allow users to follow social media of the website

Allow the users to choose a delivery method

Create a logo for the website

Expected Outcomes:

Get users to be impressed by websites front page

Get users to sell items with a percentage going to website

Get users to buy items

Get users to follow websites social media

The users can easily navigate the website

The user can easily find the item they're looking for.

Software and Hardware Technology Requirements

Software Requirements

- Visual Studio/React/Spring Boot (Creating website)
- Github (Code Repository)
- MySQL (Database)
- Mybatis (JDBC framework)
- Chrome
- Google Drive (Docs)
- Marvel Prototype (Wireframes and prototyping)
- Axios to connect the front end and back end.
- Postman (Backend Testing)
- Intellij(Spring Boot Framework)

Hardware Requirements:

- Computer
- Internet

Software Construction

We'll upload changes made during the coding process onto GitHub to keep track of commits made by different group members. For the project, we'll also follow the iterative design model, with around 3 cycles of improvement.

Milestones and Reporting

Task	Progress
Project Report	Done
Wireframe	Done
Database Design	Done
Prototype	Done
Barebones Website	Done
Connect Database to website	Done

Webpage Checklist

Date	Group Member	Webpage	Completion
3/8/2023	William	Sell page	Completed
2/17/2023	Meggie	Homepage, Navbar, and footer	Completed
2/17/2023	Michael	Search page	Completed
2/27/2023	William	Login and Signup page	Completed
2/10/2023	William	About and Contact page	Completed
3/7/2023	Meggie	Shopping Cart page	Completed
3/3/2023	Meggie	Brands page and FAQ page	Completed
4/20/2023	Meggie	Product details page	Completed
5/8/2023	Meggie	Checkout - Address feature	Completed

Project Management

The project will be managed by a check-in/works distribution system where a task will be assigned to a given group member with a deadline as to when it has to be completed. For the project, we also decided on checking in with our supervisors once every month to update them on the changes we've made and with further guidance from evaluations on 4/7 and 5/10.

Estimated Timeline and Tasks

Dates	Tasks to Complete
1/25-1/31	Gain a firm understanding of what the project will be about.
1/31-2/14	Create a wireframe, storyboard, database, and general architecture of the website. Update the supervisor on our progress and for any guidance.
2/14-2/28	Create a prototype of the website. Update the supervisor on our progress and for any guidance.
2/28-3/24	Learn more about the programs/software we'll use for the project and create a barebones website. Update the supervisor on our progress and for any guidance.
4/7	Ask for an evaluation from the supervisor.
3/24-4/18	Work on implementing the back end of the website.
5/10	Ask for an evaluation from the supervisor.
4/18-5/16	Make improvements to the website based on feedback from the supervisor and any quality-of-life changes we think of.

Group Member Responsibilities

Group Members	Role	Description
All	Other	Attend All Group Meetings
All	Documentation	All members are responsible for version control
All	Documentation	Complete Project Proposal
All	Documentation	Create webpage layout and create a wireframe, prototype, and database.
All	Documentation	Completed all wireframes and finished the prototype. Create the database.
William, Meggie, Michael	Research	Learn how to use the React library.
William, Meggie, Michael	Programming	Create webpages with filler data that will be linked to the back end when back-end functions are complete.

Chao Chen	Programming	Create back-end functions and access the database.

Actual Timeline

Date	Group Members	Duration	Role	Description
1/25/23	All	1 Hour	Other	Reviewed the syllabus and got an overview of how the course will be conducted.
1/26/23	All	2 Hours	Other	Project Idea Research
1/26/23	William	2 Hours	Documentation	Created a plan for what needs to be discussed during the meeting. I also created a group chat, docs, and GitHub.
1/27/23	All	6 Hours	Documentation	Discuss what the project will be, while also working on the project report and project logs.
1/27/23	William	2 Hour	Documentation	Cleaned up project report and sent emails to possible supervisors.
1/29/23	William	3 Hours	Documentation	Revised project report, by making goals clearer and improving grammar.
1/30/23	All	4 Hours	Documentation	Completed Project Proposal
2/3/23	All	4 Hours	Documentation	Created webpage layout, figured out an application to use for wireframes, and started working on wireframes.
2/7/23	William	3 Hours	Documentation	Completed homepage wireframe, fixed up other wireframes, and prepared for what needed to be completed for 2/10.
2/10/23	All	30 minutes	Other	Met up and checked in with supervisor
2/10/23	All	4 Hours	Documentation	Completed all wireframes, finished the prototype, and create the database. We also decided on which applications to use for the website.
2/14/23	William	4 Hours	Research and Creation	Figure out how MySQL worked, created a database for the project, and fixed up the entity relationship diagram.

2/15/23	William	2 Hours	Research	Research on full-stack development and deployment to figure out how to delegate work.
2/17/23	All	2 Hours	Documentation	Discussed the programs were gonna use, and how it's going to be implemented. Also, we delegated work to each group member to work on.
2/17/23	William	30 Mins	Documentation	Created some logos
2/20/23-2/22/23	William	8 Hours	Research	Learning React
2/22/23	William	5 Hours	Implementation and Error Handling	Started working on the About page and worked on linking it together.
2/24/23	All	2 Hours	Other	Group meetings on progress, issues, and discussed further steps.
2/24/23	William	30 Mins	Documentation	Figured out how to version control and upload the logo with transparency.
2/27/23	William	3 Hours	Implementation	Finished the About page, struggled a bit with linking and image sizing so it took a bit.
2/27/23	William	1 Hours	Implementation	Learned and implemented CSS module to give every component a unique style.
2/27/23	William	2 Hours	Implementation	Completed contacts page.
3/1/23	William	5 Hours	Implementation	Created Sign-in page, took a while cause I wanted an animation for clicking on enter email, but it required jquery import which can mess up the way react formats the DOM, so I kept it simple.
3/1/23	William	1 Hour 30 Min	Implementation	Tried to figure out how to combine bootstrap CSS with react modules, but it didn't work. Found custom bootstrap classes.
3/3/23	All	2 Hours	Documentation	Meet with supervisor to describe the progress we've made and the steps we'll take going forward. Also, shared our web pages and ideas with each other.

3/6/23	William	6 Hours	Implementation	Created the signup page and is trying to figure out how to validate and change DOM elements using react refs. Also did some research on arrow functions and states.
3/7/23	William	1 Hour	Implementation	Trying to figure out how to stop the form from wiping when the password doesn't match.
3/8/23	William	1 Hour 30 Min	Implementation	Worked on the Sell page and create the option to change between different states. I just need to create the individual components now and display the required component base on the button clicked.
3/8/23	William	2 Hour	Research	Learning more about react hooks and more specifically how to use "state" and how to create elements. Link to create: https://stackoverflow.com/questions/72285009/how-to-remove-object-from-array-based-on-id-in-react Link to delete: https://www.youtube.com/watch?v=jWWW9Wyl0mY&t=21s&ab_channel=Devtamin
3/8/23	William	1 Hours	Documentation	Described each member's progress and the steps forward, like the functions we still need. Going to start working on combining the front end with the back end next week.
3/14/23	William	2 Hours	Research	Trying to figure out how to use Axios to connect react with spring boot.
3/15/23	William	2 Hours	Merge Repositories	Fixed some merge conflicts that require further input from team members to fully merge everything because of the use of Bootstrap and class names.
3/17/23	All	2 Hours 30 Mins	Documentation and implementation	Figured out how to export and import Springboot and MySQL to another system. Also, discuss and resolve any problems we faced.

3/20/23	William	2 Hours	Implementation	Assisted group members with various things like importing database and tried to figure out how to use Postman.
3/20/23	William	4 Hours Mins	Research	Reading backend code and trying to figure out how to connect them. We looked into Ajax implementation and Axios to see which would be better for our situation. Lending toward Axios as a jquery import can mess up the DOM and they don't seem too different. Just need to figure out how the URL and JSON file works in Ajax. Mainly confused about what the URL to access the data would be as most tutorials use CORS but we didn't and I'm not sure what the address is without it.
3/22/23	William	2 Hours	Implementation	Created the forgot password page and was trying to figure out why the column size wasn’t changing correctly, the card tag was in the wrong location.
3/22/23	William	4 Hours 30 Mins	Implementation	I Was trying to figure out how to render different components conditionally because if-else statements don’t work in react and I completed the layout for the Sell page. setState() solved this issue. Completed the basic layout of the SellNewItem Component. Combing Bootstrap with CSS modules: https://stackoverflow.com/questions/43917636/how-to-use-css-modules-and-bootstrap-at-same-time
3/24/23	All	2 Hours	Research and Implementation	Figured out how to connect frontend with backend with cross-origin and Axios.
3/26/23	William	1 Hour 30 Mins	Documentation	Cleaned up project report.
3/29/23	William	4 Hours	Debugging and Research	I Got stuck cause I thought it had worked before since data was posted inside the database, but it was from Postman. Figure out it was a security issue with cross-origin and tried to figure it out.

3/30/23	William	1 Hour	Research	Still trying to resolve the conflict.
3/31/23	All	1 Hour	Other	Meet with supervisor and updated her on our progress.
3/31/23	All	4 Hours	Debugging and Research	Figure out the get request works but Post doesn't, tried extension and global configuration.
4/2/23	William	2 Hours	Research	Did some more research, going to test out some PHP code, more Chrome extensions, or perhaps a different web browser.
4/3/23	William, Chaocheng	1 Hour	Implementation	Worked on installing react on Chaocheng's laptop, so he can assist with resolving the error.
4/4/23	William, Michael	1 Hour	Debugging	Figured out why React dropdowns weren't working.
4/6/23	William	4 Hours	Debugging	Watch Spring-boot videos and kept reading through forms and trying different things until I finally figured it out. As the filter has to be individually enabled for every type of CRUD operation. Currently trying to figure out why the post request from Axios isn't sending the correct information.
4/7/23	All	1 Hour	Other	Talked about our current progress, so issues that came up and potential solutions, and the features we still need from the backend.
4/7/23	William	3 Hours	Debugging and Research	Tried multiple methods to solve the post-issue, researched Springboot and Ajax to find a solution. The solution was to change the datatype being sent, which I tried before but later realized I wrote the incorrect URL last time.
4/12/23	William	4 Hours	Implementation	Connected the signup page and login page to the backend and tried to figure out how to get the incorrect login message from the backend. As we have successful logins with no entries that caused constant success logins. Also,

				suggested online solutions used response.Status which returns if Post went through and always returned success, instead response.Data.State(this needed to be used to find if the login was correct, it took a bit to figure out.
4/13/23	William	6 Hours	Research and Implementation and Debugging	Looking at the backend code to try and figure out how to connect the SellNewItem page. Figured out that getElementById works but only for getting, it can't be used to alter the data, and references need to be used for that. Got stuck on data submission and figured out the name call was incorrect which took a while. Trying to figure out "current request is not a multipart request" now. Figured it out, just had to change content-type input and past the data directly without any deconstruction.
4/14/23	All	1-Hour	Other	Group meeting to talk about everyone's progress.
4/14/23	William	3 Hours	Debugging and Uploading	Windows update messed up my JDK and had to reinstall it to run Springboot. Uploaded changes to main. Note: Cross-origin also randomly started working for some reason even without the Corsconfig (maybe Chrome updated?).
4/17/23	William	30 Mins	Debugging	Cross-origin stopped working again, had to add Corsconfig to fix it. I'll just keep it in there just in case.
4/17/23	William	2 Hours	Debugging and implementation	Working on the for sale page, but realized I can't fetch data from the backend and was looking through code to try and figure it out. Tested a view outcomes as well.
4/17/23	William, Chaocheng	2 Hours	Debugging	Trying to figure out the issue, and updated the MySQL database just to make sure. I figured out it has something to do with how HTTP sessions are different when accessed from the front end through cross-origin.

4/18/23	William	1 Hour	Research	I believe the HTTP session id changes every time a CORS request is sent. Thus something like spring security is needed to authenticate and record the session of the user.
4/19/23	William	5 Hours	Research	Just to make sure, I tested the code and found that the session id was changing from every Cors call. Watched a few tutorials and looked into Axios Authentication which involves using cookies. I also tried to see if I can fetch the CID to the front to use as a token, which will probably work but isn't a good practice(probably do this based on time constraints). Good Tutorial for tokens and spring security if we decide to use this: https://auth0.com/blog/spring-boot-authorization-tutorial-secure-an-api-java/
4/19/23	William	2 Hours	Implementation	Created the general formats for the ForSale and Sold pages with temporary data that will replace once we figure out how the backend authentication works. There are tutorials to do this with function components, but I wanted to get it to work with class components which took a bit of testing.
4/21/23	All	1 Hour	Other	Meet with supervisor and as a group to talk about our current progress and what needs to be completed.
4/21/23	William, Chaocheng	3 Hours	Debugging	Worked on fixing the login system, and got it to work with credentials.
4/26/23	William	2 Hours	Implementation and Debugging	Created the purchases page and was trying to figure out how to fetch data from Axios call in-class component. It wasn’t working because regular functions don’t have this component which makes this.State impossible. The solution was to change the then and catch functions to arrow functions. I also messed around with

				componentDidMount, but I can't use it because of the implementation of Axios in app.js, refreshing the page would cause my get request to activate first which ends up with the wrong data.
4/26/23	William	4 Hours	Implementation, Debugging, Research	Turns out componentDidMount does work, it just displays the other data after and I assumed it wouldn't. Created the basic template, but after looking at the backend function. Parameters need to be changed since we aren't using jquery with React. Added backend information to the Forsale page. Also, try to figure out how images will work, need to ask the backend later to see if I can move the image location to the frontend of make a link to a URL online. Was told I need parameters are the Cid and was trying to figure out how to send it back.
4/28/23	All	1 Hour	Other	Meet as a group to discuss any issues we might be facing and resolved the problems.
5/3/23	William	6 Hours	Implementation, Debugging, Research	I wanted to list all items purchased on the purchase page, but because I was using a class component, I didn't want to change it to a function component and was trying to figure out how to display the data that way. I featured that the information would have to be stored in a JSON to continually render it and did so. But had trouble accessing the data afterward. I lefted it for now and complete the image import for the for sale page. I try a bit more, I'll implement it with function components and react hooks if I really can't get it to work.
5/4/23	William	2 Hours	Implementation	I figured out why I had trouble accessing it because React doesn't have access to refs in the render function. I'll also go back to using react states since they have better access and I figured out how to create the data in them. Set state seems to only be accessible from the render

				function since it's called asynchronously, so I'll declare it normally in the function for now and pass it onto the other function.
5/4/23	William	4 Hours	Debugging	Created the JSON and had I hard time actually accessing the file. So I recreated it with several methods, but got stuck again on accessing, and solved it by accessing with mapper. I need to still figure out when refreshing the page messes up my order.
5/5/23	William	3 Hours	Research and Implementation	Played around with the code and changed async off on Axios request for one type of Axios request. This fixes the data not being called in time. But this should be don't with large data sets since it can mess up UI and freeze the screen as well. This is a temporary solution, I'll try to handle the promise by returning it when I have time. But this does involve referencing class names for <td> and I'm not sure if it'd work for my situation. Still need to figure out why the order is messed up on refresh, this might be fixed if I try async and wait, but I have an if function to check for now.
5/5/23	William	1 Hour	Implementation	Added all the information and completed the purchase page. I have to fix up the images, but I wanna ask the backend before I do.
5/5/23	All	1 Hour	Other	Talked about elements we still need to figure out and the solutions and what needs to be completed before next week.
5/5/23	William	30 Mins	Implementation	Fixed up the images on the purchase page.
5/7/23	William	1 Hour 30 Min	Implementation and Debugging	Started working on Sold page, but ran into some issues with the home page not loading. Talked to Maggie and figured out it was .toFixed(2) which was used for rounding that was causing a problem on my end. But, it works for her so it's kinda weird.

5/8/23	William	1 Hour 30 Mins	Other	Looked for a recording software to use and informed group members about it. I also wrote the technical implementation for the pages I worked on.
5/8/23	William	3 Hours 30 Mins	Implementation, debugging, and Other	Complete the sold page which had a weird bug where I could call a function that worked for the backend. After I lefted it for a while and came back, it randomly started to work. Also, worked on the slides for the presentation and listed what each group member will present.
5/11/23	William	4 Hours	Other	Finish the slides, create the videos, helped group members, and clean up the project report.
5/12/23	All	30 Min	Other	Last meetings with supervisor and group to discuss final changes and submission.
5/12/23	William	1 Hour	Other	Reorganized project report and merge branches.
5/13/23	William	2 Hours	Other	Helped group members with various tasks like debugging and video recording.
5/14/23	William	3 Hours	Other	Piece together all videos and uploaded it to youtube. I also made final changes to the project report.

Webpage Layout

Note: Numbers represent the order of importance. Navigation Bar and Footer will appear on all pages.

Home page:

- Navigation Bar
    - Search Bar
    - Login/Signup(replace with username after they log in)
    - Logo
    - Start selling
    - Cart button
    - Columns
        - Brands
            - Nike
            - Supreme
            - Prada
            - Gucci
        - Menswear
            - Tops
            - Bottoms
            - Outerwear
            - Footwear
            - Accessories
        - Womenswear
            - Tops
            - Bottoms
            - Outerwear
            - Footwear
            - Accessories
        - About

- Body
    - Slideshow
    - Brand Selection
    - Menswear Selection
    - Womenswear Selection
    - Recommended Items
    - Extra: What the user recently looked at.
    - Extra: List items under $50, $100, and $200.

- Footer
    - Logo
    - Social Links
    - Contact information for customer support
    - FAQ
    - Subscribe to the newsletter

Login page:

- Email/Username
- Password
- Signup
- Forgot Password

Signup page:

- Username
- Email
- Password
- Password Confirmation

Forgot Password Page:

- Email
- Username

About:

- History about the company
- Founders
- Slogan
- Purpose

Contacts(customer support):

- Emails
- Phone numbers
- Operation hours

Search Page:

- Side Bar
    - Department
        - Menswear
            - Tops
            - Bottoms
            - Outerwear
            - Footwear
            - Accessories
        - Womenswear
            - Tops
            - Bottoms
            - Outerwear
            - Footwear
            - Accessories
    - Price Range
        - Min Price
        - Max Price
    - Size

        - S
        - M
        - L
        - XL
    - Brand
        - Nike
        - Supreme
        - Prada
        - Gucci
    - Condition
        - New
        - Like New
        - Very good
        - Good
        - Fair
    - Shipping Option
        - Local
        - Delivery
    - Sort by
- Listings
    - Row for the number of listings
- Back and forward arrows to navigate between listing

Sell page(4):
- Upload items
- Set price
- Sales profile
    - Selling
    - Sold

Shopping Cart page(5):
- Remove items
- Add to Cart from different pages
- Items in cart
- Checkout button
- Total Cost
- Price of each item

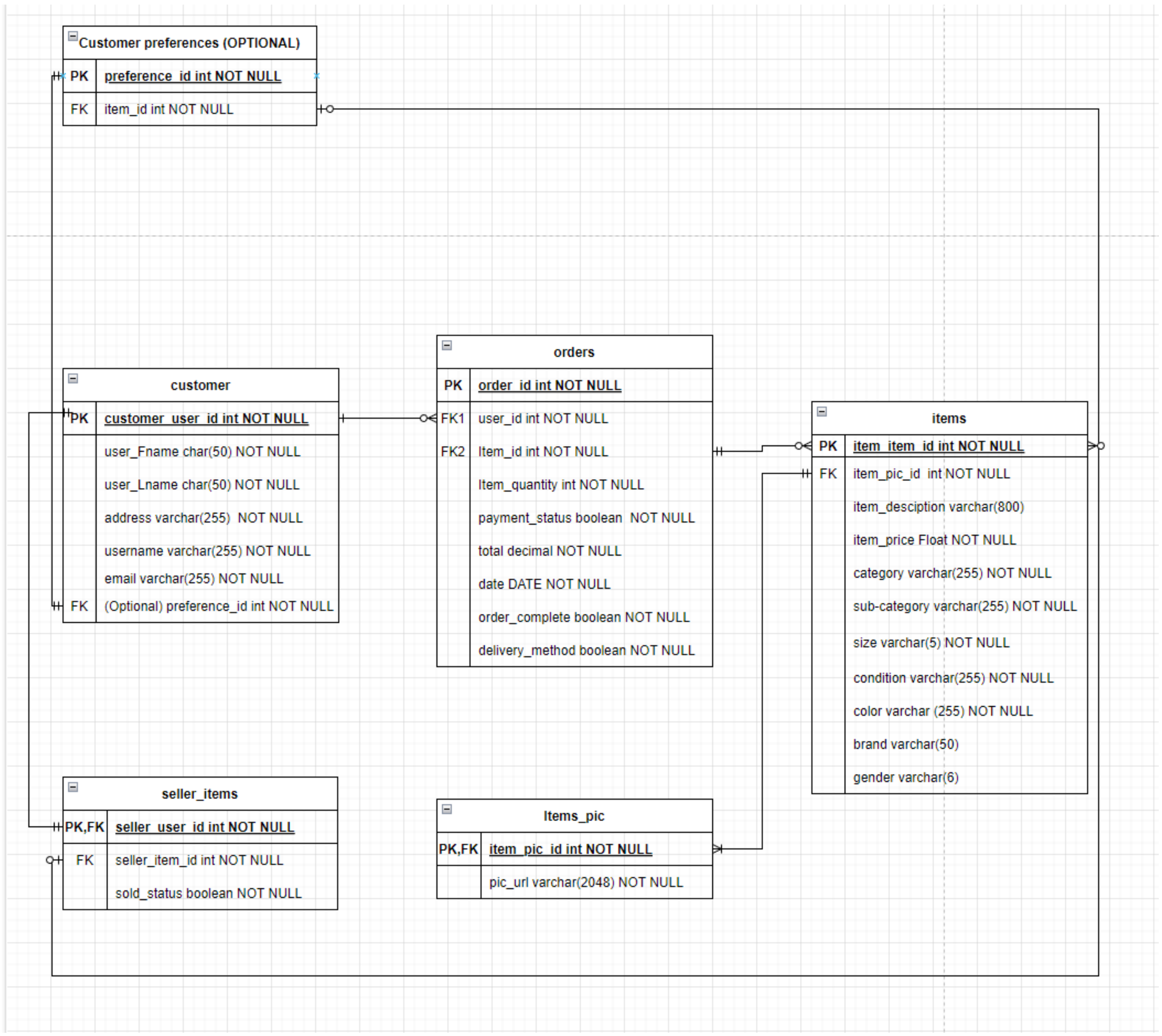
Extra: Profile Page
Extra: Check out page
Extra: Menswear and Womenswear pages
Extra: Brand pages
- Brand logos will link to their selection of products.
- Extra: Sort companies

Extra: Shipping Page/Return Page

Wireframe and Prototype

The wireframes and prototype can be accessed with the link below:
https://marvelapp.com/prototype/2i390356

Database



Customer preferences (OPTIONAL)
PK
preference_id int NOT NULL
FK
item_id int NOT NULL
orders
PK
order_id int NOT NULL
FK1
user_id int NOT NULL
FK2
Item_id int NOT NULL
Item_quantity int NOT NULL
payment_status boolean NOT NULL
total decimal NOT NULL
date DATE NOT NULL
order_complete boolean NOT NULL
delivery_method boolean NOT NULL
customer
PK
customer_user_id int NOT NULL
user_Fname char(50) NOT NULL
user_Lname char(50) NOT NULL
address varchar(255) NOT NULL
username varchar(255) NOT NULL
email varchar(255) NOT NULL
FK
(Optional) preference_id int NOT NULL
items
PK
item_item_id int NOT NULL
FK
item_pic_id int NOT NULL
item_desciption varchar(800)
item_price Float NOT NULL
category varchar(255) NOT NULL
sub-category varchar(255) NOT NULL
size varchar(5) NOT NULL
condition varchar(255) NOT NULL
color varchar (255) NOT NULL
brand varchar(50)
gender varchar(6)
seller_items
PK,FK
seller_user_id int NOT NULL
FK
seller_item_id int NOT NULL
sold_status boolean NOT NULL
Items_pic
PK,FK
item_pic_id int NOT NULL
pic_url varchar(2048) NOT NULL

Url Version:
https://drive.google.com/file/d/1S4gUuh7REqWuRf-yPFrfbdVGlAO7MD0E/view?usp=sharing

Technical Implementation

William:

The basics of all web pages like the About and Contact pages were implemented with HTML, CSS, and Bootstrap with simple formatting to place elements in intuitive locations based upon standard industry practices. The linking of all pages is done through React routes located in index.js. For the signup page, the information of the user is set to the backend through Axios with CRUDE operations and recorded in the database. There are also checks to ensure that the user has entered the correct password with confirmation. The formatting of the page is mostly reliant on Bootstrap with some custom CSS formatting for certain tags. The login page is built in a similar manner where an Axios request posts the information to the database. We are able to save the session id of the login through Axios authentication by sending cookies with information about the session to the backend. This allows to then access to information about a specific user, which will be used for other pages. The purchase page fetches information about the user from the backend, saves the order number, and fetches the purchase information from another table in the database. The sell page implements different components to be rendered. It's also set up to change the format of the page if needed when a different component is rendered. On default, it displays the items the user is selling with data fetched with Axios from the backend. The second button renders the items that have been bought by the user, which is also fetched from the backend. The last component of the Sell page saves the input of the user into an array and sends a form consisting of information about the item they are selling to the backend. There are also various checks to make sure all required information is given. For debugging, I would often console log any at the location in which I believe the error occurred and request to the backend. Postman was used to figure out the specifics of a request, so I have a better understanding of what I need to pass or what I'm pulling.

Meggie:

Navbar and Footer are present across all pages as it is present in App.js where all the routes are. In the navbar, there is a shopping cart icon with a count that records the number of items added to and present in the cart. When items are added to the cart, there will be an alert(bootstrap toast) indicating that an item is added to the cart. One issue is that the number will not be updated on the home page unless the page is refreshed, however, it updates on other pages. The home page has a carousel implemented through Bootstrap as well as products are displayed in a grid format. Information on the products is retrieved from the database using Axios and stored in states using the useState and useEffect hooks. The products are divided into sections using javascript functions including map and filter. Each product has an add-to-cart button that adds items to the cart. When users click on the picture of the item, they will be redirected to a product detail page that contains a description, size, clothing condition, and other information. This redirection is done through the useNavigate and useParams hook from react-router. The use navigate hook specifies the path and useParams specifies the item the user has clicked into, in this case, it is based on the name of the item. There is also an add-to-cart function on the product details page. The add to cart function is done through the API and axios where the product id and amount parameters are specified by user input (clicking the add to cart button). The cart page has all the items added to the cart from other pages. The data is fetched from the database using Axios and put into a useState hook and is then mapped out. Adding, subtracting, and deleting the quantity of an item is done through Axios by specifying the id. If there are no items in the cart, the page will render an empty cart message. Price information, subtotal, shipping, tax, and total in the cart are implemented with the javascript reduce function. At the end of the cart summary, there is a checkout button. The button checks the number of items selected for checkout and proceeds to the checkout page. If no items are selected, there will be an alert to prompt users to select an item. This is done through the handle check function which takes in an event (checking or unchecking a checkbox) and checks the value of each checkbox. This value is then passed to a verification function in a useNavigate hook that navigates the user to the checkout page if items were selected. The function is not complete as it does not post or retrieve data to and from the backend. The checkout page is not fully functioning, however, user input for the address is complete. The Address of the user is retrieved from the database using Axios. To add a new address, users will have to select the add new address option in the dropdown menu and a bootstrap modal (window) will open up. Here the user fills out information including the name associated with the address, select the state, select the city based on the state selected, address, etc. State and city data are retrieved from the database and are initially stored in useState arrays. User input is defined by a separate useState hook for both state and city. The value for City is dependent on State and values are changed and specified in onChange functions.

Chaocheng:

The backend of this webpage was built using Maven, Spring Boot, and a MySQL database, and follows the MVC architectural pattern. The controller is responsible for receiving requests and data sent by the Frontend to the server in the form of URLs, and calling the corresponding model and view to fulfill the user's needs. The model is responsible for performing further business logic operations and accessing the database based on the data and requests received from the Frontend, in order to fulfill certain functional requirements.The viewer is responsible for displaying the data processed by the backend and returned to the Frontend. The backend has most of the functionalities commonly found in web platforms, such as registration, login, password modification, uploading avatars, user-info detail, updating personal info, uploading user's product, shopping cart, shopping, product details, adding addresses, order details, and logout, etc. After registration, each user's information is stored in the database, and a user ID is assigned to display their uniqueness. The login function is achieved by creating a session and storing the user's unique user ID in the session to maintain the login status of that user. The logout function is achieved by deleting the session that contains the user's unique information, thereby logging out and ending the session to exit the login state.

On the other hand, people can upload their own products by going to the uploading product page. After the user uploads the product information, the backend stores the product in the database. When the user opens the product display page, the corresponding database table will be printed, and all the uploaded product information will be displayed on the front end through the viewer. Users can add the desired products to their own shopping cart, which is bound to each user by their user ID. Therefore, the shopping cart is bound to the account, so the user must have a login status beforehand.

However, there is a prerequisite that users need to complete their delivery address before purchasing the product. Users can input the corresponding address on the Frontend, and Backend will store all this information in a database table. Later, users can choose the delivery address they want on the payment page. Then, users can go to their own shopping cart to select previously added products and corresponding addresses for transactional operations. When the transaction is completed, the corresponding items in the user's cart will disappear.

Michael:

Im responsible for the search page which is a particularly complicated page since it has multiple filters, complicated buttons, and a layout. It has 4 sections on the sidebar, one main section for the listings to appear in a row as results, and two headings for the two sections. The filters include a dropdown with two checkboxes, a section with two textboxes to take the minimum and maximum price respectively, two multi-level dropdowns aka nested dropdowns for menswear and womenswear sizes respectively where the inner dropdown has different sizes for each category of clothes (tops, bottoms, outerwear, shoes, etc). The final filter is for brands. Each filter can update the search once completed (checking a checkbox or pressing a go button for the min and max price section).

Technologies used are HTML, CSS, JavaScript, React, a little Bootstrap, and Axios for making Ajax calls to the backend which uses Springboot. The minPrice and maxPrice use input tags with multiple attributes. The attributes allow a pattern to be set, mainly a regEx expression, which allows the input to only accept numbers: text cannot be input into the box to begin with so no errors have to be worried about. An onChange attribute calls a set method that checks for validity before amending the value. Because we use react I need to have values stored in a useState React Hook which updates the state of something when called.

The two multi-nested dropdowns are particularly tricky and took a lot of time. Due to the complexity and lack of out, if box solutions that worked, multiple very different custom solutions were tried before one worked and was saved as a separate React Component from the search page.

The way it essentially works is by taking an array of objects which have children nested, just like a multi-level dropdown. It also takes a var indicating if it's open. Using these it loops through the array adding items as list elements with onClick handlers waiting to call handleClick to drop the inner menus. It calls itself recursively until all of the dropdowns is created. The bottommost elements are checkboxes with text to indicate size. It was designed this way so I could build a custom array based on the user's search with all matching items, grab only the possible sizes from the search and add them to the filter. Because this is also in react hooks have to be used to remember if a dropdown was clicked and to change it making it even more complicated

Data for grabbing all items is grabbed by Ajax and then applied filters and then placed in the Listings body for the user to scroll through.Css is used to style the page, using CSS grid and to position all the boxes and CSS Flexbox to position items within those boxes and plain CSS for styling. Bootstrap was also used for some buttons.

Additional Project Documents

Initial Project Ideas:

https://docs.google.com/document/d/1n48ZPAC9kuGrmYLo0t_VQQGLkQ9wWTOuU0yUhxoa
Lm8/edit?usp=sharing

Github:

https://github.com/William321012/Thriftara

Database Design:

https://app.diagrams.net/?src=about#G1S4gUuh7REqWuRf-yPFrfbdVGlAO7MD0E

Member Background

William Lin(Computer Science Major with Multimedia Computing Minor): I'm interested in front-end development and have created simple web pages using Html, CSS, and JavaScript. I also have some experience using other programs like NetLogo, Processing, Clickteam Fusion, Unity, and more.

Meggie Cheng (Multimedia Computing Major): I am interested in frontend/web development and game development. I have some experience creating simple websites with React and Bootstrap, created CRUD API with Firebase, and games with Unity.

Michael Newman (Computer science Major with Physics minor): Im interested in full-stack web dev and using/managing data in general. I have some experience creating simple websites and creating a CRUD API with Java/Springboot.

Chaocheng Zhu(Computer Science Major): I am interested in Java full-stack development. I have some experience developing a website that includes CRUD, transaction, and JDBC with Java.

Recording Guideline

1. Slide about the programs used and a general background of the project. (3-4 Mins)(William)
2. Description and functionality of the home page, cart/checkout page, navbar, footer, and some code to explain the functionality. (3-10 Mins)(Meggie)
3. Description and functionality of the login, signup page, and how it works with the backend. Also, go into detail for the purchase and sell pages with some code examples for the functionality. (3-10 Mins)(William)
4. Description and functionality of the search page, maybe go into more detail about what you focused on this semester. (5-10 Mins)(Michael)
5. Explanation about how backend functions are done, how users can call the functions, database access, and more. (3-10 Mins)(Chaocheng)