SW Engineering CSC 648/848 Fall 2018

Vertical Prototype

Project/application Title and Name: GatorList

Team 12:
Pablo Escriva
Syed Kazmi
Stephanie Santana
Harry Zhang
Shi Yi Li (Jack)
Marlo “Mars” Sandoval
Johnny Huynh

Frontend

app.js

Our application initiates the tools that we need here, as well as stating the different routes of the webpages. The application is constantly updated using the ".latest()" function. It can also be ran on port 5000 locally.

package.json:

Contains the dependencies and tools our project will utilize.

knexfile.js:

Contains the details of the connection between the Heroku server and postgres database.

index.js:

This file contains the back-end elements of the home page. The possible webpage routes and the functions needed for the relative page belong here, i.e. queries asked by the web user and answered with the relative contends from the database. The logic behind displaying the user's search query and criteria are coded here. The logic for registering a new user is also contained here.

items.ejs

The front-end layout of the items webpage. Each item displays its official post title, its seller, its category, and its image. The body of the page uses a simple for loop to show all the relevant items within the user's selection/search query.

Backend

We have chosen to use Heroku postgres for our web application. In this vertical prototype we have only used two tables, "Items" and "Categories". For our index page (which is our items listing page) we needed to first get all the categories stored in the Category table so we can display them in the dropdown table.
Once the user chooses a category and searches for something using the searching bar we query the database with the necessary conditions and we display the resulting items. We used "iLike" comparison instead of exact search so that strings with similar characters appear in the results (case insensitive).

In case that no category was selected, we search in all of the items. Also, if the result of a search shows no items in the result we just deliver the list of all the items in the database. This is not our final searching algorithm but we are working on making it easier for the user to find more accurate results.

Backend connection with frontend

To make it easier to work with the postgres database we decided to use a npm module called “Knex.js”. It basically treats the SQL tables as javascript objects, that way doing queries is much more intuitive for people with no SQL background.
We can also make raw SQL queries if we need something more technical with the database.