Submission 05

April 2017

Team Mango Members

John Drakos | Yixiang Xu | Michael Feuerstein

Yi Fung | Bryce Bodley-Gomes | Ryan Semple

React UI functionality and changes

App.js: completely rewritten to use react-router.

Homepagebody.js: The “Recommended Items” carousel now dynamically pulls the items through the server to get an array of recommended items from the database to render (fixed bug where the title for an item was not displaying).

Navbar.js: Added “reset database” to the component. Added extra option to the select on search bar “Classes” (allowing to search for a specific class which will load classpage.js). Removed “watching” button from navbar (future implementation). Added routing so that:

- Clicking logo: returns user to homepage
- Clicking on search button: brings user to search page
- Clicking on “Profile”: brings user to their profile page
- Clicking on “Sell item”: brings user to a page where they can add an item to the the database that they are selling.
- Clicking on “Contact Us”: brings us to a static page (will be implemented at a later date)
- Clicking on “Help”: brings us to a static page (will be implemented at a later date).
- Clicking on “Classpage” brings us to the classpage. This class page is set for class ID: 1

Classpagebody.js: Class information(Title, description, instructor, credits, term)now pulls dynamically from from a class entity in the database using server functions. Additionally, the item rows for books and tech in the “Materials” tab populate class items based on the respective item lists from a given class entity. The given class entities depends on an ID passed through routing and the classpage.js.

Classinfo.js: Classinfo now has a dynamic link that displays the course subject of the class. The link takes the user to a search results page of category “Classes” which lists all classes of that subject.

Classitem.js: Each item is dynamically created utilizing item information passed in from the classpagebody.js. This information fills in the item’s picture path, description and

price. Clicking on an item picture links the user to an item page. The item page linked is based on the item's respective ID.

Searchpagebody: Each of the items in the search result is also pulled from the database.

Itempage: The item being viewed is pulled from the database as are the other items currently being sold by the seller.

Profilepagebody: The user data including personal information and sold/selling items are now dynamically pulled from the database. Changes on personal information and avatar (can only use image in img file) are now able to be updated to database.

Submission Form: Allows user to submit items to the database. Items require a title and a price. All other information is optional.

Bugs and Unimplemented Features

Homepage: No known current bugs.

Classpage: No current bugs. "Schedule" and "Other Info" tabs are not in use. Requires full search functionality so that class pages can be searched for.

Profilepage: The selling history container can only hold six items. Slide bar will be added in the future to fix this problem. Needed to downgrading history to version 2.0 to make images visible. Password is invisible which may cause users to forget their password when updating. A button to enable reading password will be added in the future.

Itempage: Have yet to implement the dynamic react components "Item Description", and "Item Hero".

Searchpage: Search does not return results according to search term pulled form navbar. It currently presents a list of items according to a premade search results list.

Submission Form: The formatting has been altered for some unknown reason. Plan is to revisit the css to reformat and fix the layout and design of the page. It is also currently undecided what the best way to store a new item's data is. (For example, an item can belong to multiple categories. Have yet to decide *best* way to store that information).

Plan to deal with this is to see how well new items are read from the database and modify how info is stored to make it efficient.

Cut Features

Navbar: Cut “Watch List”

Search (function): We did not implement the search function for classes. This feature will be added in future versions. For now the “classpage” link in the navbar takes the user to the class in our database.

SearchPage: Cut “Search Pagination” for now. The heart glyphicon associated with each item is not a button now but remains for decoration purpose.

Profilepage: Cut “edit” button which is used to make information editable due to its inconvenience.

Itempage: Cut “Add to Watching” and “Contact Seller” buttons. Seller information is now displayed in the Item Description component. The “Watching” feature has been cut as a whole.

Responsibilities

Bryce Bodley-Gomes: Built initial_data for the database, turned the homepage dynamic, fixed css with homepage so the spacing between navbar and body conformed to the spacing showed on other pages, revised navbar and app.js to add dynamic routing. Assisted in writing the startup submission document. Built various server functions to help with implementation for other pages.

Michael Feuerstein: Dynamized the class page. Assisted in linking items to respective item pages. Helped fix css related issues with pages. Assisted in research related to passing objects into routing. Assisted in fixing scrolling issues when linking to pages

while not currently at the top of a page. Assisted in UI design for search results. Assisted in writing the startup submission document.

Yi Fung: Made displaying the search results dynamic so that data is pulled from the object in the database associated with the user. Adjusted searchpage.css to accommodate for the dynamic code. Assisted in writing up the startup submission.

Yixiang Xu: Dynamized the profile page, assisted in enabling routing for the application. Enable user information and avatar changes through “save” button on profile page.

John Drakos: Dynamized the item page. Made changes in item css to accommodate dynamic React. Assisted in writing the startup submission document.

Ryan Semple: Dynamized the submission form to allow items to be added to the database at the click of a button. Assisted in writing the startup submission document.