DSCI 551 Project Final Report

Title

iRecycler

Topic

Information exchange, environment friendly, resources saving, social networking.

Motivation

Our team bought a lot of new things to meet our basic requirements after arriving at university, and we've seen multiple times before that information on resale of similar items were posted on different platforms, such as Facebook and Wechat, at much lower prices, which made us feel somewhat unfortunate. In addition, we realized that some information was posted in an unorganized manner, which took time to read and compare. As a result, this time we decided to create an information platform about re-leasing, re-sale and other demands within a certain area (for example ktown, downtown, DPS and other places near USC), hoping it can help corresponding users exchange information effectively and save time and money to some extent. Besides that, we also hope it can do something with the environment by promoting people to buy less new items and instead to utilize their own items to the greatest. Afterall, we expect this platform to be a convenient information exchange tool that can meanwhile enhance the awareness on resource recycling and environmental protection.

Architecture & Components

Homepage

- Javascript, HTML, and CSS used
- Features
    - iRecycler logo at the upper left corner
    - Navigation bar at the upper right corner
    - Search bar #1 for item name keywords - dynamic & case insensitive
    - Search bar #2 for item category keywords - dynamic & case insensitive
    - Search bar #3 for zipcode - dynamic
    - Posts containing keywords are matched and returned as individual grids below the search bar

Login Page

- Javascript, HTML, and CSS used
- Features
    - Spaces for inputting email and password combinations
    - Login button for existing users - redirect to homepage if successfully logged in
    - Signup button for new users - redirect to signup page

Signup Page

- Javascript, HTML, and CSS used
- Features
    - Spaces for inputting username, email, password, city, zip code, and phone number
    - Signup button - redirect to login page if successfully signed up

New Post Page

- Javascript, HTML, and CSS used
- Features
    - User information on the left panel as a confirmation
    - Spaces for inputting item name, category, buy price, sell price, and picture of item
    - Submit button - show message if successfully posted

Profile Page

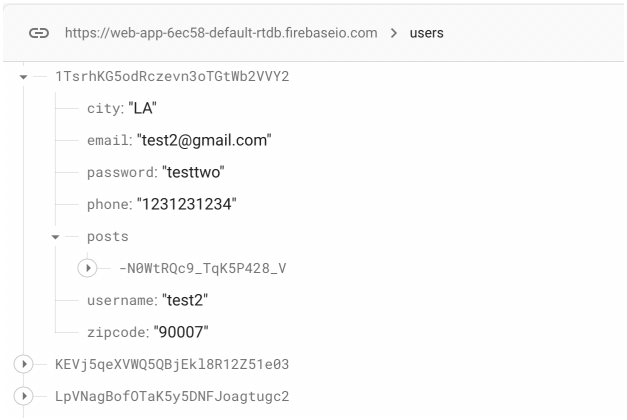
- Javascript, HTML, and CSS used
- Features
    - iRecycler logo at the upper left corner
    - Navigation bar at the upper right corner
    - Search bar #1 for item name keywords - dynamic & case insensitive
    - Search bar #2 for item category keywords - dynamic & case insensitive
    - Posts containing keywords are matched and returned as individual grids below the search bar - only current user's posts are returned

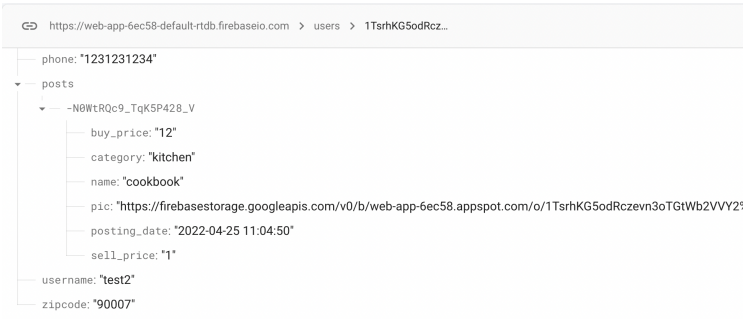
Data Flow

Database Establishment

- The data for this app, including user authentication and postings, will mainly come from users, but we also felt it necessary to collect some existing data to populate the database and for testing purposes.
- Data collection - user authentications
    - Generated fake usernames, emails, passwords, cities, zip codes, and etc. from fakenamegenerator.com
- Data collection - postings
    - Crawled item names, buy price, and images from the first five pages of each product category in Amazon Warehouse using Selenium and BeautifulSoup.
- Data processing
    - Designed the structure of our data
    - Randomly assigned posting data to the users
    - Generated sell price by using extracted buy price multiplied by a random multiplier
    - Generated random timestamp for each posting data
    - Uploaded the final data to Firebase realtime database

- Database structure
    - User_id under users, users from fakenamegenerator.com with sequential number as id and user sign up by function with their uid in firebase authentication.
    - User has attributes like user_id(key), user_name, age, city, email, gender, password, phone, zip code and posts including their history postings.
    - Under posts, every postings has it unique post_id, other attributes including name, category, buy price, price want to sell, a multiplier, posting date, and picture of it
- Sample of database

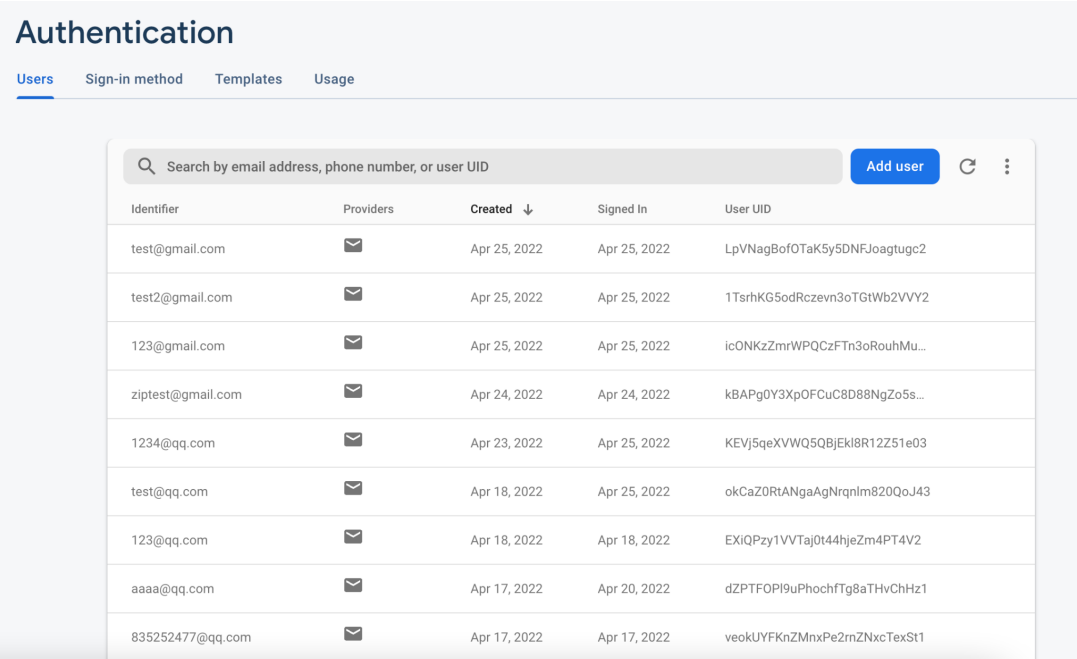




Database Utilization

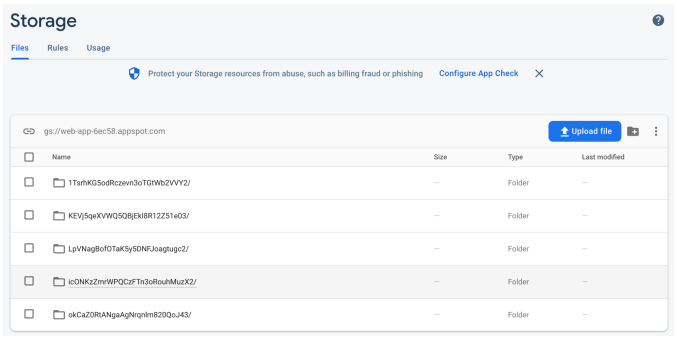
- Homepage (see homepage.js)
    - Added Firebase Realtime Database configuration in Javascript and initialized the database
    - To load all the posts stored in the Firebase database, data is referenced by using **firebase.database().ref("users")**, which indicates references on the "users" node.
    - Then, function **fetchPosts()** is created to push the posts into an array of objects named "postArray", with each object corresponding to a post and containing key-value pairs (keys: city, phone, …).
    - Next, function **displayPosts(postArray)** is created to transform each post object into an html string by extracting values from keys.
    - For filters on name and category, **.filter()** is used on "postArray", and the filtered posts "filteredPosts" and "filteredPosts2"are transformed into html strings using **displayPosts(filteredPosts)** and **displayPosts(filteredPosts2)** correspondingly.
    - For filters on zip code, **.match()** is used on **post.zipcode.toString()**, and the filtered posts "filteredPosts3" are transformed into html strings using **displayPosts(filteredPosts3)**. In the match methon on string of zipcode, we use **match('^'+input_zipcode)** to select the item whose beginning zipcode is same as out input zipcode.
    - All html strings are formatted with CSS styles.
- Login Page & Signup Page(see login.js & signup.js)

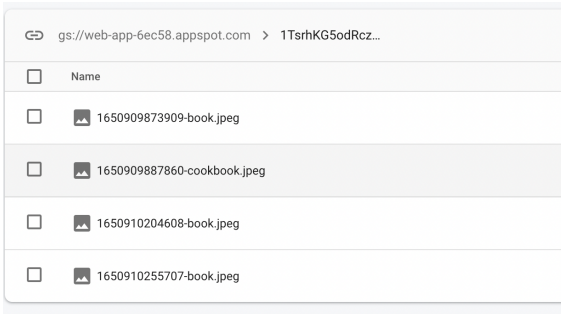
- Added Firebase Realtime Database configuration in Javascript and initialized the database
- To extract the user's input in the signup form, function **signup()** is created to get the input values.
- Then, the email and password values are passed to the user authentication database by using **firebase.auth().createUserWithEmailAndPassword(email, password)**. Once the user is created, other input values including username, city, zip code, and phone are pushed to the corresponding child node under the "users" node.
- With the authentication data stored, users will be able to login by inputting the email and password combination. The input values are passed to a **login()** function that uses **firebase.auth().signInWithEmailAndPassword(email, password)** to make sure the combination is correct.
- Sample of firebase authentication



- New Post Page (see posting.js)
  - Added Firebase Realtime Database configuration in Javascript and initialized the database
  - function **submitForm()** is used to extract input values from the form, including item, category, buy_price, and sell_price
  - Function **savePost(item, category, buy_price, sell_price, posting_date, url)** is used to push input data of item to firebase database by calling **firebase.database().ref(post_path)..push().set(object of input data)**.
  - Then, function **uploadInfo(item, category, buy_price, sell_price)** is created to upload picture of item to firebase storage by using **firebase.storage().ref().child(uid+"/"+file_name).put(file, metadata)** and then use **firebase.storage().ref().child(uid+"/"+name).getDownloadURL()** to get the

url of uploaded picture. After all these are down, **uploadInfo()** will call **savePost()** function to upload all information to firebase database.
    - add the key value pairs to the corresponding user's post.
    - Sample of firebase storage



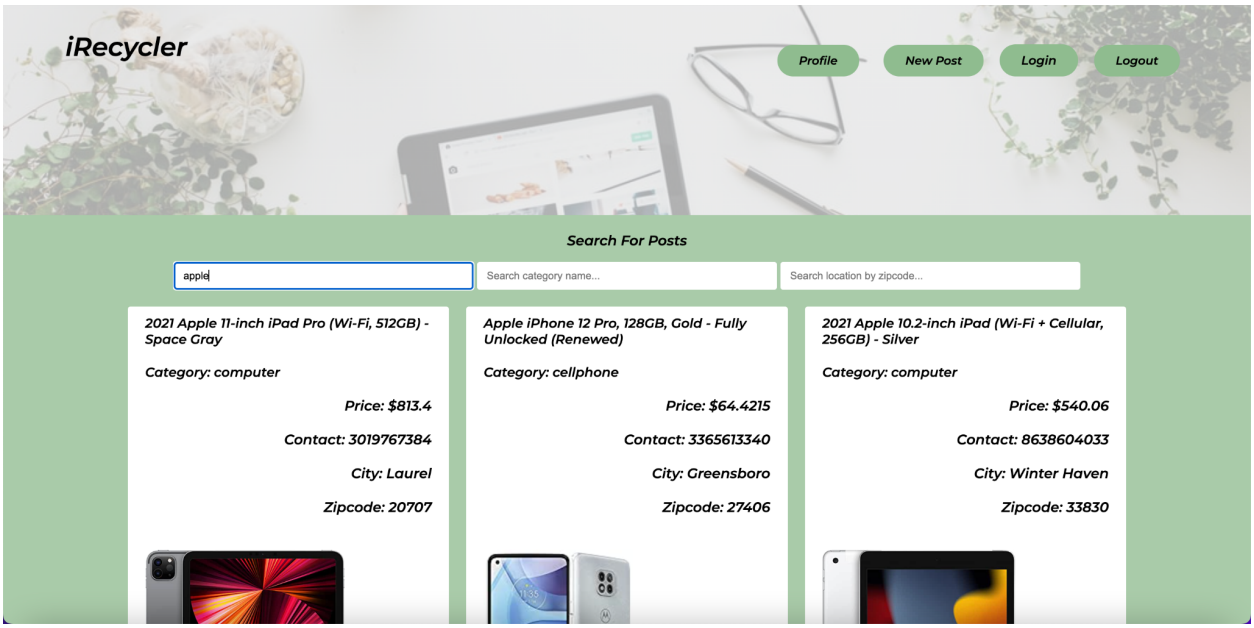


- Profile Page (see profile.js)
    - Added Firebase Realtime Database configuration in Javascript and initialized the database
    - First, **firebase.auth().currentUser()** is used to return the current user, which is saved as "user", so that user.uid can be used to indicate the uid of this user.
    - To load only the posts for this user, data is referenced by using **firebase.database().ref("users/" + uid)**, which indicates references on the certain uid child node under the "users" node.
    - Then, function **fetchPosts(uid)** is created to push this user's posts into an array of objects named "postArray", with each object corresponding to a post and containing key-value pairs (keys: uid, city, phone, pid, …).
    - Next, function **displayPosts(postArray)** is created to transform each post object into an html string by extracting values from keys.
    - For filters on name and category, **.filter()** is used on "postArray", and the filtered posts "filteredPosts" and "filteredPosts2"are transformed into html strings using **displayPosts(filteredPosts)** and **displayPosts(filteredPosts2)** correspondingly.
    - The html strings are formatted with CSS styles.
    - To delete posts, function **deletePost()** is created to remove a certain post with the id "pid" by using **firebase.database().ref("users/" + uid + "/posts/" + pid).remove()**.

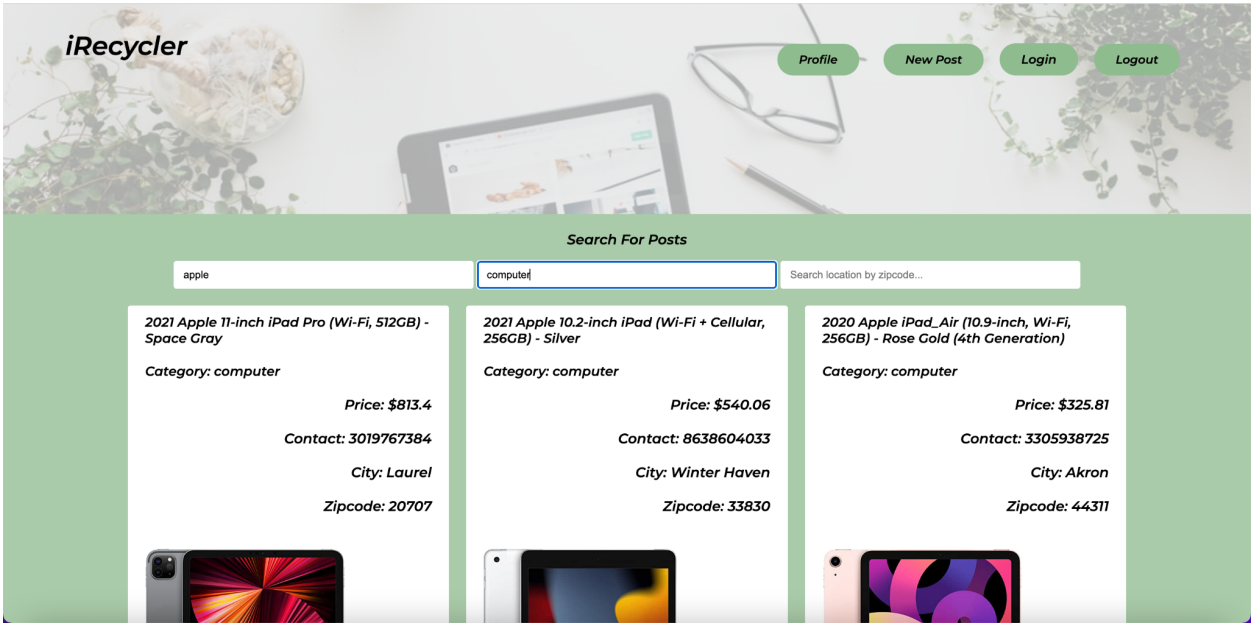
Main Functions w/ Screenshots

Homepage

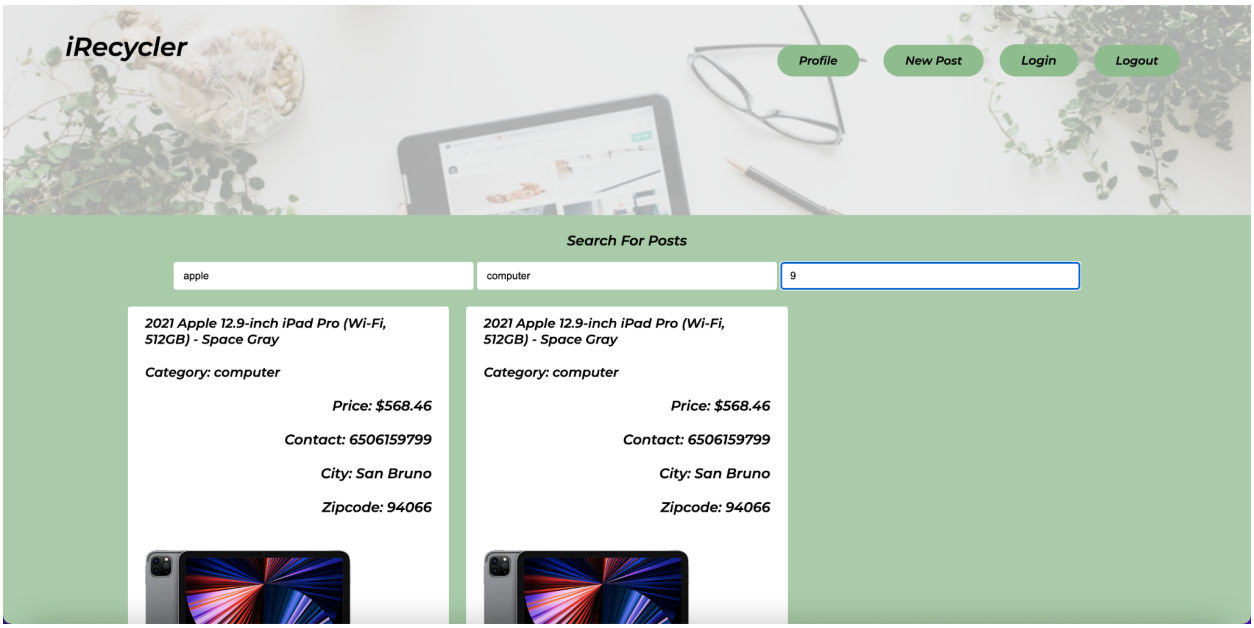
- Search item name



- Search item category

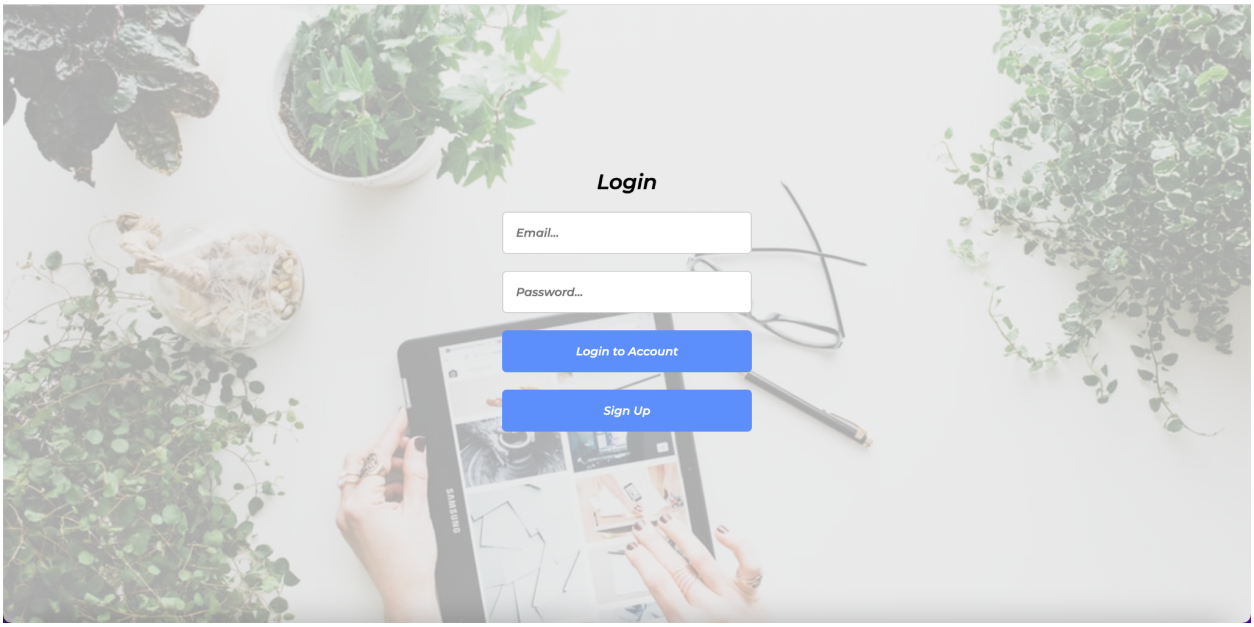


- Search item zipcode

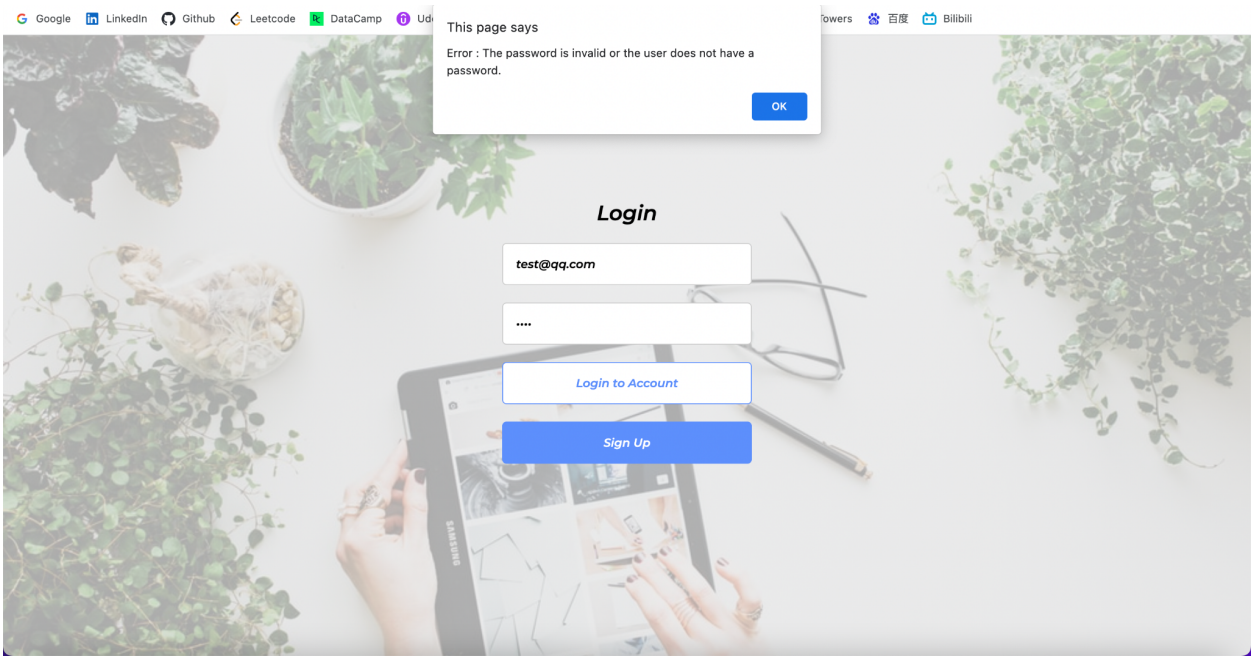


Login

- Required fields

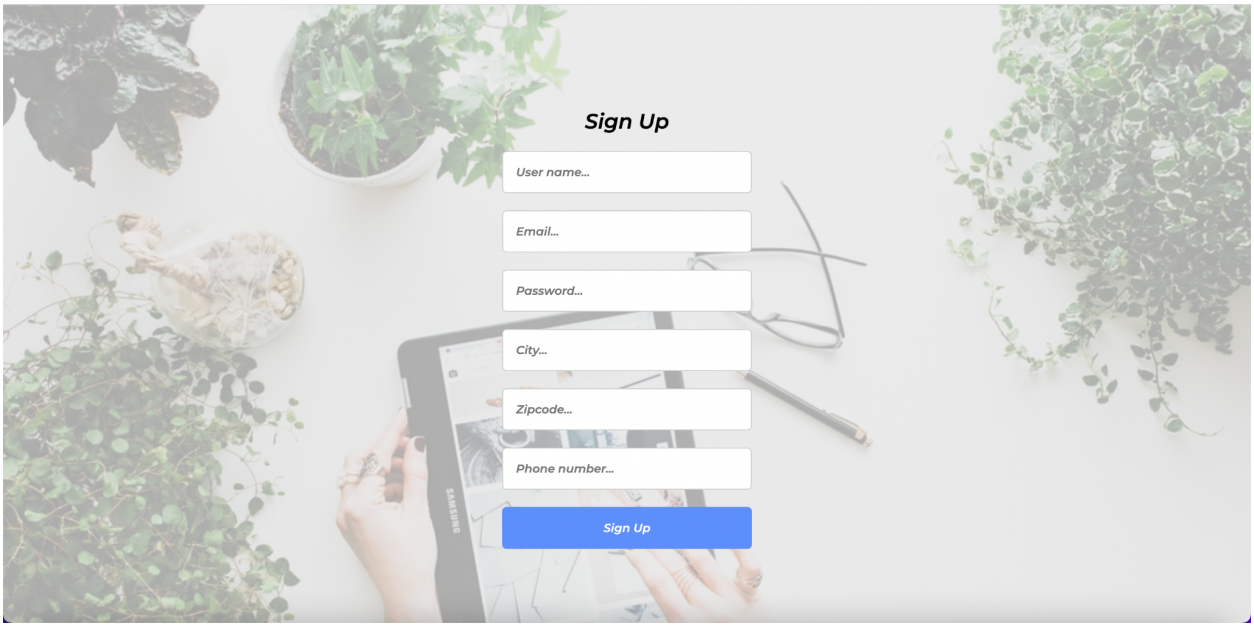


- Wrong email & password combination - alert

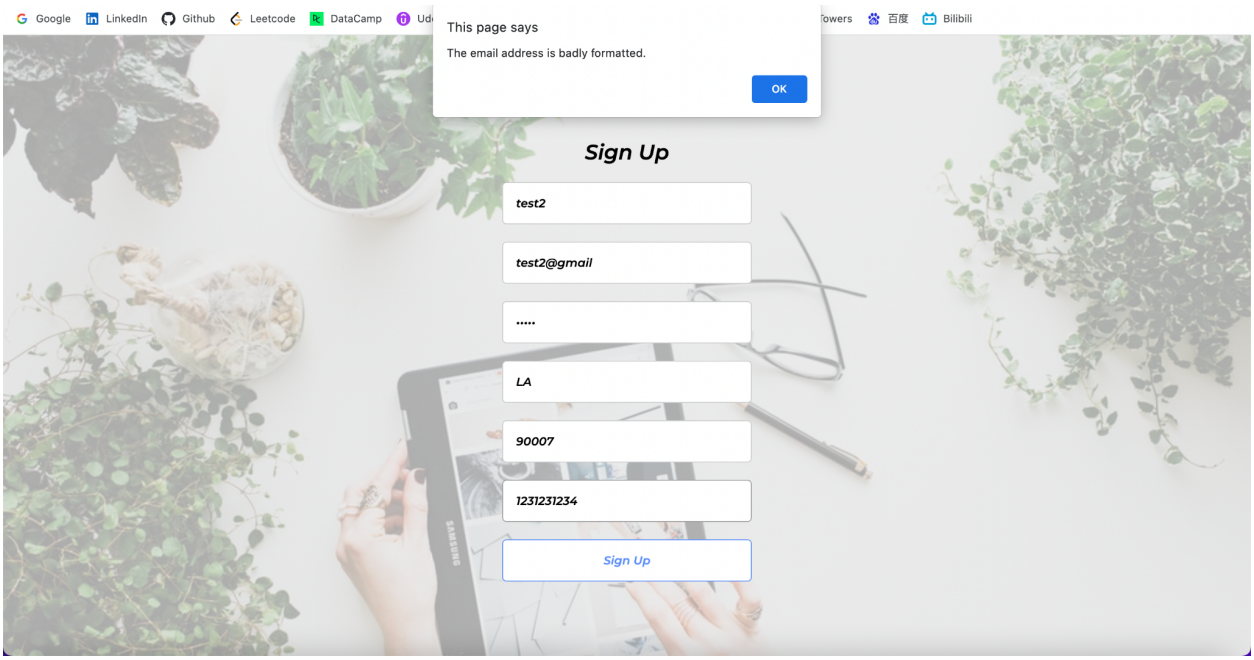


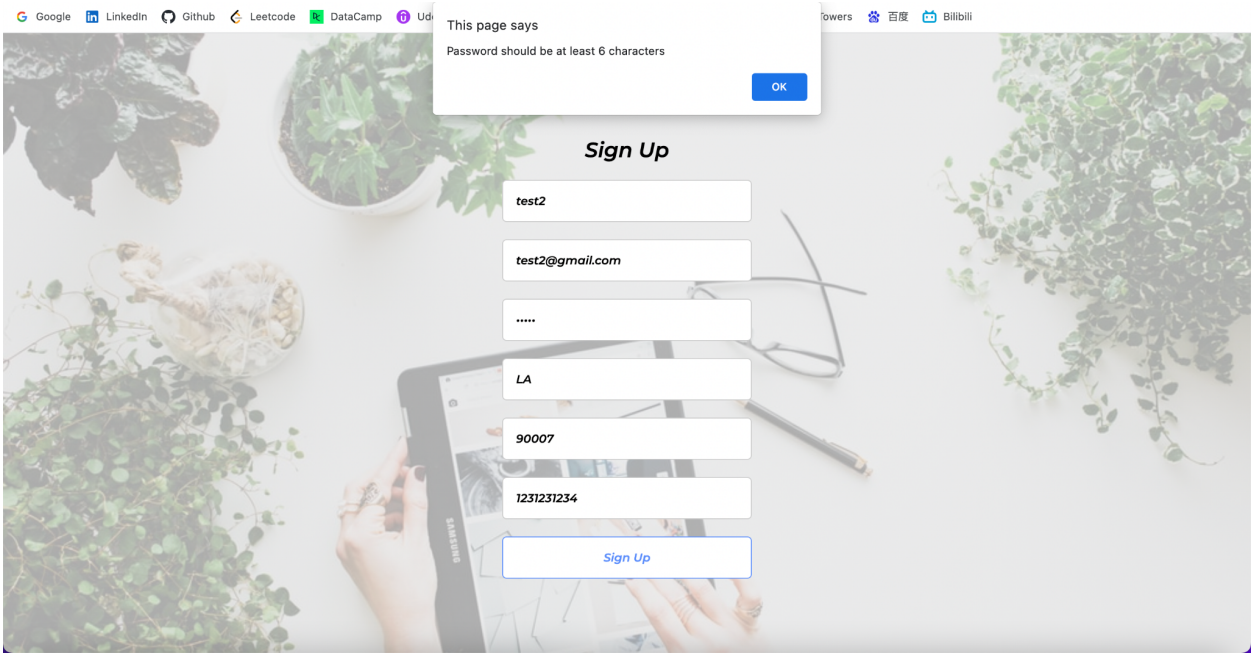
Signup

- Required fields

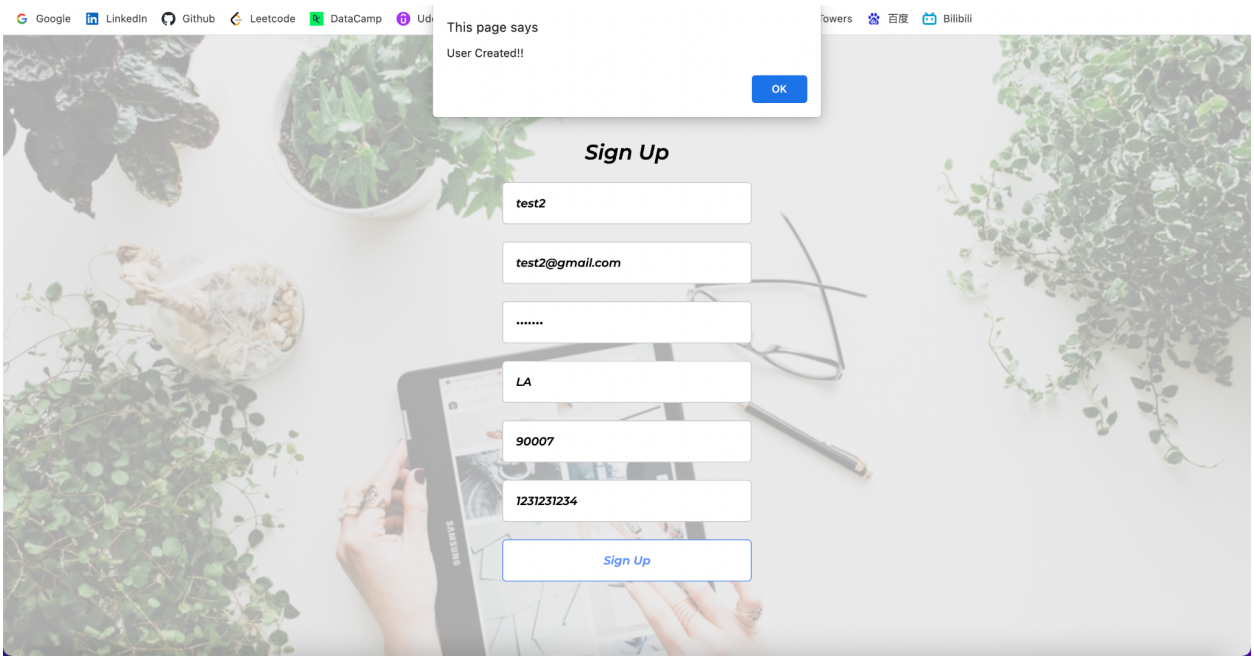


- Wrong email format/password length - alert



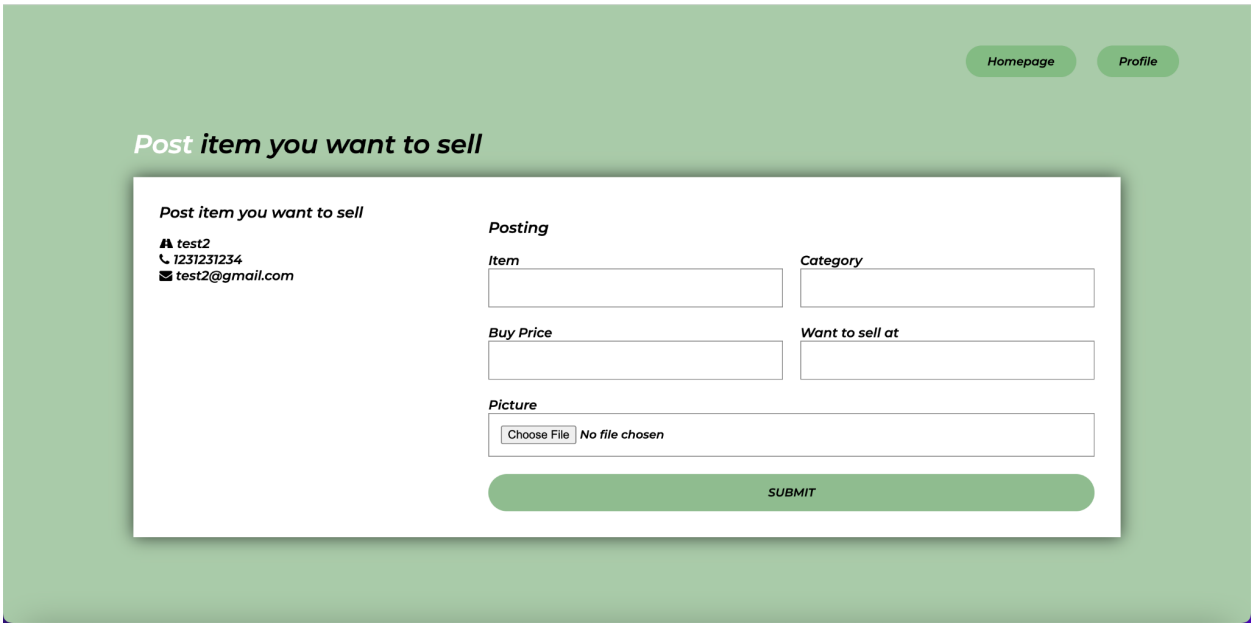


- User successfully created - alert

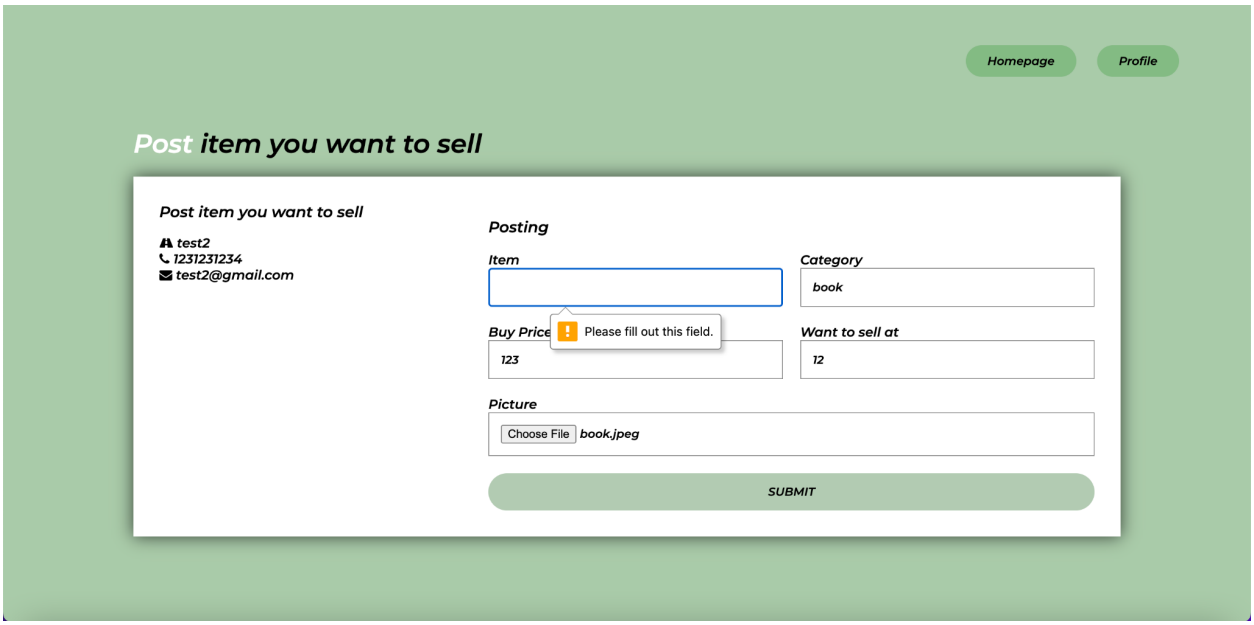


New Post Page

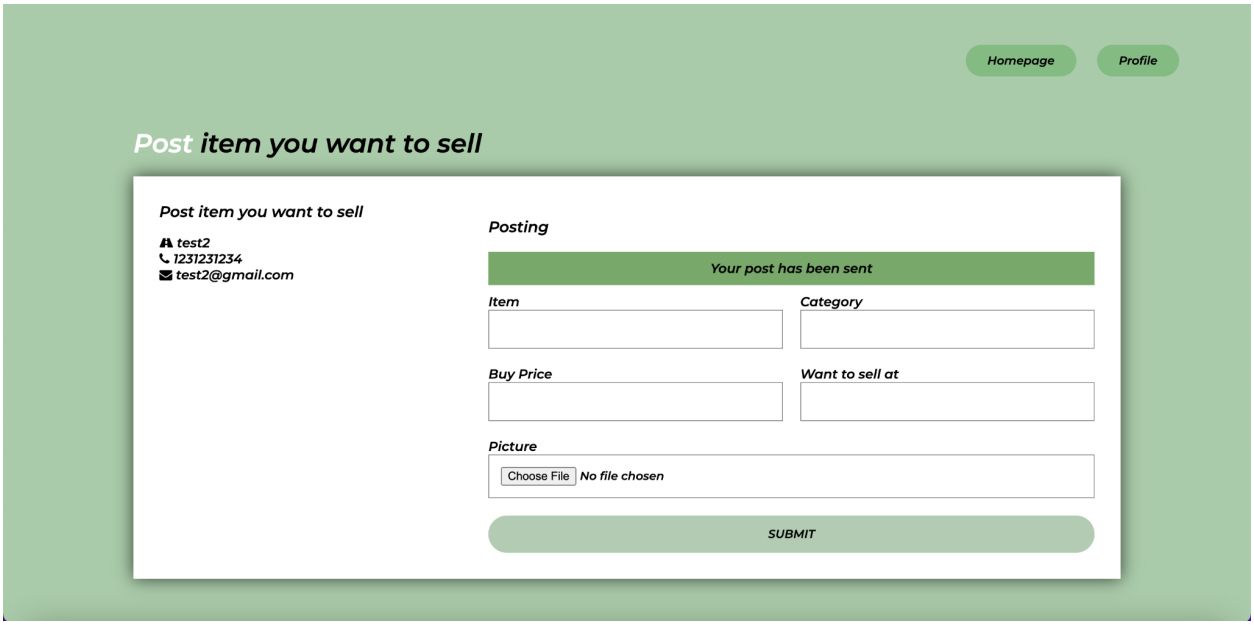
- User info & required fields



- Invalid input - alert

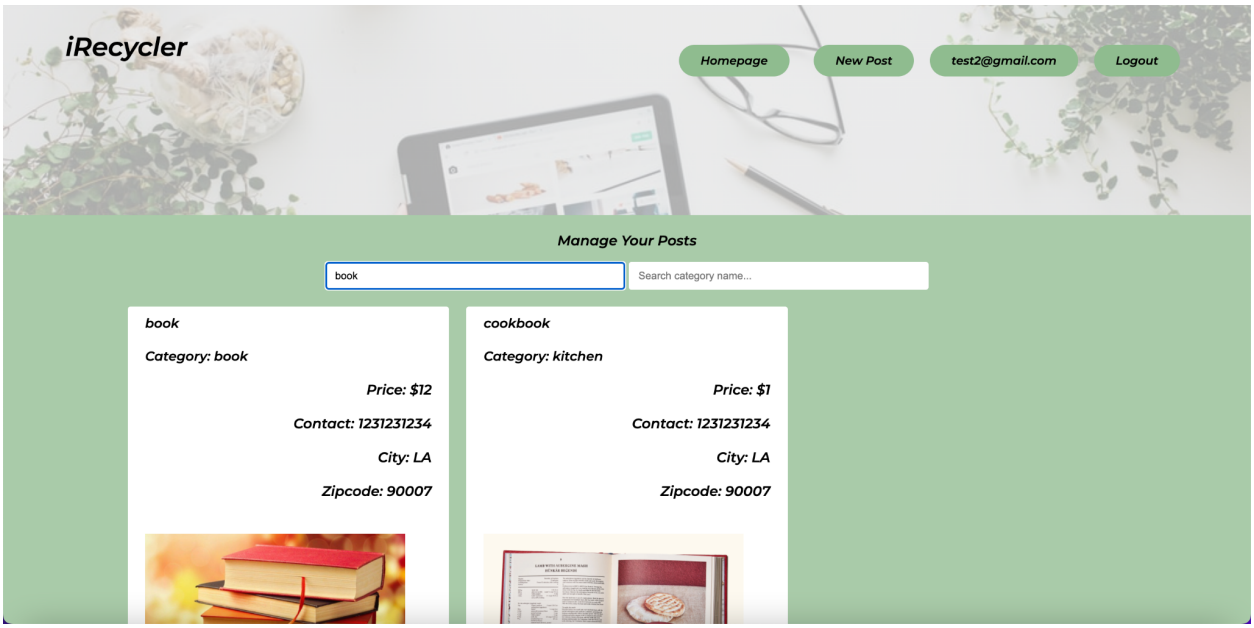


- Successful input - alert

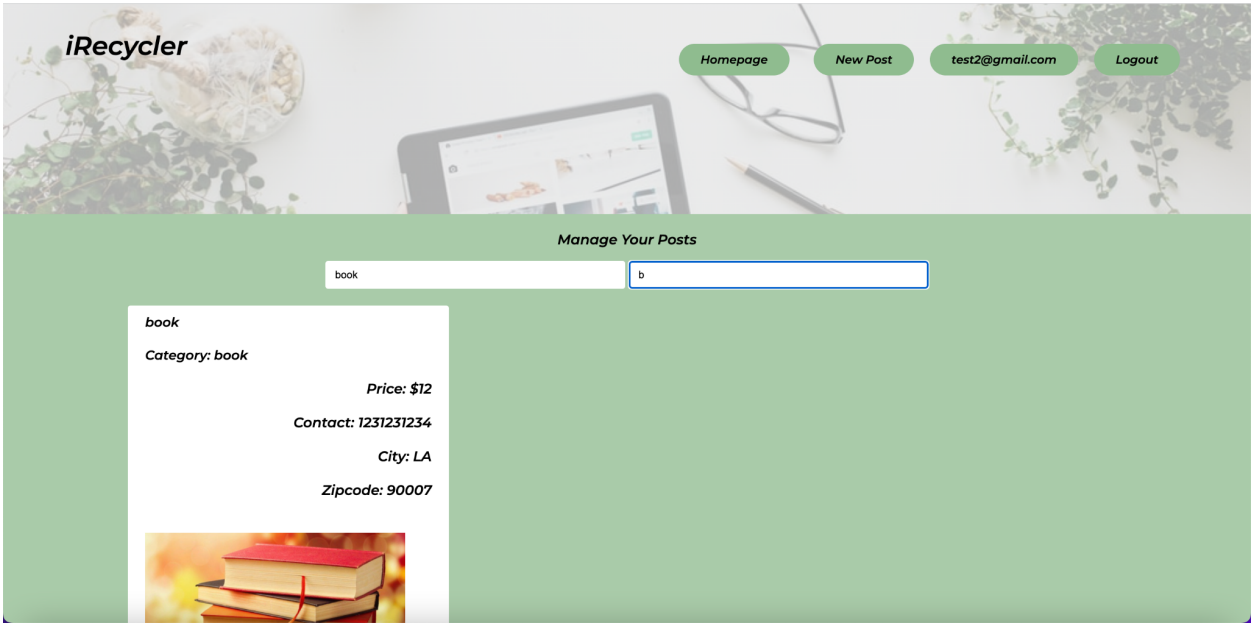


Profile Page

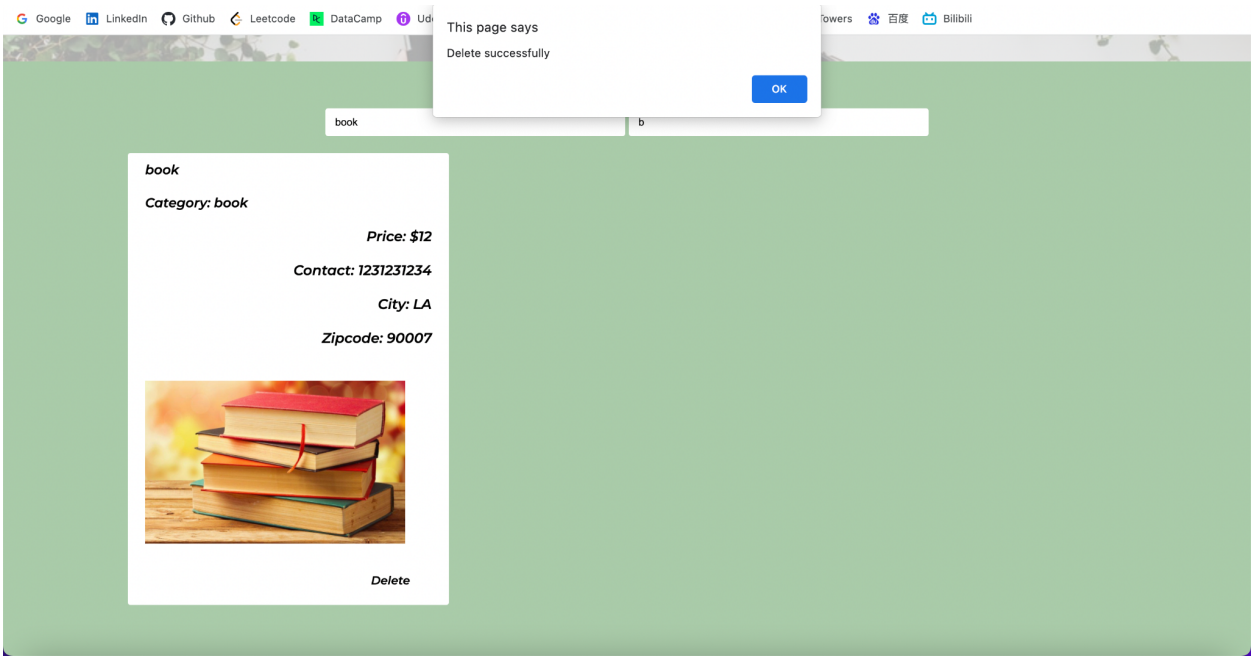
- Search item name

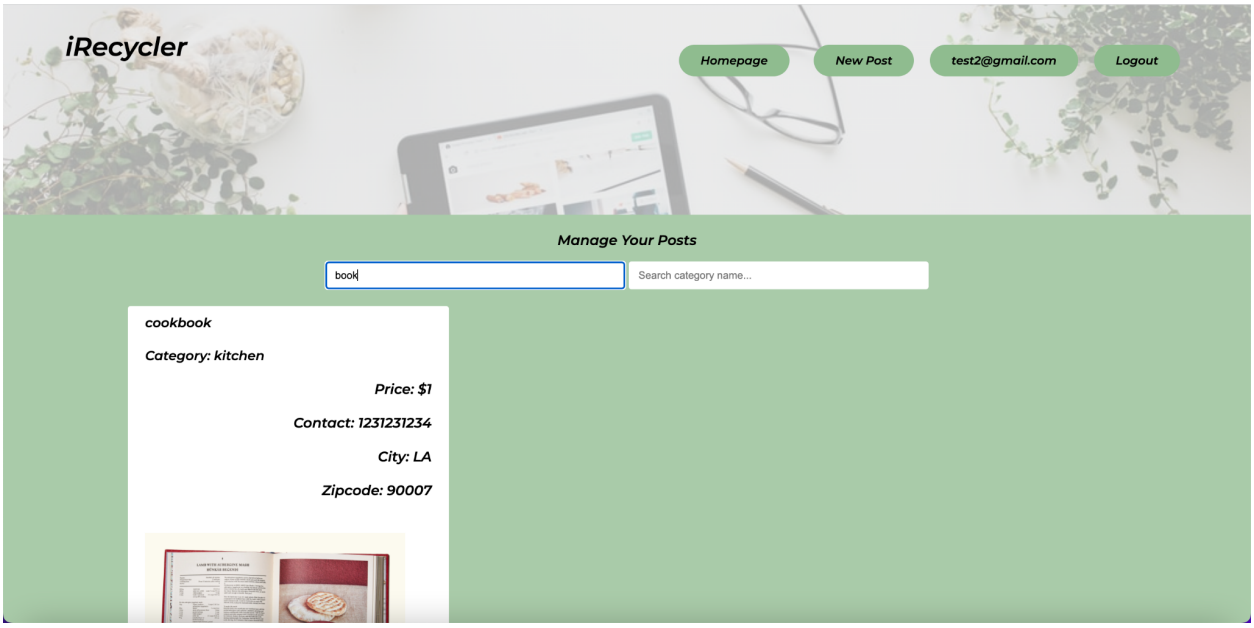


- Search item category

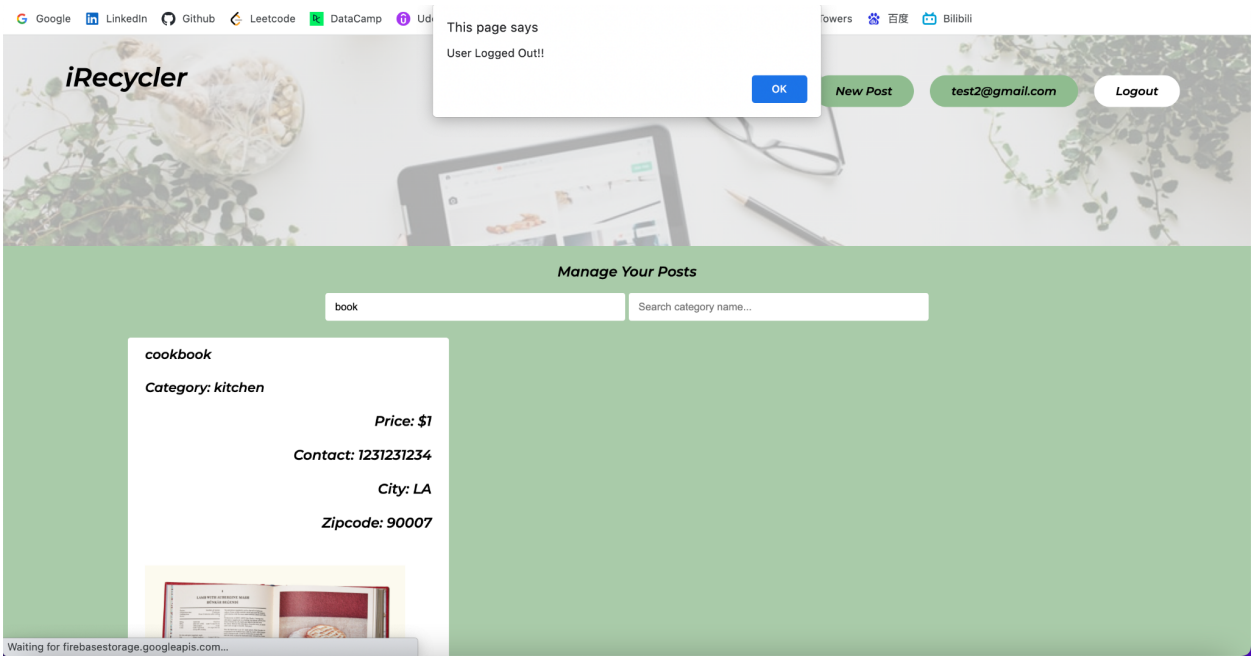


- Delete post


Google
LinkedIn
Github
Leetcode
DataCamp
Towers
百度
Bilibili
This page says
Delete successfully
OK
book
b
book
Category: book
Price: $12
Contact: 1231231234
City: LA
Zipcode: 90007
Delete


iRecycler
Homepage
New Post
test2@gmail.com
Logout
Manage Your Posts
book
Search category name...
cookbook
Category: kitchen
Price: $1
Contact: 1231231234
City: LA
Zipcode: 90007

- Log out



Learning Experiences & Challenges Faced

Database structure design

- We initially thought this part should be straightforward, but then we realized the structure could greatly affect how we would write scripts to fetch data in the following steps.
- Therefore, we spent some time designing and several times modification, we think we made the most organized structure that both makes sense and also facilitates data referencing.

Connections to Firebase

- We considered this as the most important part of our project as we need to fetch both the user authentication information and the post information from our database to the actual web page, and also delete existing data from database. This is the part where we encountered the most challenges due to lack of knowledge on Javascript and Firebase references.
- Fortunately, we have the lecture slides on this topic and also many online resources to make ourselves become familiar with Firebase configuration statements, database initialization, and functions associated with **firebase.auth()**, **firebase.database()**, **snapshot.forEach()**, and etc. Then, we practiced applying them to our database and successfully fetched data, uploaded data and deleted data.

Search & Filter

- After we fetched the data into an array of objects, we also spent time figuring out how to apply search and filter on this array of objects. We wanted to allow users to search for posts

by typing in case-insensitive keywords and also to apply filters to their searches based on category and location. This is also new to us.

- We discovered the filter function for an array of objects, and it worked very well picking out posts that contain a certain value for a certain key, in our case the item name, category, and the zip code. We also implemented the toLowerCase() function to make the search case-insensitive.

UI Design

- We wanted to make our pages more user friendly, but we have no previous experience on web page design
- Therefore, we consulted many online resources on the CSS language and learned by trial and error. Although slightly time consuming, we were still able to develop our own CSS style sheets.
- This was a really fun and satisfying process as we can see plain texts gradually turned into visually appealing components of the pages.

Team Members & Responsibility

Our team shares the responsibilities equally and members are able to benefit from practicing the same skills at the same time.

- Ying Wang - Project proposal & reports, database structure design and connection, javascript & html implementation, page UI design
- Ziyue Chen - Project proposal & reports, database structure design and connection, javascript & html implementation, page UI design

Codes and Relevant Files

https://drive.google.com/drive/folders/1i2Y9gX6JumokT4ESmNkMy1OfB6Kjc_ov?usp=sharing