Startup 05: DYNAMIC UI MOCKUPS

Founders:

David Zagreda
Seyedeh Arta Razavi
Colin Stern
SeihakRithy Muth
Brendan Shaughnessy
Jeremy Lee

***User Profile* (*Jeremy Lee*)**

Bugs: The props that were non existent in the previous React startup assignment are now there to make the website fully in React. The page itself cannot be refreshed yet. In order for the buttons “Edit Profile” and “Edit Listing” to work, they need to link to pages to allow users to edit their profile/listings that have not been implemented yet. Additionally, the search bar is yet to be fully functional in the Navbar.
Cut Product Features: The Help ribbon will be taken out for the remainder of the semester as it is not a feature that UBarter does not need to prioritize for now. If anything, it will be a future requirement once we get the main functionalities of the front and back end working. Another feature I cut from the initial React mockup is the “Account Settings” button. I did not see a need for it, nor did I find any settings separate from those that could be addressed in “Edit Profile”.

Main React Components:
Navbar: Almost every main page on UBarter will incorporate the shared Navbar component.

User Profile’s main components currently are *Userprofile_Panel* which encompasses the section with the book items.

Another one is *Userprofile_Sidebar* which includes all of the user profile’s information such as a user profile picture, name, and title. The sidebar includes an option to “Edit Profile” albeit the button is not properly implemented yet. The ribbons on the bottom of the profile section of the page: “My Listings” and “Matches” are linked to their proper sections on the website. “My Listings” links to the user profile and “Matches” links to the Matched page.

Userprofile_Item is a new component added since the last startup assignment. It is very similar to the Search feed item and Index item components. The component contains infor. As mentioned previously, the”Edit Listing” button is not fully implemented.mation regarding a book on the user profile page. Right now, in the mockup user profile page, there are two item components. The “Edit Listing” button would edit the props that are inside the item component if it were fully functioning.

Home **(*Seydeh Arta Razavi*)**

Bugs: When you click on the user image it just links to the default user page instead of that user's specific page. Also the search bar is yet to be fully functional in the Navbar. There are items in the user feed that are currently not click able the ideal functionality would be when you click on an item that it will take you to the message page to message that user. When the user clicks on their own items for sale it should take you to the edit items page but that functionality is missing.

Cut Product Features: We cut our item information list down to just book title price picture and isbn number and cut out author and publisher to downsize our database.

Main React Components:
Navbar: Almost every main page on UBarter will incorporate the shared Navbar component.

Home's main components currently are:

Home_Feed: this includes a feed of items for sale by other users (these are home_feed_items) which the user should be able to click on and contact those people to buy their items.

Home_Feed_Item: these are items that are for sale either by the user or by other users. The home_feed_item includes a picture of the item for sale the name of the item and the price of the item.

Home_Side_Bar: the home side bar includes the user's image and the user's username. Clicking on the user image takes you to the user profile page. The sidebar also includes the items the user has for sale. The functionality that is missing from this page is when the user clicks on their own items for sale it should take you to the edit listing page but that functionality is missing.

***Message* (*David Zagreda*)**

Bugs: The links for Beta, and Alpha link to Jeremy Lee’s Userprofile because we have not implemented profiles for Alpha and Beta. Also, my sidebar, and main panel do not feature scrollability because I have not figured out how to properly introduce scrolling using CSS. Lastly, the message button next to the message bar does not actually work, rather the user must hit enter to send a message.

Cut Product Features: I might decide to cut the conversation search bar in the sidebar panel, right now it provides no functionality. Also, I am going to remove the message button next to my message bar because it makes more sense for a user to hit enter when done sending a message, rather than hitting a messaging button.

Main React Components:
Navbar: Almost every main page on UBarter will incorporate the shared Navbar component.

Message’s first main component is *Message* which renders the entire messaging page, a.k.a the sidebar panel, and the main panel where users actually talk.

Another one is *Message_Sidebar_User* which is used for creating a new sidebar user. Each sidebar user includes fields such as avatar, username, a timestamp of the last message sent, and the number of unread messages you have from that particular user. Also, the user can click every username in the sidebar panel to link to the appropriate userprofile.

Next, we have *Message_Main_User* which is used for creating the main user for the main messaging panel. Each user in this field contains an avatar, username, and timestamp for when they sent each message. Also, similar to the sidebar users, the user can click the corresponding usernames in the message panel to be linked to the appropriate userprofile.

Lastly, I added a new component called *Message_ChatInput* which creates the message bar at the bottom of the message panel. This component provides functionality for allowing the user to send a message, and upon hitting enter, the user is able to send a message.

***Search* (*Colin Stern*)**

Bugs: The navbar only does a search if the query is made from the search page.
I intend to fix this by passing the query to the Search component via the URL.
The navbar query must be an integer between 1 and n, where n is the id of the item in the database. I intend to fix this when we implement MongoDB.
Cut Product Features: The left sidebar, which was intended to filter search results by item and location, has been removed due to complexity of implementation. It may be added back if there is time.
The Search_Feed no longer will sort items.

Main React Components:
Navbar: A query typed into the navbar on the search page will initiate a search for that item in the database.

Search: Sends a query to the database, then puts the returned list of item objects in the Search_Feed and renders it.

Search_Feed: Renders a list of items and displays the number of items, the search query and the way the items in the feed are sorted.

Home_Feed_Item: I borrow the Home_Feed_Item component from Home in order to display the items that the search returns.

Create Listing (Brendan Shaughnessy)

Bugs: The ubarter image for the navbar doesn't load. You cannot refresh the page. Listing objects in the database are not instantiated with the correct item id numbers. The submit button does not link to any page yet. The upload image button is not hooked up properly but the database will serve the correct placeholder image. Other pages that obtain the items from the database do not update with the new items instantiated by this page.

Cut Features: We've cut some of the fields from item objects and I need to rethink exactly what fields they need.

Main React Components:

createlisting: Container class that holds the lifted state for the form elements. Takes the state values and creates database entries for the user's item and the item they are looking for and combines them to create a listing entry in the database.

listingentry: Holds the forms for data entry and passes these values to a lifted state in createlisting so that item database entries can be made.

Matched (SeihakRithy Muth)

Bugs: The Previous and Next buttons does not correctly iterate through the Matched listings because the user's ID isn't saved locally across all the pages yet. I'm not sure how I'm going to fix this but I think there are two possibilities: 1. Have an App component that store the user's ID and all other pages are subcomponent of the App(root component) that way all components can have a user ID props. 2. Pass the user's ID from pages to pages, which may solve this problem but doesn't seem to be robust, however may make more sense for JSON Web Token later on as those token can change from pages to pages.

Cut Features: Originally, I was going to only display the items matched from the listings but that proved to be a bit tedious and require a lot more processing so instead, I'm just going to show the Listing that was matched and the users can see themselves what items in the listing matched.

Main React Components:

Matched: This is the main component, it acted as the Page. It basically gets its parameter from the router and poll the servers for data then pass it to its subcomponents to display. This page is pretty static so there really isn't a need for state changing. It only went from blank state to a fulfilled state when the server response with data.

Matched_User: This subcomponent display the user. Two of these are used in Matched, one for each user on the left and right side.

Mathced_User_Item: This subcomponent display the items in each of the User's listing.