Project Milestone 4

Revised Feature List:

Making an Account - *All features under this tree relate to account creation and verification of users.*

Registration Feature - Registration page where new users are prompted to enter information to create their new account. New user's information is posted to the database.

Verification Feature - Users are vetted to ensure they are current University of Colorado students or alumni through their registration email. Emails are checked for "@colorado.edu" domain to keep the submit button active.

Student Status Feature - Users are required to select their current relation to the university. Student/Alumni status input fields are hidden until users select their status where they subsequently choose their academic year or enter their graduation year.

Sign-in - *Features in this section relate to the login and log out loops of user interaction*.

Account Access Feature - Users sign in via the login page to access their accounts and recommended listings based on their prior search tags. User's accounts are fetched from the database at sign in.

Log Out Feature - Users log out from their account via the log out button located in the navigation bar.

Home Page - *Features in this section are located in the home page of Bufflist.*

Post Listings Feature - Section located in the center of each user's home page to post listings of their own with relevant information and pictures.

Search Feature - The search bar is located in the center of the navigation bar, utilizing this feature redirects the user to the listings page.

Recent Listings Feature - Located below the navigation bar. This feature shows recent website listings to the active user.

Listing Details - *Features in this section pertain to listings accessed from the listings page.*

Listing Carousel Feature - Large carousel to display listing images to the user viewing the listing.

Contact Seller Feature - Button located at the bottom of the current listing that allows a user to send their contact information to the seller.

Google Maps Location Feature??

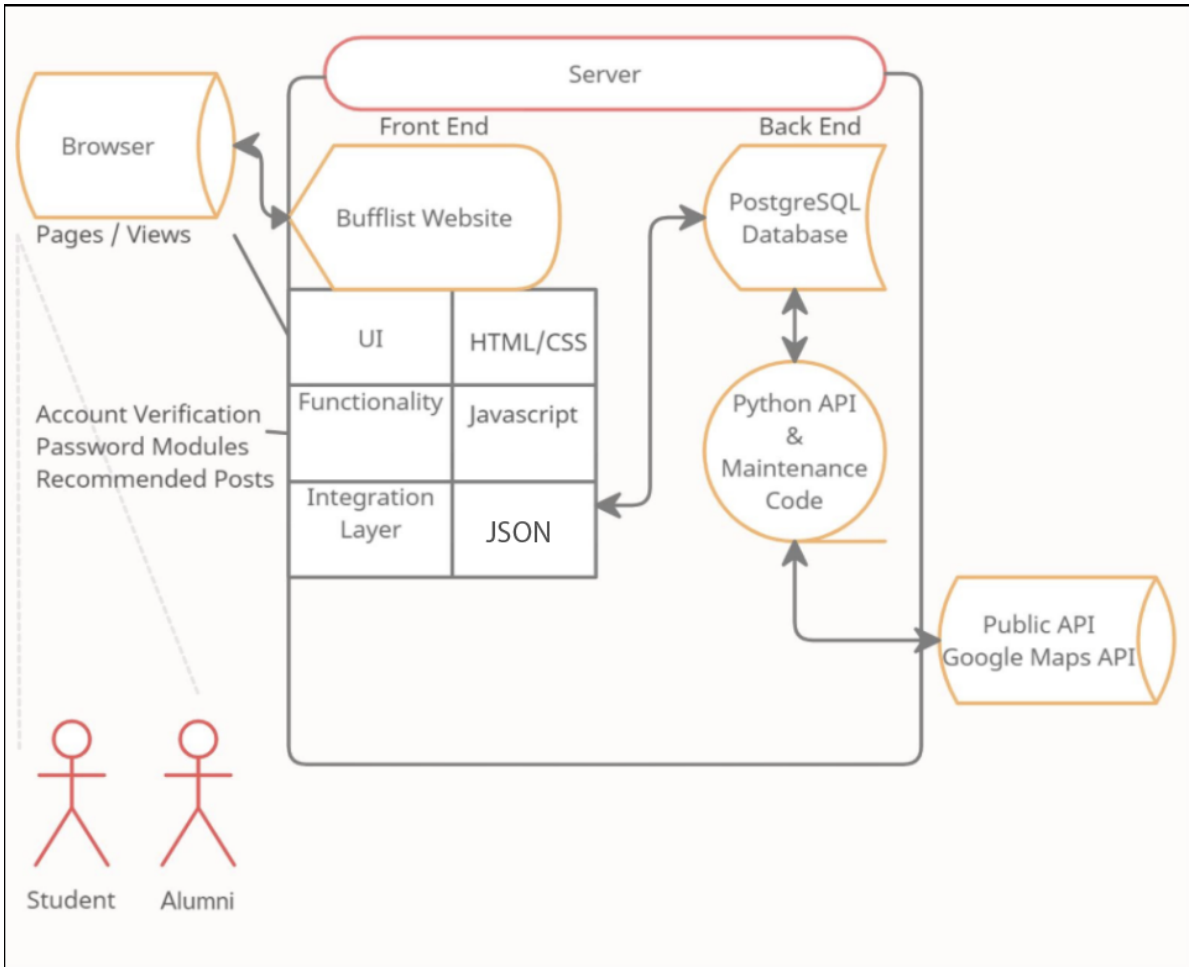
Account Settings - *Features in this section pertain to the user's ability to update their account preferences.*

My Listings Feature - Location for users to interact with or remove their current listings.

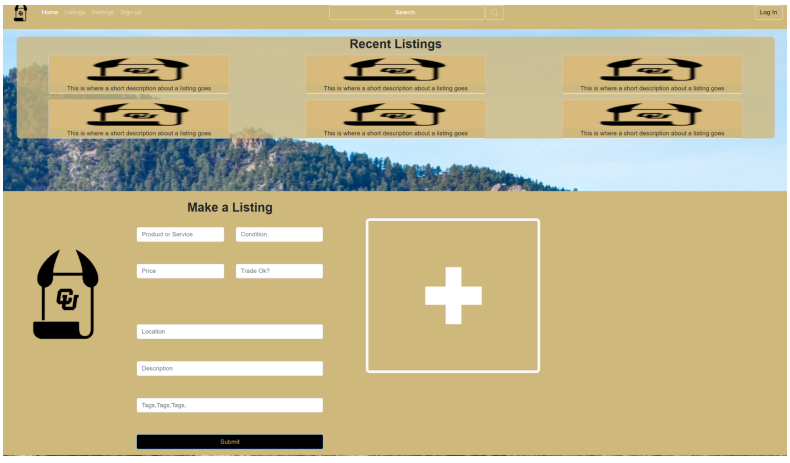
Account Feature - Settings pertaining to the users email, password, phone number, etc.

Preferences Feature - Settings allowing users to manipulate the viewing settings pertaining to their listings and account.

Architecture Diagram:



Front End Design:



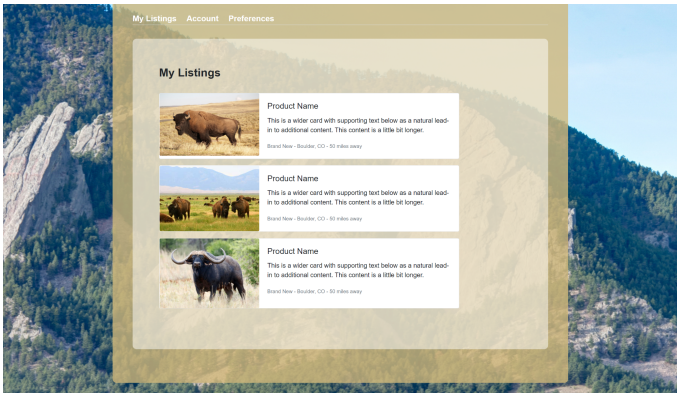
- Search Bar
- Recent Listings Feature

- Create listing feature

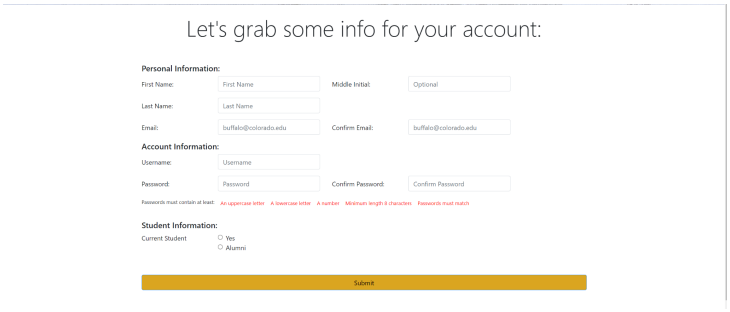
- View listing carousel

- Listing details

- Contact Seller

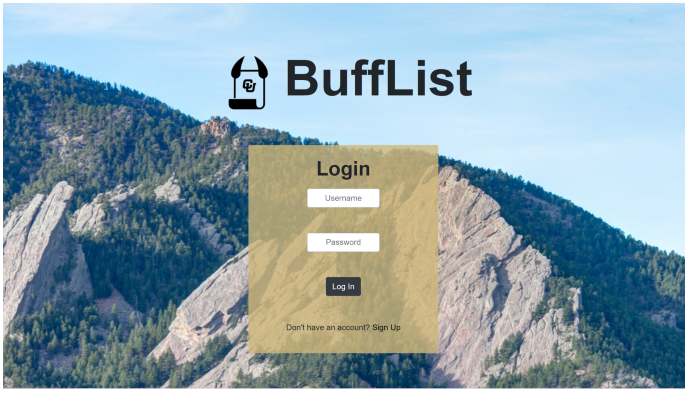


- Listings
- Account Settings
- Account Preferences



- Verify User Email

- Password Verification

	- User login

Web Service Design:

As an optional feature for the creation of listings, we could use the OpenLayers API to show the locations of listings as an embedded map if time allows. This feature is tentative right now.

Database Design:

Our entire backend end database design was built in PostgreSQL. When developing the database, we used a docker to create a local website to view how our tables and mock data would interact together. Our database has three tables called Listings, Pics, and Users, all of which are shown below. The Listings data table has to do with each listing posted on the website. The Listings tables has the following ten variables:

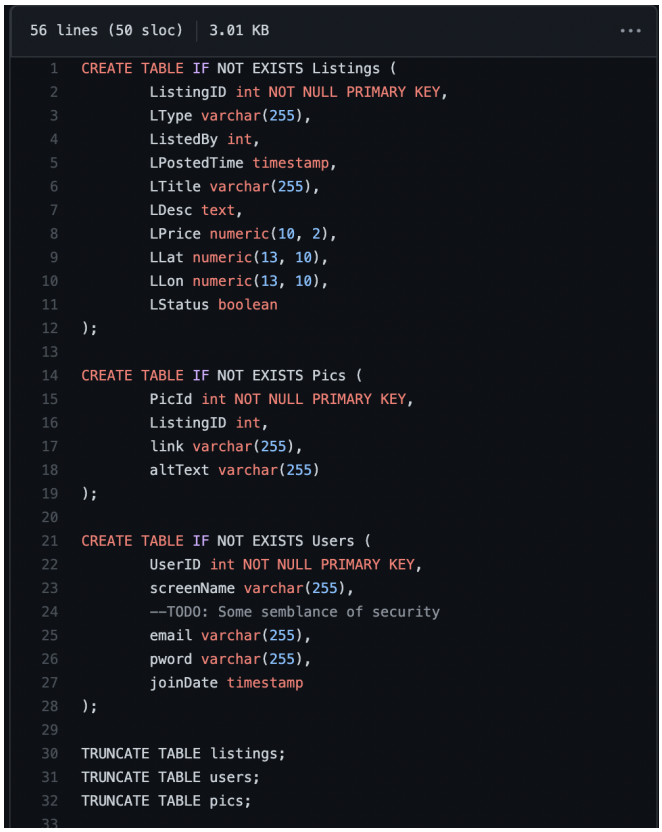
- ListingID - Integer variable that gives every listing a unique ID number. This variable is also in the Pics Table as each listing has a number of pictures that go along with it.
- LType - This name is shorthand for listing type which represents whether the listing is something being sold, a service, a possible trade, or many other things. This variable is a varchar.
- ListedBy - This integer variable represents a unique integer id of the person who posted the given listing.
- LPostedTime - This timestamp variable represents the time in which the listing was posted onto the website.
- LTitle - this varchar variable holds the title of the listing, which is a small sentence or few words giving an overview of the listing.
- LDesc - this text variable represents a longer description of a given listing which must be included in every listing posted on the site.
- LPrice - this numeric variable holds the price of the listing. This variable can be null if the listing is a trade.
- Llat - this numeric variable represents the latitude of where the listing was posted from or where the listing poster chooses to display.
- Llon - this numeric variable represents the longitude of where the listing was posted from or where the listing poster chooses to display.
- LStatus - this boolean variable simply holds the value of false if the listing has not been purchased, and true if the listing has already been purchased.

The second table, Pics, that is included in our database has to do with the pictures included in each listing posted on the website. This table allows each listing to load up pictures of the product or service that they are putting up onto the website. The Pics table holds the following four variables:

- PicID - This integer variable cannot be null and holds each picture's unique ID number.
- ListingID - This integer variable holds the unique identification number of each individual listing on the website. Although multiple pictures can hold the same Listing ID, only one singular listing can have a given Listing ID.
- link - This varchar variable holds a link to the picture.
- altText -This varchar variable holds a short description of the photo in the given listing.

The final table, Users, that is included in the database has to do with each user who makes an account on the website. This table allows each user to have individual usernames, passwords, and create and interact with their own listings. The Users table holds the following five variables:

- UserID - This integer variable which cannot be null represents a unique identification number for each individual member on the site. No two user IDs can be the same and no IDs can be null. This variable is also in the Listings Table as each listing posted by a user must be connected to the user.
- screenName - This varchar variable holds each user's desired username on the website.
- email - This varchar variable holds each user's email address.
- pword -This varchar variable holds each user's password.
- joinDate - This timestamp variable represents the date in which each user made their account.



56 lines (50 sloc) | 3.01 KB

```
CREATE TABLE IF NOT EXISTS Listings (
        ListingID int NOT NULL PRIMARY KEY,
        LType varchar(255),
        ListedBy int,
        LPostedTime timestamp,
        LTitle varchar(255),
        LDesc text,
        LPrice numeric(10, 2),
        LLat numeric(13, 10),
        LLon numeric(13, 10),
        LStatus boolean
);

CREATE TABLE IF NOT EXISTS Pics (
        PicId int NOT NULL PRIMARY KEY,
        ListingID int,
        link varchar(255),
        altText varchar(255)
);

CREATE TABLE IF NOT EXISTS Users (
        UserID int NOT NULL PRIMARY KEY,
        screenName varchar(255),
        --TODO: Some semblance of security
        email varchar(255),
        pword varchar(255),
        joinDate timestamp
);

TRUNCATE TABLE listings;
TRUNCATE TABLE users;
TRUNCATE TABLE pics;
```

In order to start working with our data table and work out possible issues that might arise with any given table, we developed five sample data for each table as follows:

```
TRUNCATE TABLE listings;
TRUNCATE TABLE users;
TRUNCATE TABLE pics;

INSERT INTO Listings(ListingID, LType, ListedBy, LPostedTime, LTitle, LDesc, LPrice, LLat, LLon, LStatus)
VALUES
    (1, 'Selling', 1, '20210901 10:30:05 AM', 'Physics Textbooks', 'I''m graduating and selling all of the textbooks a physics major would need for a standard 4
    (2, 'Service', 2, '20211001 08:38:05 AM', 'Plumbing', 'I know getting a plumber to come in and fix a sink can be very expensive, so I am offering a cheap pr
    (3, 'Selling', 3, '20211007 02:30:05 PM', 'Desk', 'I have a desk leftover from last year that I no longer need. It is in great shape and a fraction of the c
    (4, 'Service', 4, '20211013 08:50:05 PM', 'Odd Job', 'I am very handy in most odd jobs. Whether it be car problems, a broken cabinet, or anything else, I ca
    (5, 'Trading', 5, '20211018 11:10:05 AM', 'Calculus Textbooks', 'I recently changed majors from math to computer sceince, so I am looking to trade my calcul

INSERT INTO Users(UserId, screenName, email, pword, joinDate)
VALUES
    (1, 'John Doe', 'john@doe.com', 'john123', '20210901 10:30:05 AM'),
    (2, 'Sophie Smith', 'sophie@smith.com', 'Sophie357!', '20211001 08:38:09 AM'),
    (3, 'Nick Brown', 'nick@brown.com', 'nickb123', '20211007 02:30:24 PM'),
    (4, 'Lisa Sanders', 'lisa@sanders.com', 'Lis@1999', '20211013 08:50:38 PM'),
    (5, 'Bob Ross', 'bob@ross.com', 'b0br0ss123', '20211018 11:10:55 AM');

INSERT INTO Pics(PicID, ListingID, link, altText)
VALUES
    (1, 1, 'https://images-na.ssl-images-amazon.com/images/I/512yEu6OOPL._SX346_BO1,204,203,200_.jpg', 'physics textbook'),
    (2, 2, 'https://trusteyman.com/wp-content/uploads/2019/02/how-does-plumbing-work-e1548696261445.jpeg', 'plumbing'),
    (3, 3, 'https://i5.walmartimages.com/asr/3642a808-c507-44b3-8dac-be0457884606_1.321ec95aa2f61e4bcd07f36a7f66bf96.jpeg', 'desk'),
    (4, 4, 'http://www.jwslawncare.com/wp-content/uploads/2015/11/tools1-1.png', 'odd jobs'),
    (5, 5, 'https://images-na.ssl-images-amazon.com/images/I/510xyAJdgJL._SX387_BO1,204,203,200_.jpg', 'calculus textbook');
```

Challenges:

1. **Database connection with Front-end**

At this point, our website exists as two separate entities. We have built out a visual, non-functional front end and a backend which can only display information on a blank page. Our mission now is to bring these two pieces together, and allow the database to display on the pages we have built it for. This will populate our listings data on the home page and allow the listings page to be used as a template for any information. While this is a challenge, it is a core aspect of our project and we will put all of our energy into getting it working.

2. **Features that are too complex/unnecessary**

Many of the features we initially dreamed up for this project are going to be too complicated or time consuming. We thought up many cool ways to display listings and some fun preferences as well, but every extra button or section can lead to many more lines of code. Not only that, but some of our ideas regarding listing recommendations are going to be too complex to implement without new algorithms or additions to the listing data. In order to address this “scope creep,” we have better prioritized the features that are absolutely necessary for Bufflist and will only work on others as time allows.

3. **Converting to Node.js**

Our visual pages are currently built with pure CSS and HTML, and in order to integrate them with the database we have to make use of Node.js. This means that all of our work now must be converted to ejs files, partials, and functions on a server.js file. This is the main challenge we face this week, and we will make good use of our previous two labs for guidance.

Individual Contributions:

Nick - Latest Commit
https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-03/commit/c9ea30febacf63e45553bd323bde084668c3995b

I removed the "For You" feature from the home page and added a "tags" section to the "Create Listing" feature. Also added a phone number input to the registration form in signup.html. Lastly, created our final website directory structure in our Project Code that will bring together the database with the front end; I created blank ejs files for all our current pages and partials to be converted into. This week I will be helping Trevor begin to convert our HTML files to their respective EJS counterparts.

Alex - Latest Commit:
https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-03/commit/ed18e05b52aec3708b7e7075bf463629d6e224e8

I implemented some new features on the home page as well as the navigation bar throughout all the pages, as well as fine tuned some layout details. This week I will be helping Spencer convert our pages into ejs.

Trevor - Latest Commit
https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-03/commit/0c6af7ec4fe792c5144cfc4e3bdfcf60f50b32a1

Gio - Latest Commit
https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-03/commit/34d4fd6d9a47a86de434965f2b7eec6b177e64ba

Over the last weeks of the project, I have focused solely on the backend of our product. Working on the backend of the project has entailed collaborating with Trevor on developing the schema of our database, generating sample listings to see how our SQL data interacts with each other, and discussing how we will eventually merge our backend with the frontend of our project.

Spencer - Latest Commit:
https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-03/commit/e1130cc9b917292c63c61e70db59a2193bfe5ac5

Over the last couple of weeks I have been focused on fine-tuning a couple of pages, and implementing JavaScript to make things more functional. I would mostly call this front-end clean up, and I am now working on converting our basic pages into ejs files.