SW Engineering CSC 648/848 Fall 2020
Team 4

Milestone 5
GatorTrader

Niall Healy (Team Lead, nhealy@mail.sfsu.edu)
Aaron Lander
Dale Armstrong
Lukas Pettersson
Joseph Babel
Vern Saeteurn

Date submitted	Date(s) Revised
December 18, 2020	

Table of Contents:

1. Product summary:

GatorTrader is a unique website that allows San Francisco State University students to search for, purchase, sell, and trade essential materials for their courses. Our free-to-use website provides users with core functions that will allow them to make use of excellent features such as search by category or class, post item listings, and a messaging system in addition to a clean user interface. **GatorTrader**'s core functions include:

1. **Unregistered Users**
    1.1. Shall be able to search for items by item name, class, category
    1.2. Shall be able to use their SFSU email to create an account
2. **Registered Users**
    2.1. Shall be able to post item/service **listings**
    2.2. Shall be able to send **messages** to sellers
    2.3. Shall log in with a username and password

GatorTrader is unique due to the need for a medium that provides SFSU students the ability to make transactions with other students. With **GatorTrader**, students are able to message owners of post listings so that they have all the information they need before committing to a purchase.

URL: http://ec2-3-21-104-38.us-east-2.compute.amazonaws.com/

2. Milestone Documents:

M1:

SW Engineering CSC 648/848 Fall 2020
Team 4

Milestone 1
GatorTrader

Niall Healy (Team Lead, nhealy@mail.sfsu.edu)
Aaron Lander
Dale Armstrong
Lukas Pettersson
Joseph Babel
Vern Saeteurn

Date submitted	Date(s) Revised
September 24, 2020	

Table of Contents:

1. Executive Summary:

As new students who are proficient in using technology enroll at San Francisco State University, a need for a platform that provides the ability to buy, sell, and trade essential items increases. Buying textbooks straight from a vendor is expensive, and finding good deals can be challenging. Our company would like to develop a free-to-use website that gives students the ability to find textbooks and other goods, at an affordable price.

Since many students attend the same classes at San Francisco State University, and classes usually use the same textbooks and supplies every semester, these goods should be able to be passed down from student to student each semester. This does not happen, however, because there is no communication medium in place to facilitate these transactions. Our company would like to provide this communication medium in the form of an intuitive website named **GatorTrader**. **GatorTrader** will allow the students, professors, and faculty of San Francisco State to easily buy, sell, and trade items. Some of **GatorTrader**'s features include: search by category, search by class, a messaging system, and a clean user interface.

Our team of six students is excited to provide an excellent product for our fellow Gators to use. Our ultimate goal is for **GatorTrader** to be an easy to use website that allows buyers and sellers to communicate with each other over a secure and reliable interface. In a time where the price of attending university is skyrocketing, we believe **GatorTrader** will make college life more affordable.

2. Personae and Main Use Cases:

---

1. David Miller

- Characteristics
  - Sophomore SFSU Student
  - History Major
  - Holds Part-time Job
- Goals
  - To become a professor.
- Skills
  - Competent with office applications and general computing.
  - Unfamiliar with programming.
- Paint Points
  - It is the end of the semester and he wishes to sell textbooks to someone in need of them.

---

2. Sarah Nelson

- Characteristics
  - Senior SFSU Student
  - Electrical Engineering Major
  - Full-time Student
- Goals
  - To pass courses with an A and maintain a high GPA.
- Skills
  - Competent in general computing.
  - Has experience with Python for general scripting purposes.
- Paint Points
  - Is in need of tutoring for Calculus III to help her prepare for an upcoming exam.

3. John Anec

- Characteristics
  - Senior SFSU Student
  - Undergrad Kinesiology Major
  - Part of the wrestling team
- Goals
  - Find somewhere to sell his workout equipment that is currently in the dorms before he graduates.
- Skills
  - Bodybuilder and knows about all the muscles in the body.
  - Loves to entertain guests
- Paint Points
  - Very little experience with computers.
  - Most internet usage has been on his cellphone. Finds some websites difficult to navigate.

4. Mike Jones

- Characteristics
  - Sophomore SFSU Student
  - Biomechanical Engineering Major
  - Works part time to pay for school
- Goals
  - Lowering the costs of school
- Skills
  - Proficient with computers and programming
  - Loves reading spy novels
- Paint Points
  - Not a lot of friends because of commuting and busy from studying and working.
  - Works to pay for school, struggles to keep up with the cost.

1. **Selling Books - David Miller**

Finals week is over for the Spring semester and David passed all of his courses without issue despite having a part-time job. He took GE courses for Introduction to Philosophy and World Literature and is looking to sell the textbooks for these courses to someone else to help pay for next semester's textbooks. Unfortunately, he does not have the time to go around asking if anyone will be taking these courses, so he goes to our website **GatorTrader** on his smartphone. From there he is immediately introduced to a few options including one to list his items for sale. Upon pressing, he is introduced with a form that allows him to describe his item and include a picture. He tags the first textbook with the course type and number (PHIL 101, Section 1), as well as ISBN and other identifying information to help others find his **Listing**. Once he presses the option to submit his **Listing**, he is prompted to register for an account. After making an account, he is brought to an overview page that lists his item for sale and he notices that it is tagged to be reviewed by an administrator and to check back later to verify his **Listing** was successful. He proceeds to follow the same process for his other textbook, except now owning an account he no longer needs to register again. David's items are now for sale and he saved time allowing him to move on to other matters.

2. **Scheduling Tutor - Sarah Nelson**

There is an exam coming up for Calculus III, and Sarah cannot make time in her schedule to visit office hours at the specified times given by her professor. Since she is free on Wednesday, she decides to look towards other methods to help her prepare; she preferably wants to find a tutor that can help her work through the problems step-by-step. Upon arrival at our website **GatorTrader**, she finds an option that allows her to refine her search to show tutors for Calculus III. The **Listings** that the tutors put up show available dates/times and possible locations, such as the library or the nearby Peet's Coffee. After finding a tutor to schedule time with, she clicks an option to **Message** them and is prompted to register for an account. After making an account, she is then prompted with a **Message** box addressed to the tutor with some pre-filled information based on her search criteria. She successfully sends her initial **Message** and she and the tutor go on to discuss her preferred payment method, as well as time and location. Sarah can now rest easy knowing she will get the help that she needs.

3. **Selling Equipment - John Anec**

John Anec is graduating in a few months and is moving across the country for a job he has lined up. Recently he has been working out in his dorm room as the gym is often filled with students using all the equipment, and in that time he has accumulated a lot of equipment. Rather than throwing it all away, he would like to find somewhere to sell it, preferably to someone that is close by and/or ideally someone else in the dorms. He isn't able to find any buyers from direct contact, but some friends recommend using the site **GatorTrader**. Even though he has limited experience with computers and websites, he is able to create an account on **GatorTrader** on his phone. Then finds the **List Items** category, selects the type of item he is selling, lists information about the item along with his contact information. With this he is able to quickly and efficient list his workout equipment for other students to purchase.

4. **Buying Books - Mike Jones**

Mike has been working to pay for school and is always low on funding. He needs a way to save money anywhere he can. Some of his biomechanical engineering books cost upwards of $200. He searches for a site where he can buy discounted books for his SFSU classes and finds **GatorTrader**, a site specifically for SFSU students and alumni. He is able to quickly find the search bar and enters in the books that he needs being sold at far lower prices by other students. He can see the **Listing** with the condition they are in, pictures, and the contact information. Mike is able to send a **Message** to the seller to get into contact with them.

3. List of main data items and entities – data glossary/description:

Data Items:

1. **User**
   - 1.1. Users of the website have a login and password that are stored on the server. Stores all current and previous listings that the user has made. Stores listings that the user has saved for later. Stores what type of privileges this account has, this could be admin, student, or faculty.

2. **Messages threads**
   - 2.1. Message threads are stored in tables with the two parties usernames along with a key to a message table where individual messages will be stored. This will allow us to easily get all the message threads for a particular user.

3. **Listing**
   - 3.1. This data item is used to store sellers listings, depending on the type of listing (textbook, furniture, housing, etc.) it has different data items that need to be stored. For example, selling a textbook would require entering the ISBN, and name of the class that it is used for. If selling furniture, the color, the type(sofa, bed, table, etc.) will be stored.

4. **Categories**
   - 4.1. Our Category table will group all the listings together based on the category. This will allow us to easily have all of the listings in one sorted place when we need to retrieve them.

4. Initial list of functional requirements:

1. **Users**
    - **1.1. Registered Users**
        - 1.1.1. Shall be able to delete their account
        - 1.1.2. Shall be able to edit account information
        - 1.1.3. Shall be able to post item/service **listings**
        - 1.1.4. Shall be able to delete item/service **listings**
        - 1.1.5. Shall be able to edit item/service **listings**
        - 1.1.6. Shall be able to post **listings** with a multiple photos
        - 1.1.7. Shall be able to tag **listings** with common attributes
        - 1.1.8. Shall be able to send messages to sellers/respond to buyers
        - 1.1.9. Shall login with a username and password
        - 1.1.10. Shall be able to report users/**listings** for suspicious/illegal activity
        - 1.1.11. Shall be able to view past messages for specific **listings**
    - **1.2. Any Users**
        - 1.2.1. Shall be able to use their SFSU email to create an account
        - 1.2.2. Shall be able to search for items by item name
        - 1.2.3. Shall be able to search for items by **category**
        - 1.2.4. Shall be able to search for items by class name
    - **1.3. Admins**
        - 1.3.1. Shall be able to review **listings**/edits/reports
        - 1.3.2. Shall be able to remove **listings** at any time
        - 1.3.3. Shall be able to provide feedback if a **listing** is rejected
        - 1.3.4. Shall be able to message registered users
        - 1.3.5. Shall be able to ban user account
        - 1.3.6. Shall be able to report illegal activities to the authorities

5. List of non-functional requirements:

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO).
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers
3. All or selected application functions must render well on mobile devices
4. Data shall be stored in the database on the team's deployment server.
5. No more than 50 concurrent users shall be accessing the application at any time
6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
7. The language used shall be English (no localization needed)
8. Application shall be very easy to use and intuitive
9. Application should follow established architecture patterns
10. Application code and its repository shall be easy to inspect and maintain
11. Google analytics shall be used
12. No e-mail clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application
13. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.
14. Site security: basic best practices shall be applied (as covered in the class) for main data items
15. Media formats shall be standard as used in the market today
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
17. The application UI (WWW and mobile) shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2020. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application).

6. Competitive analysis:

Features	GatorTrader	Amazon	eBay	CraigsList
Search By Category	++	++	++	++
User-friendly UI	++	++	++	+
Returns	+	++	+	-
Price Negotiation	+	-	+	+
Reselling	+	+	+	+
Designated Pick-up	+	-	-	-
SFSU Class Search	++	-	-	-

++ = excellent + = good - = unavailable

GatorTrader will combine the best aspects of other e-commerce websites (Amazon, eBay, CraigsList) while offering exclusivity to the SFSU campus. **GatorTrader** will be different from other websites in that it allows users to search for items pertaining specifically to a SFSU class. This will allow users to quickly find needed textbooks and supplies to prepare for the upcoming semester by using class ID. Additionally, since **GatorTrader** is specific to SFSU, users will be ensured that pickup spots will always be within a reasonable distance. All of this will all be available on a clean and intuitive user-interface free from advertisements and clutter that are often found on other websites.

7. High-level system architecture and technologies used:

Software stack:

Server Host: Amazon AWS 1vCPU 1 GB RAM
Operating System: Ubuntu 18.04.5 LTS
Database: MySQL 14.14 Distrib 5.7.31
Web Server: Uvicorn 0.11.8
Server-Side Language: Python 3.6.9

Additional Technologies:

Frontend Framework: Bootstrap, React
Web Framework: FastAPI (by proxy, Starlette)
IDE: JetBrains PyCharm, IntelliJ
Web Analytics: Amazon Kinesis Data Analytics
SSL Certificate: Let's Encrypt (Cert Bot)
SASS: 3.5.6

Supported Browsers:

Firefox 81.0
Google Chrome 85.0.4183.121

8. Team and roles:

- Niall Healy - Team Lead, Document Master
- Aaron Lander - Github Master
- Dale Armstrong - Front End Lead
- Lukas Pettersson - Back End Lead
- Joseph Babel - Team Member Back End
- Vern Saeteurn - Team Member Front End

9. Checklist:

DONE So far all team members are engaged and attending ZOOM sessions when required

DONE Team found a time slot to meet outside of the class - 2 PM Fridays

DONE Back end, Front end Leads, and Github master chosen

DONE Team decided and agreed together on using the listed SW tools and deployment server

DONE Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing

ON TRACK Team lead ensured that all team members read the final M1 and agree/understand it before submission

DONE Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.)

M2:

SW Engineering CSC 648/848 Fall 2020
Team 4

Milestone 2
GatorTrader

Niall Healy (Team Lead, nhealy@mail.sfsu.edu)
Aaron Lander
Dale Armstrong
Lukas Pettersson
Joseph Babel
Vern Saeteurn

Date submitted	Date(s) Revised
October 15, 2020	

Table of Contents:

1. Executive Summary:

As new students who are proficient in using technology enroll at San Francisco State University, a need for a platform that provides the ability to buy, sell, and trade essential items increases. Buying textbooks straight from a vendor is expensive, and finding good deals can be challenging. Our company would like to develop a free-to-use website named **GatorTrader** to solve this issue. **GatorTrader** will allow the students, professors, and faculty of San Francisco State to easily buy, sell, and trade items.

Since many students attend the same classes at San Francisco State University, and classes usually use the same textbooks and supplies every semester, these goods should be able to be passed down from student to student each semester. This does not happen, however, because there is no communication medium in place to facilitate these transactions. **GatorTrader** will serve as this communication medium and provide excellent features, including: search by category, search by class, a messaging system, and a clean user interface.

Our team of six students is excited to provide an excellent product for our fellow Gators to use. Our ultimate goal is for **GatorTrader** to be an easy to use website that allows buyers and sellers to communicate with each other over a secure and reliable interface. In a time where the price of attending university is skyrocketing, we believe **GatorTrader** will make college life more affordable.

2. List of main data items and sub-items:

1. **User**
   - 1.1. **Definition:** Users of the website have a login and password that are stored on the server. Stores all current and previous listings that the user has made. Stores listings that the user has saved for later. Stores what type of privileges this account has, this could be admin, student, or faculty.
   - 1.2. **Sub-items:**
     - 1.2.1. **username:** Username used to login to site
     - 1.2.2. **userID:** Unique user ID used for database
     - 1.2.3. **phone:** The user's phone number
     - 1.2.4. **email:** The user's SFSU email address
     - 1.2.5. **password:** The password that the user uses to sign in
     - 1.2.6. **listings:** List of listings a user has posted
     - 1.2.7. **messageThreads:** List of message thread IDs the user is a part of
     - 1.2.8. **isAdmin:** Flag used to delineate admin accounts from other users

2. **Listing**
   - 2.1. **Definition:** This data item is used to store sellers listings, depending on the type of listing (textbook, furniture, housing, etc.) it has different data items that need to be stored. For example, selling a textbook would require entering the ISBN, and name of the class that it is used for. If selling furniture, the color, the type(sofa, bed, table, etc.) will be stored.
   - 2.2. **Sub-items:**
     - 2.2.1. **name:** The name of the listing; displayed on search results and listing pages
     - 2.2.2. **listingID:** Unique listing ID used for database
     - 2.2.3. **sellerID:** userID of the user who posted the listing
     - 2.2.4. **timestamp:** Date and time the listing was posted
     - 2.2.5. **description:** The description of the listing; displayed on the listing page
     - 2.2.6. **price:** The price that the seller set for the listing
     - 2.2.7. **categories:** List of categories that the listing applies to
     - 2.2.8. **photos:**
     - 2.2.9. **isApproved:** Flag used to show whether the post has been approved by an admin
     - 2.2.10. **isActive:** A flag that represents whether or not a Listing can show up in the search query

3. **Message thread**

3.1. **Definition:** Message threads are stored in tables with the two parties' ID's along with a key to a message table where individual messages will be stored. This will allow us to easily get all the message threads for a particular user.

3.2. **Sub-items:**

3.2.1. **threadID:** Unique message thread ID used for database

3.2.2. **messages:** List of messages that the thread consists of

3.2.3. **sellerID:** userID of the seller

3.2.4. **buyerID:** userID of the buyer

4. Message

4.1. **Definition:**

4.2. **Sub-items:**

4.2.1. **messageContents:**

4.2.2. **messageID:** Unique message ID used for database

4.2.3. **threadID:** threadID of the message thread that the message belongs to

4.2.4. **senderID:** userID of the user who sent the message

4.2.5. **timestamp:** Date and time the message was sent

5. Category

5.1. **Definition:** Table that holds Listing IDs sorted by category

5.2. **Sub-Items:**

5.2.1. **books:** List of Listing IDs that are part of the books category

5.2.2. **housing:** List of Listing IDs that are part of the housing category

5.2.3. **services:** List of Listing IDs that are part of the services category

5.2.4. **household:** List of Listing IDs that are part of the household category

5.2.5. **electronics:** List of Listing IDs that are part of the electronics category

5.2.6. **automotive:** List of Listing IDs that are part of the automotive category

5.2.7. **games:** List of Listing IDs that are part of the games category

5.2.8. **beauty:** List of Listing IDs that are part of the beauty category

5.2.9. **outdoors:** List of Listing IDs that are part of the outdoors category

3. Prioritized Functional Requirements:

a. **Priority 1:**

1.1. Registered Users

1.1.3. Shall be able to post item/service **listings**

1.1.8. Shall be able to send **messages** to sellers/respond to buyers

1.1.9. Shall login with a username and password

1.2. Any Users

1.2.1. Shall be able to use their SFSU email to create an account

1.2.2. Shall be able to search for items by item name

1.3. Admins

1.3.2. Shall be able to remove **listings** at any time

1.3.5. Shall be able to ban **user** account

b. **Priority 2:**

1.1. Registered Users

1.1.1. Shall be able to delete their account

1.1.2. Shall be able to edit account information

1.1.4. Shall be able to delete item/service **listings**

1.1.5. Shall be able to edit item/service **listings**

1.1.10. Shall be able to report users/**listings** for suspicious/illegal activity

1.2. Any Users

1.2.3. Shall be able to search for items by **category**

1.2.4 Shall be able to search for items by class name

1.3. Admins

1.3.1. Shall be able to review **listings**/edits/reports

1.3.4. Shall be able to **message** registered users

1.3.6. Shall be able to report illegal activities to the authorities

c. **Priority 3:**

1.1. Registered Users

1.1.6. Shall be able to post **listings** with a multiple photos

1.1.7. Shall be able to tag **listings** with common attributes

1.1.11. Shall be able to view past **messages** for specific **listings**

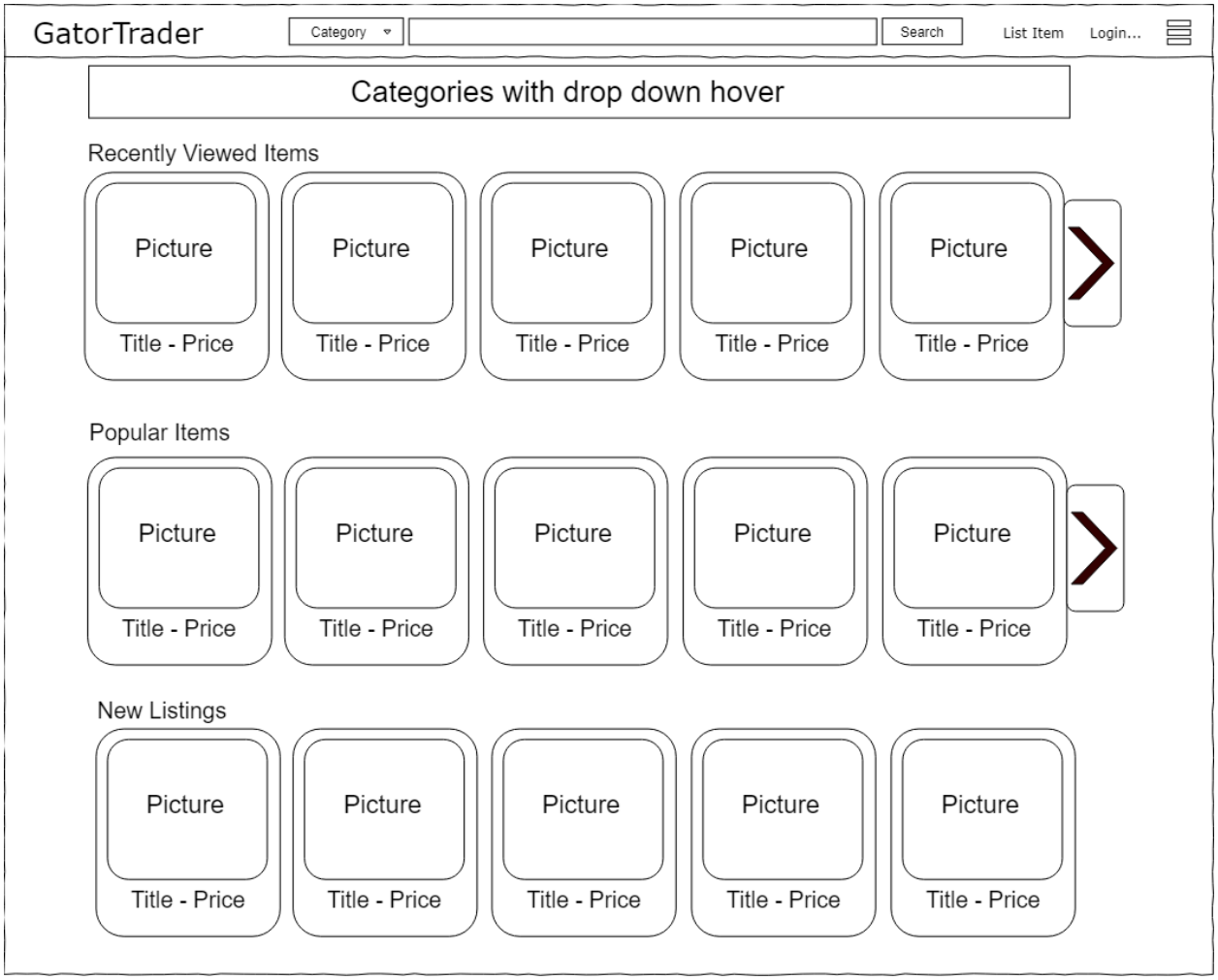
1.3. Admins

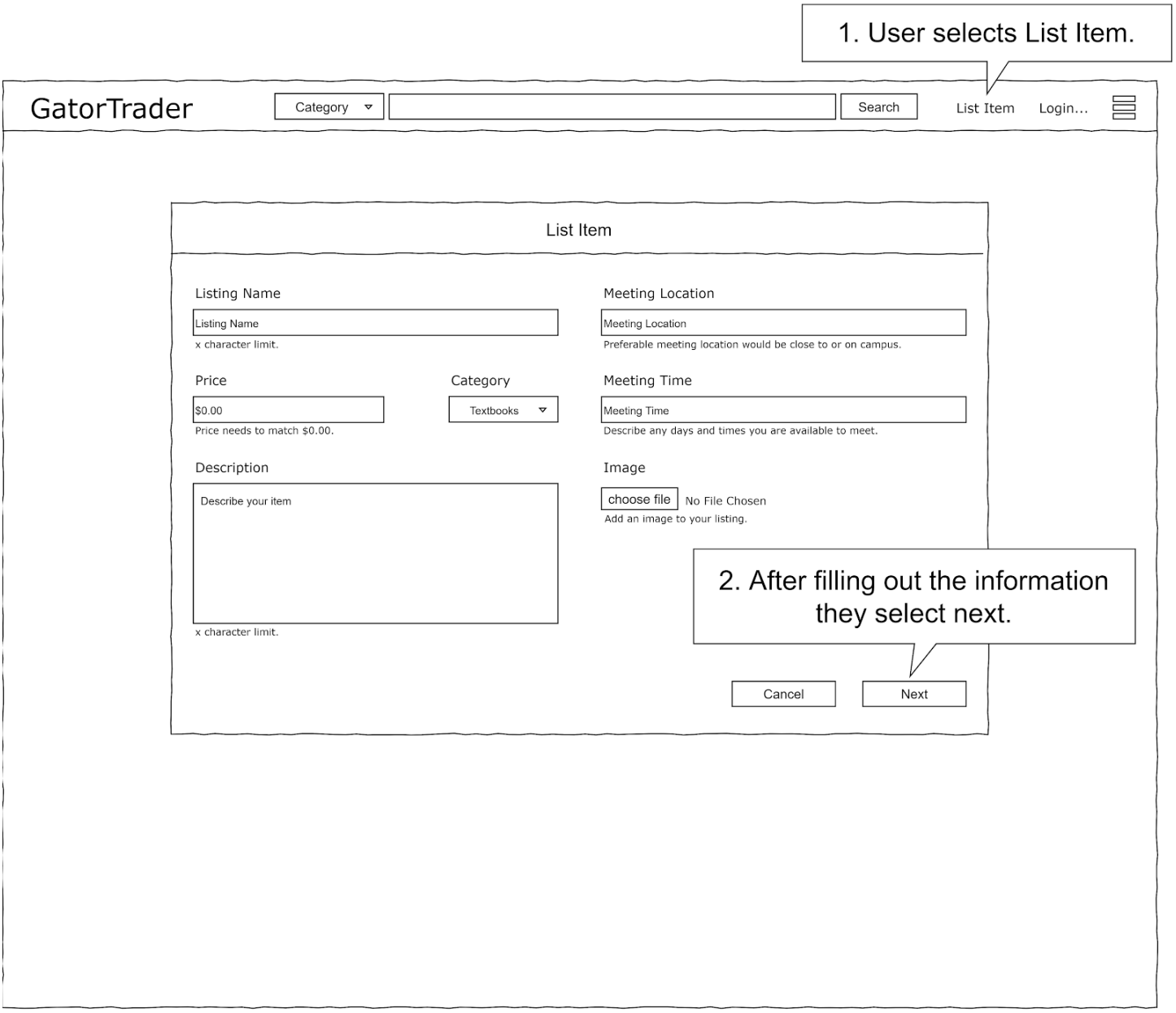
1.3.3. Shall be able to provide feedback if a **listing** is rejected

4. UI Mockups and Storyboards:

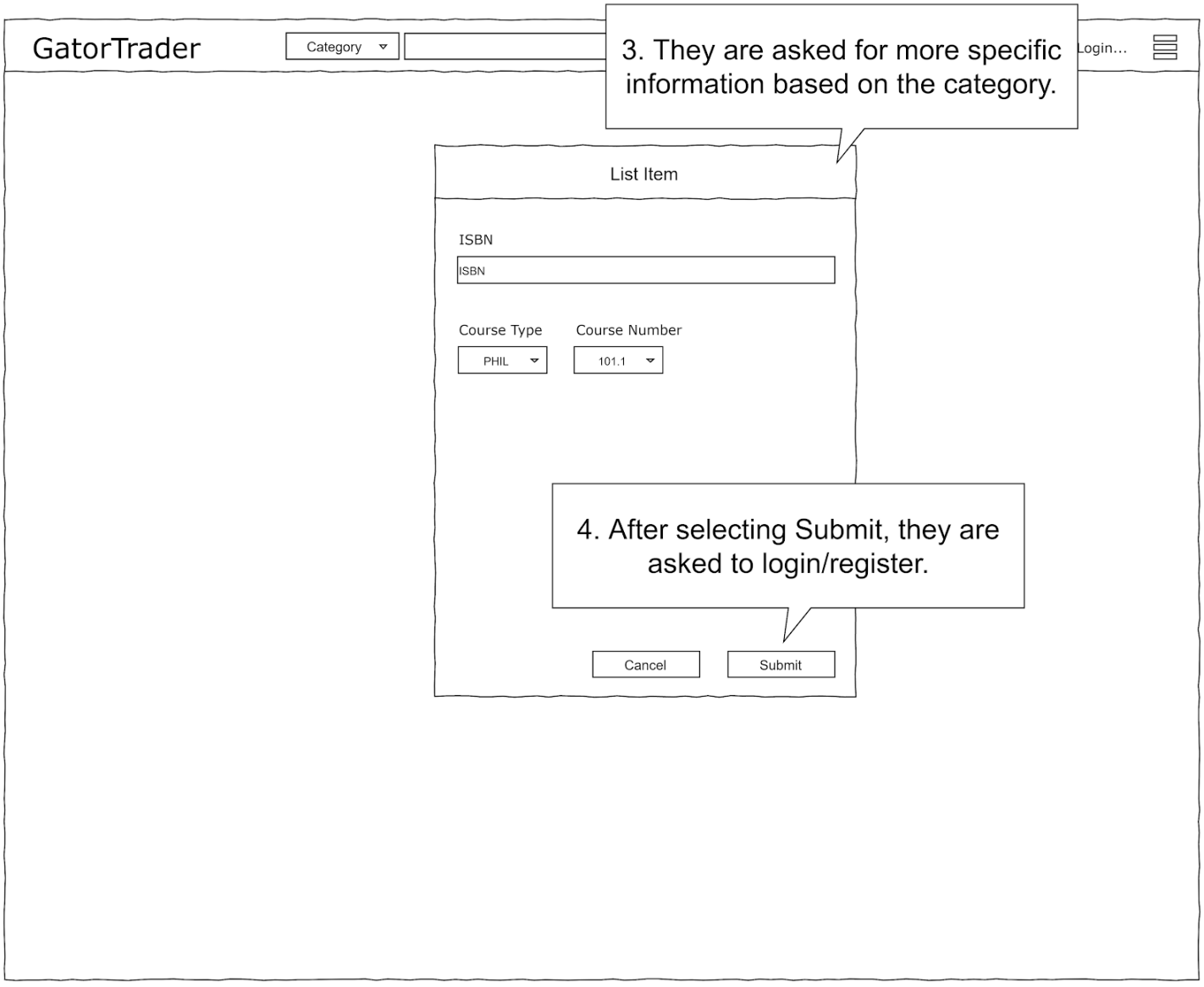
Use Case 1: Selling Books - David Miller

David wants to list his books on the site. He selects ‘List Item’. Visits the site.

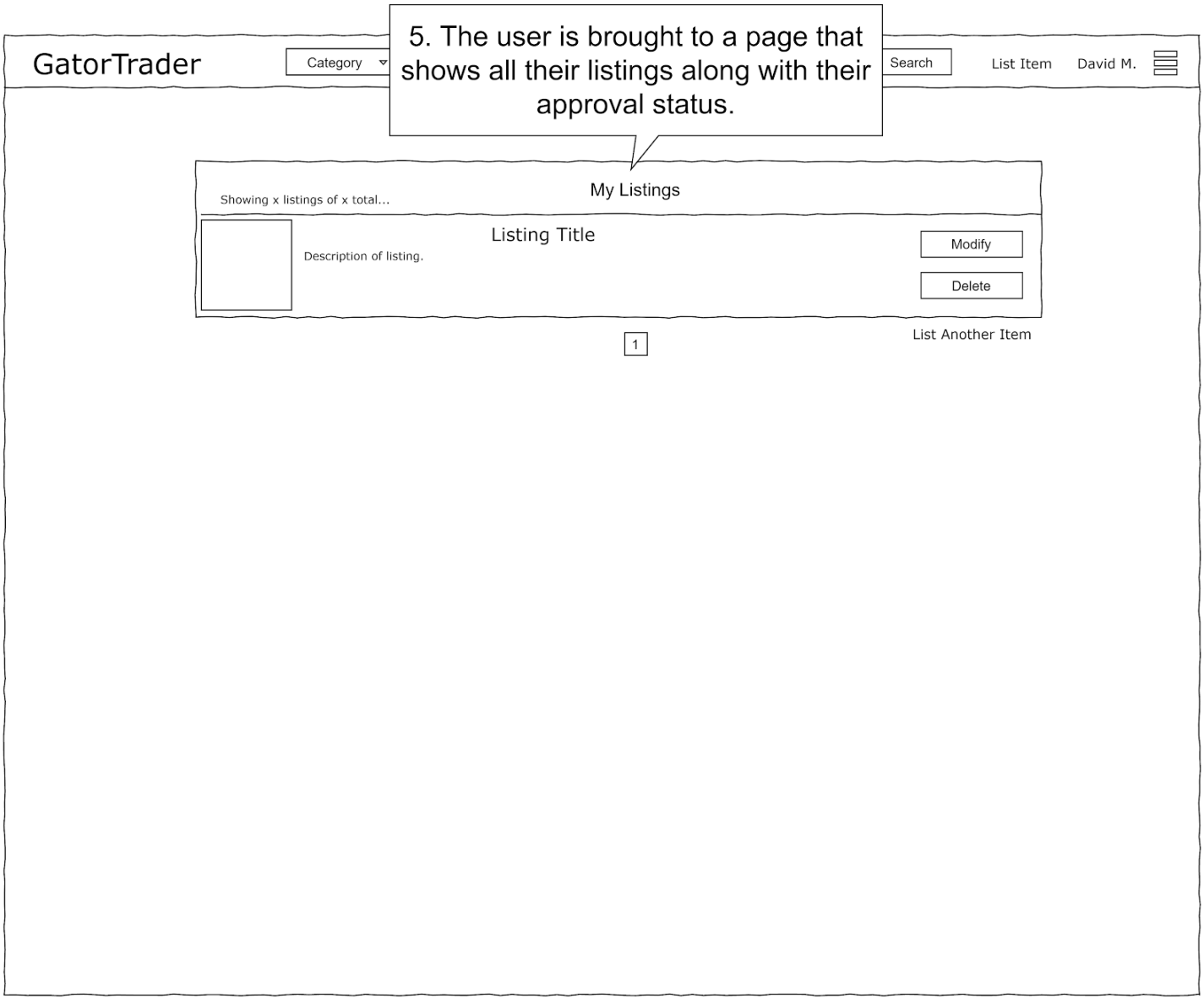



1. User selects List Item.
GatorTrader
Category
Search
List Item
Login...
List Item
Listing Name
Listing Name
x character limit.
Meeting Location
Meeting Location
Preferable meeting location would be close to or on campus.
Price
$0.00
Price needs to match $0.00.
Category
Textbooks
Meeting Time
Meeting Time
Describe any days and times you are available to meet.
Description
Describe your item
x character limit.
Image
choose file
No File Chosen
Add an image to your listing.
2. After filling out the information they select next.
Cancel
Next

After filling out the prior information the next page asks for more specific information based on the category.

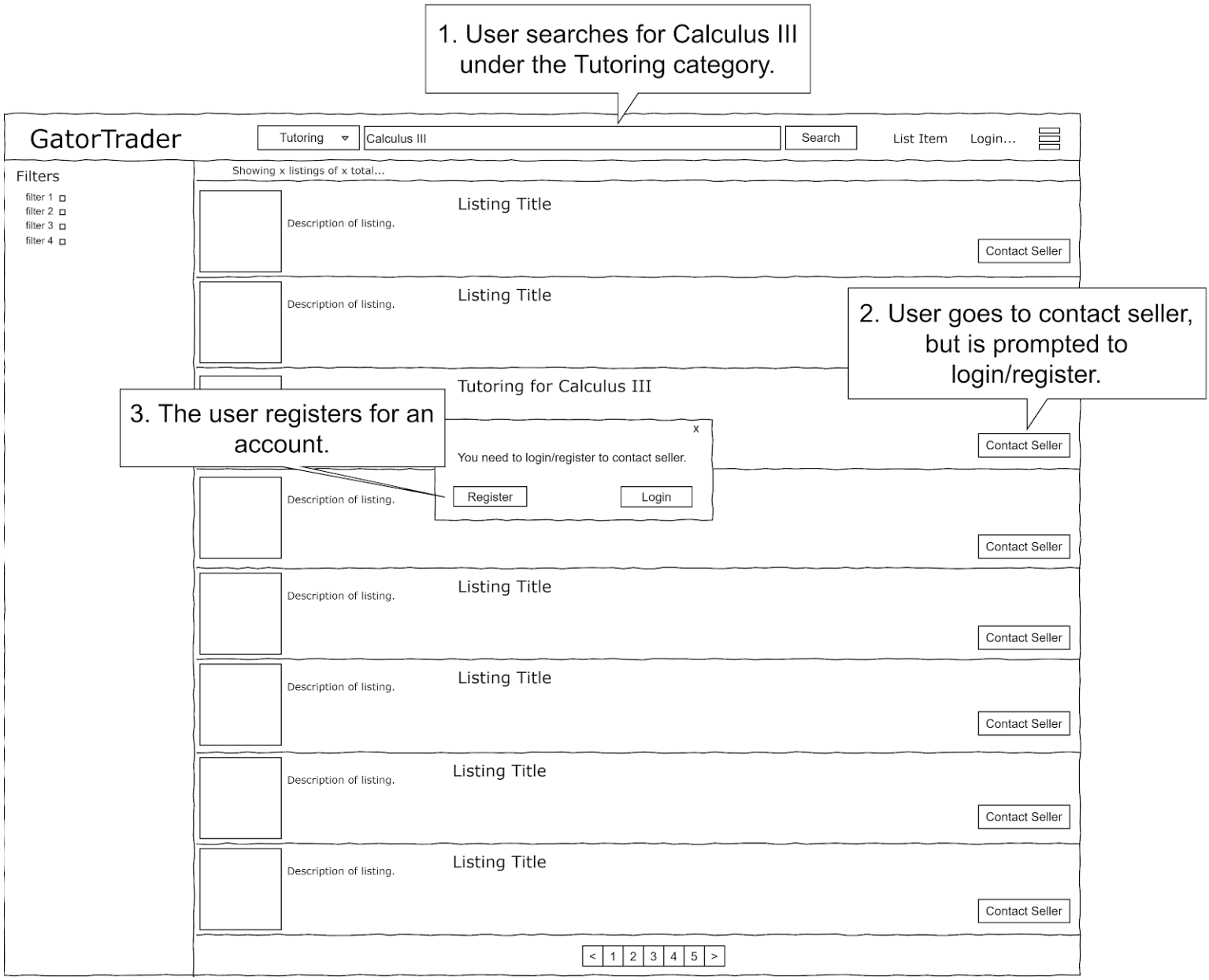


After registering for an account David is brought to the ‘My Listings’ page, which shows the status of approval for his listing.

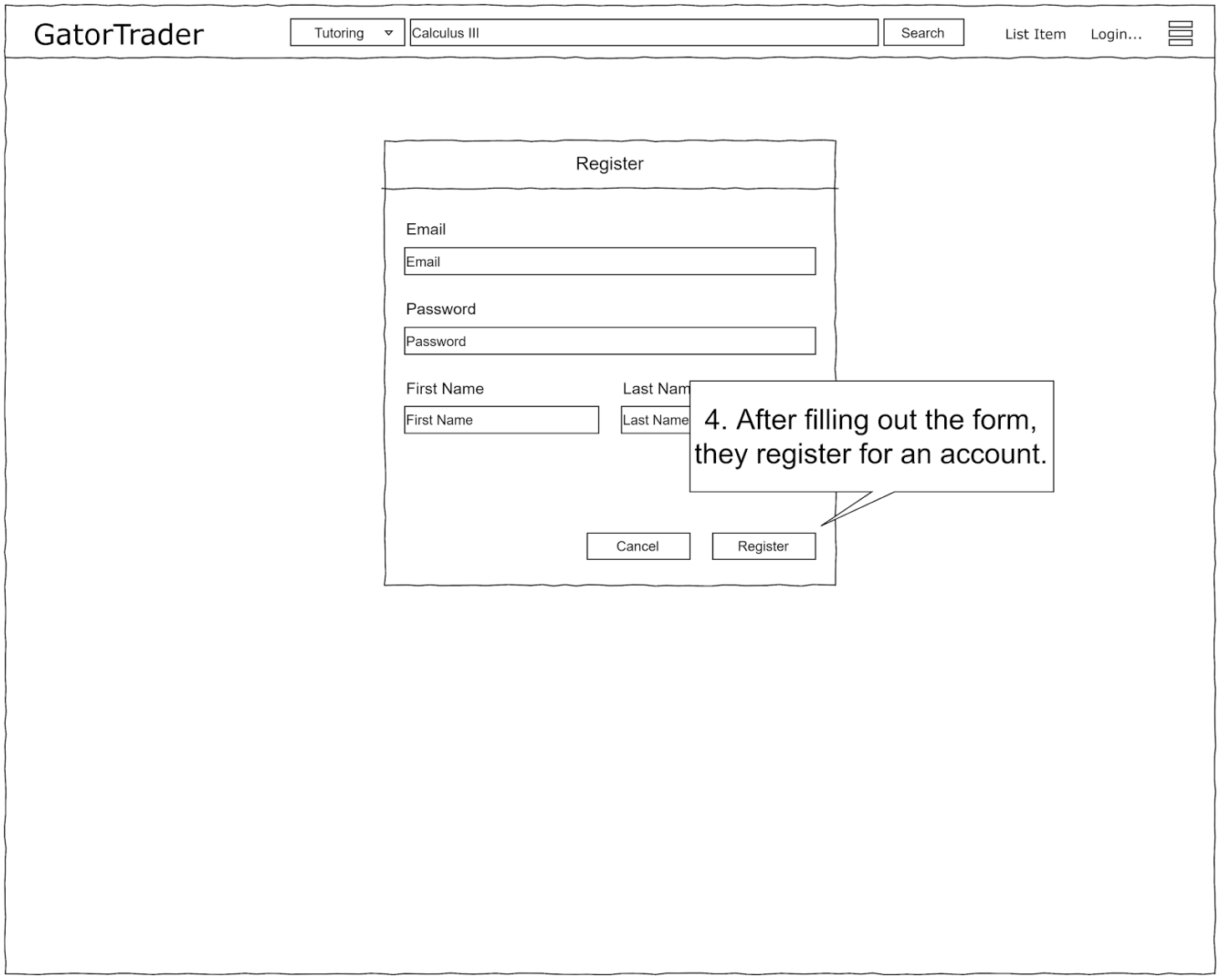


Use Case 2: Scheduling Tutor - Sarah Nelson

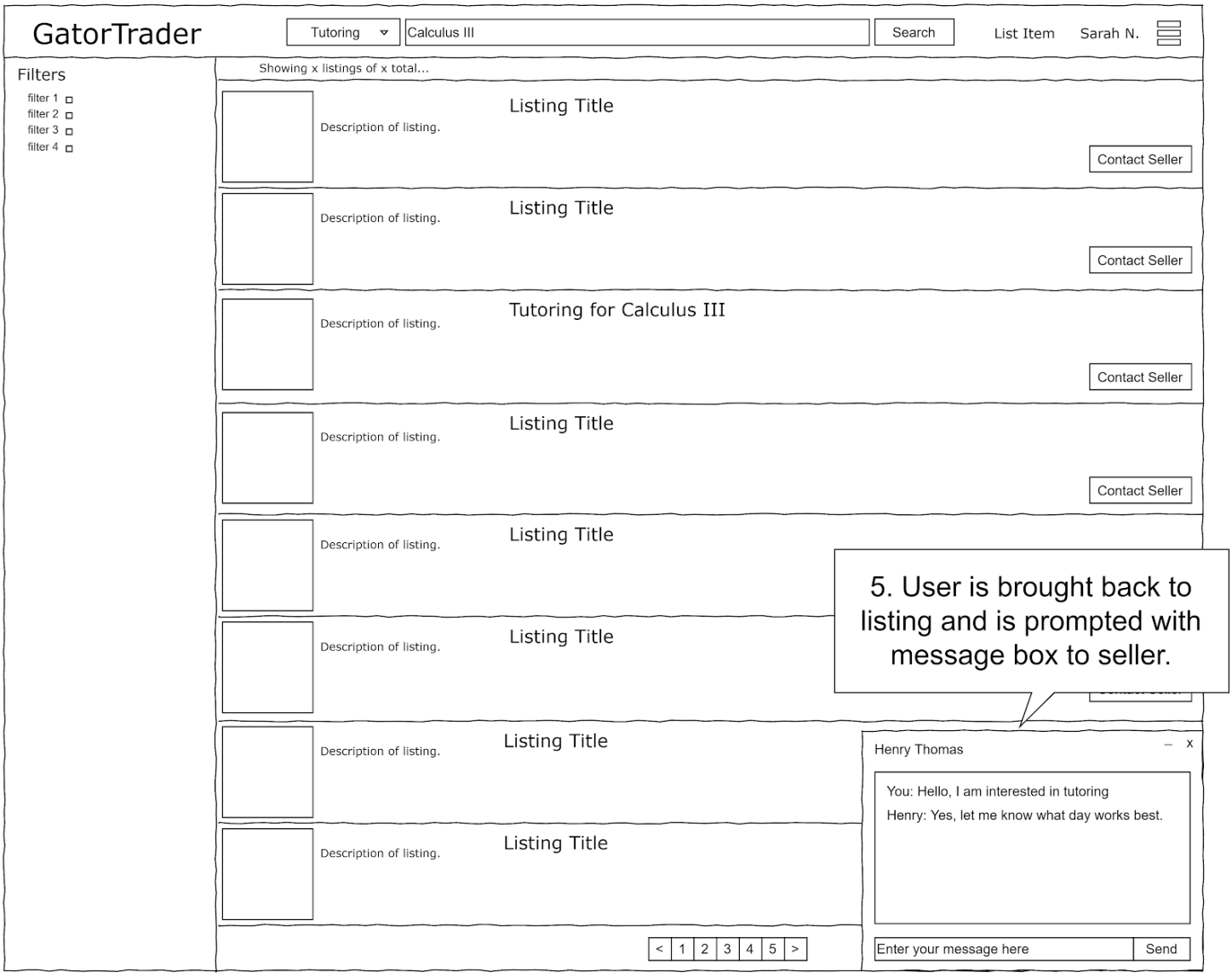
Sarah is looking for a tutor for Calculus III. She enters the appropriate search information and sets the category to tutoring.



She is prompted to make an account.

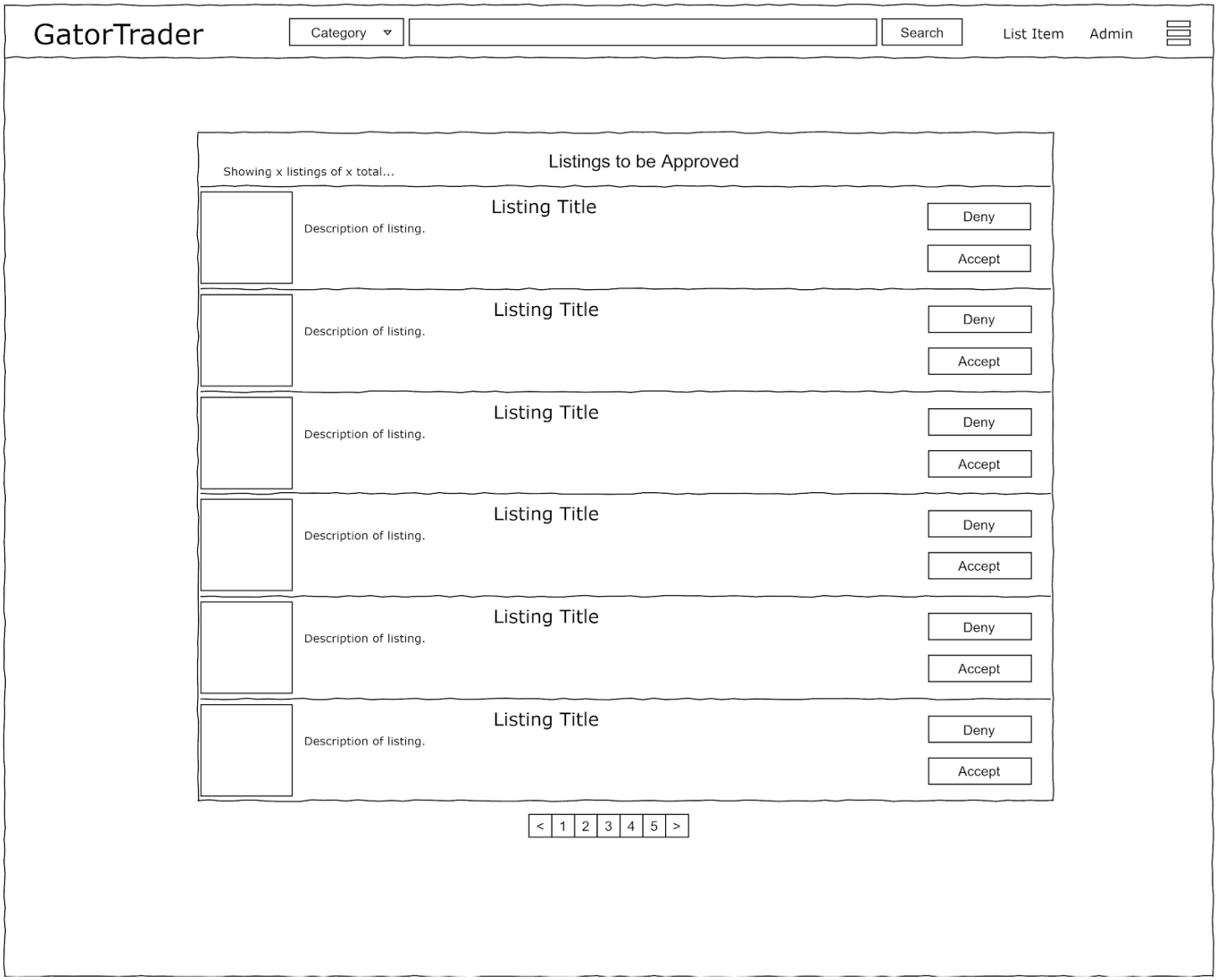


After making an account she is prompted with a message box addressed to the tutor.



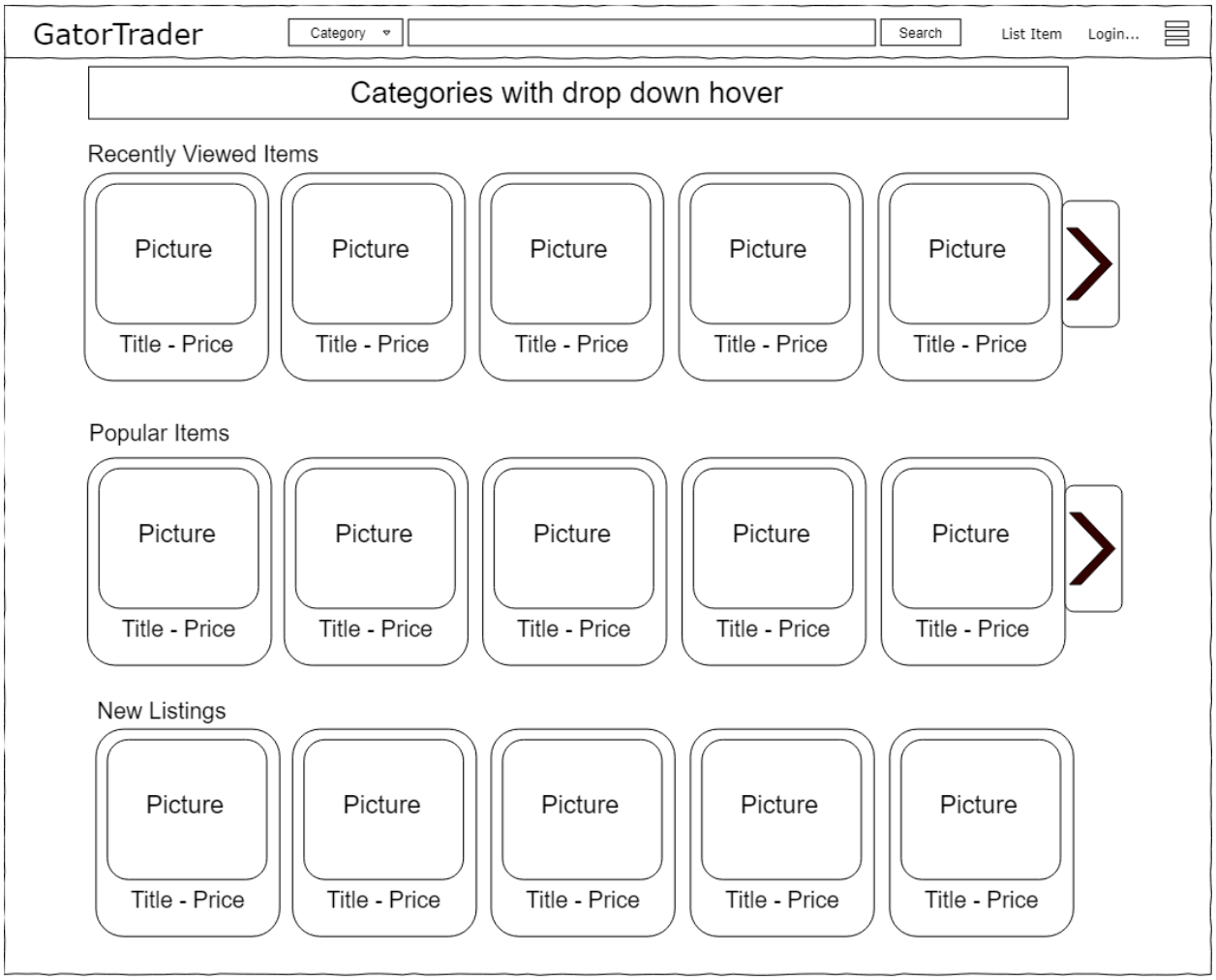
Approval of Listings - Admin

Page that allows admin to approve or deny listings.

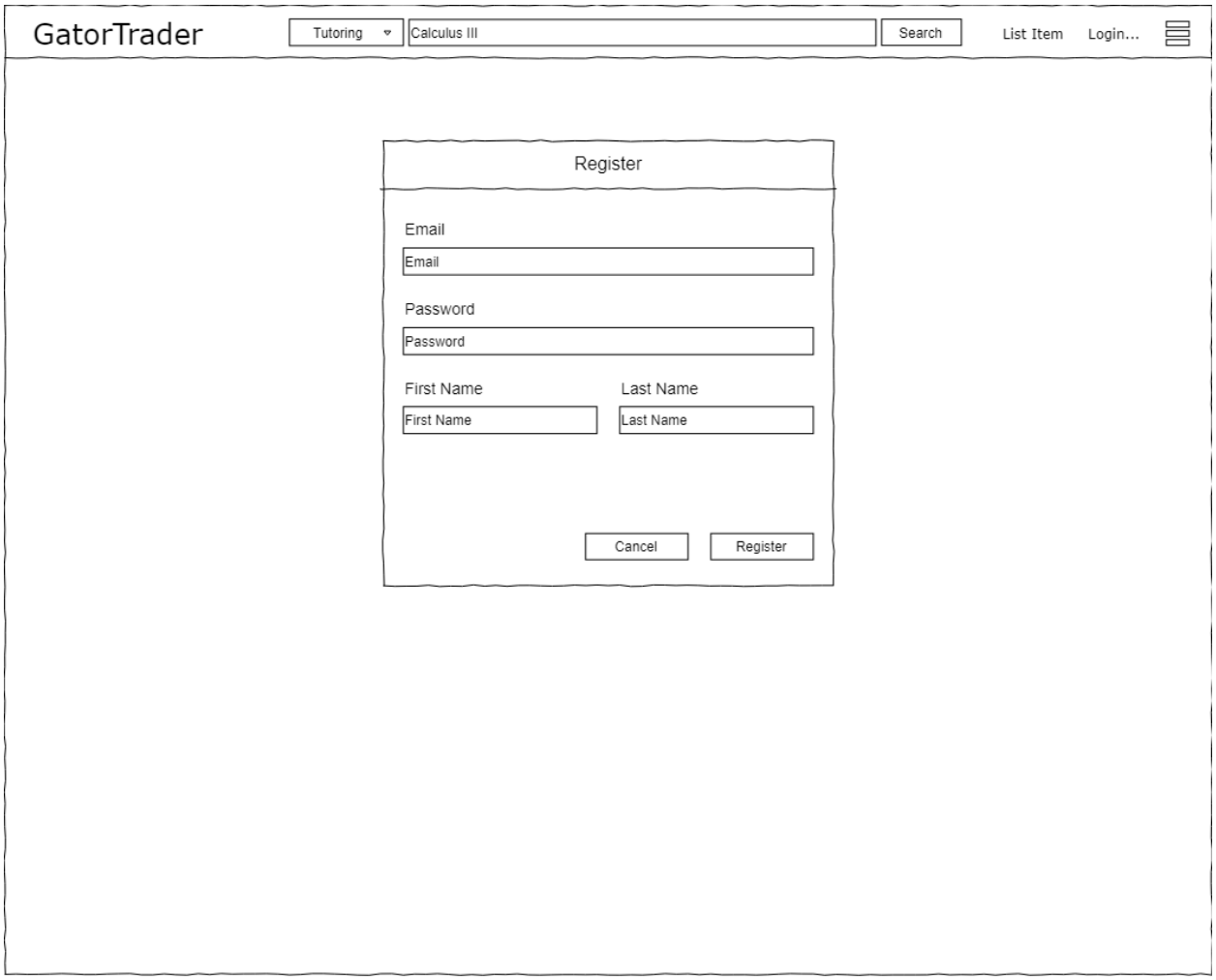


Use Case 3: Selling Equipment - John Anec

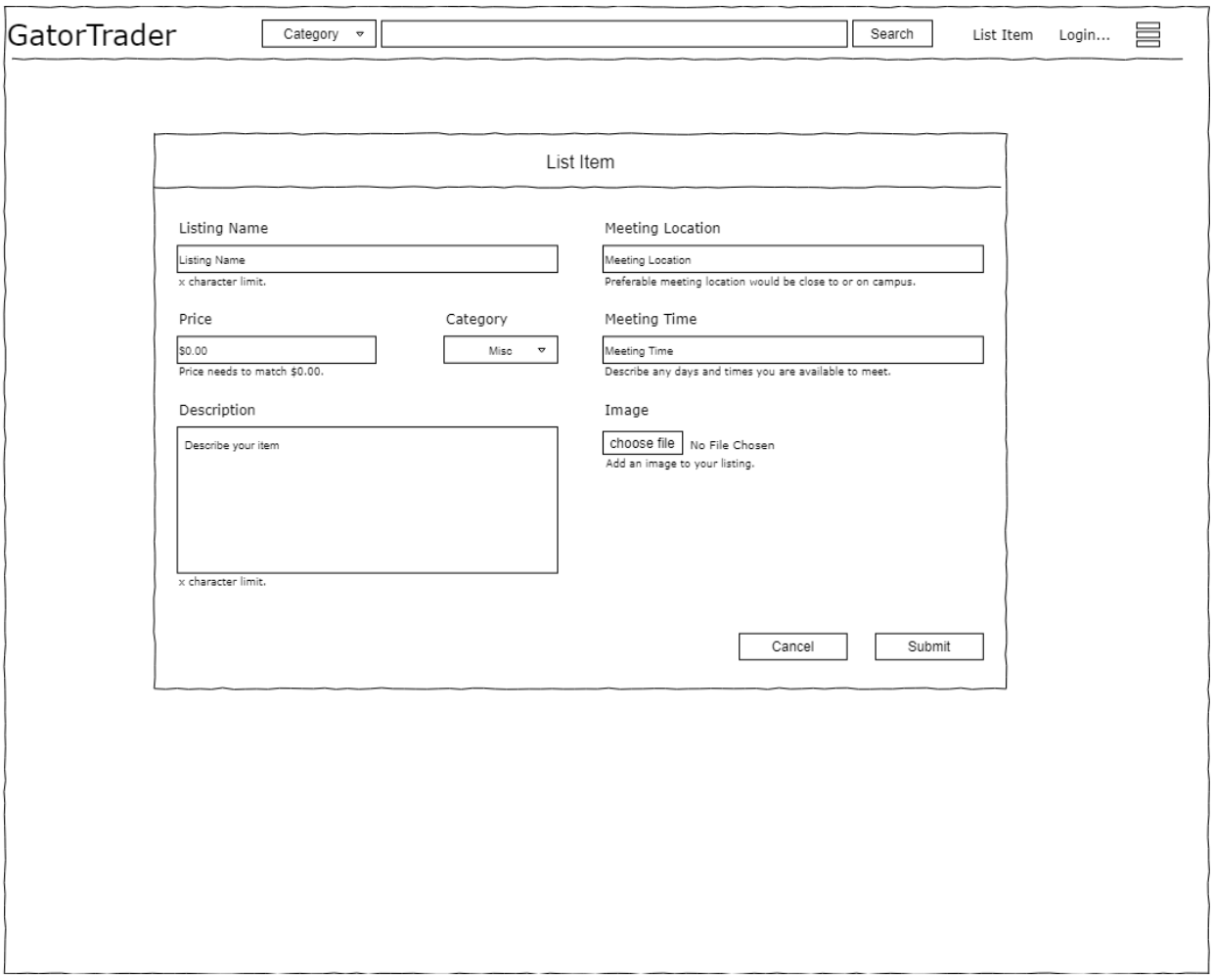
John visits the website.



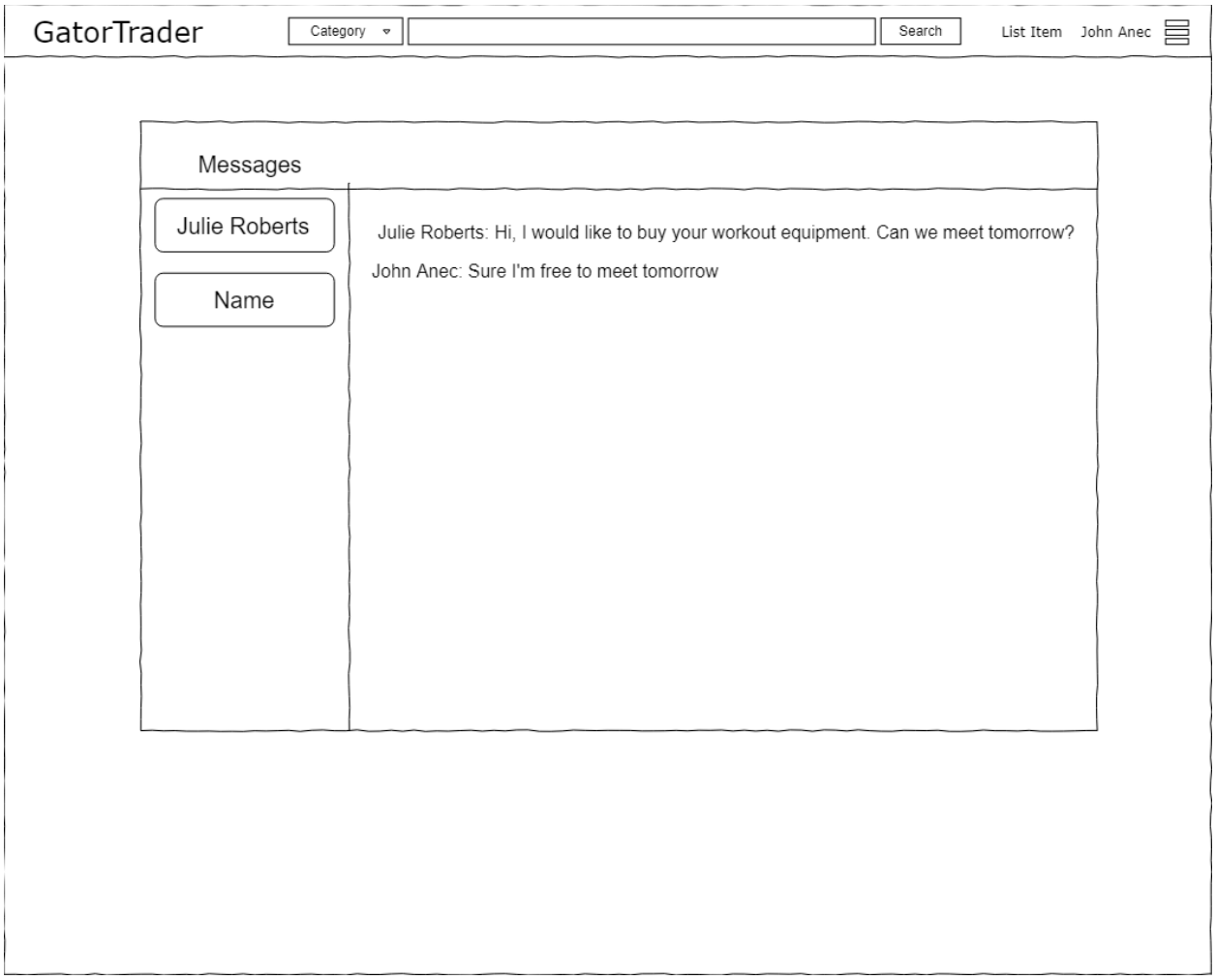
He then signs up for an account



He is now able to list his weight equipment. He selects misc for category. Clicks submit and waits for approval.

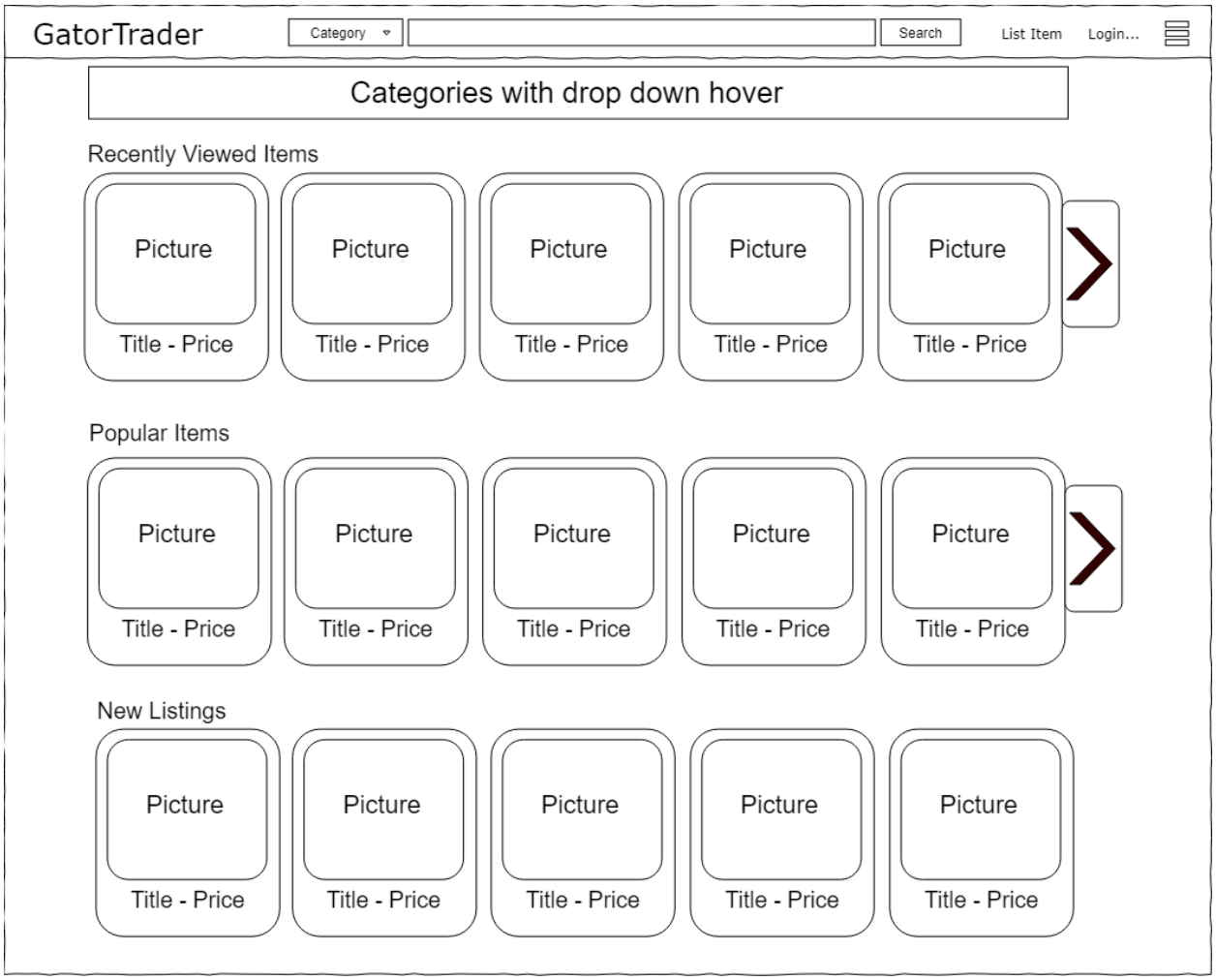


Once approved, he checks his messages and is able to communicate with a buyer.

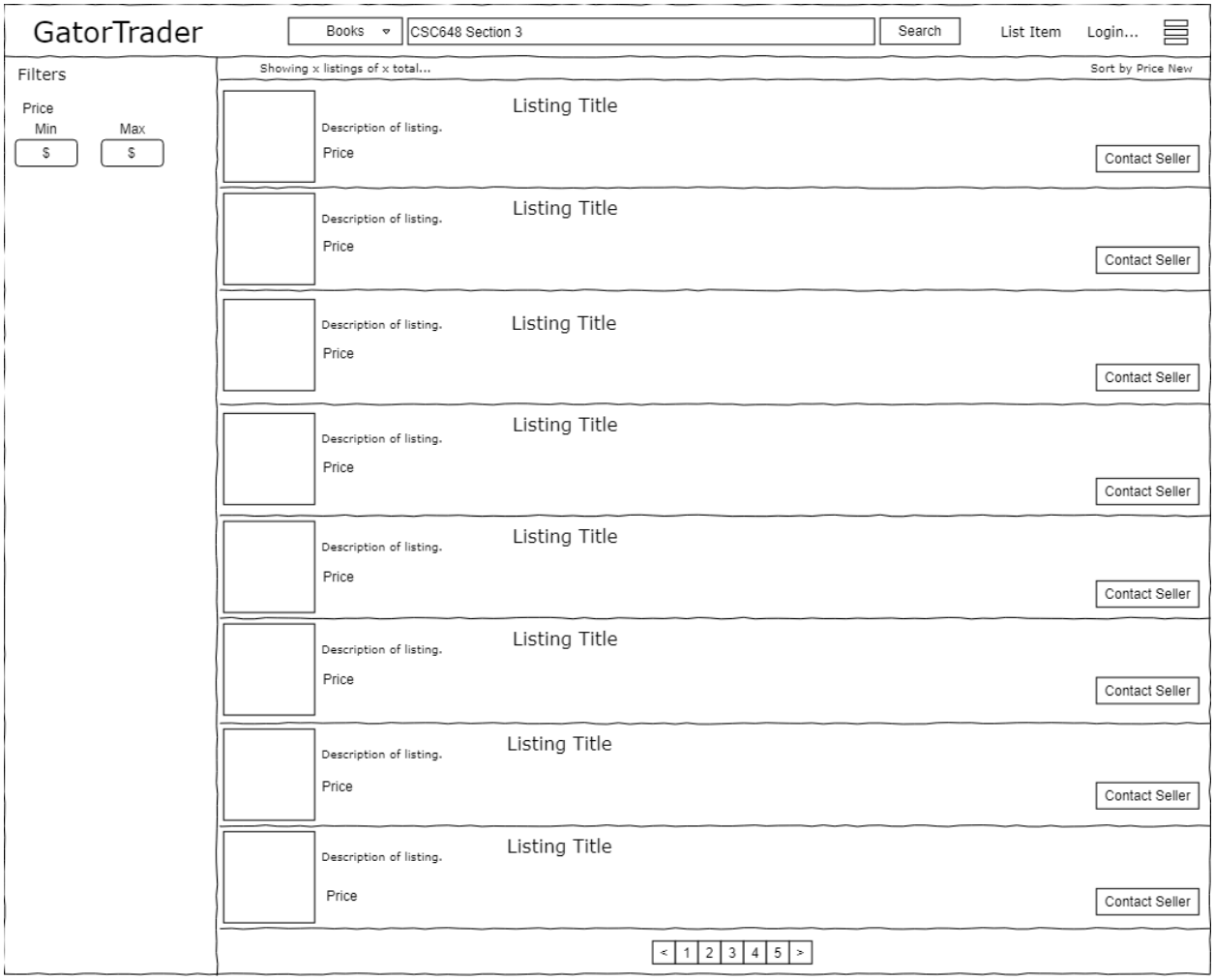


Use Case 4: Buying Books - Mike Jones

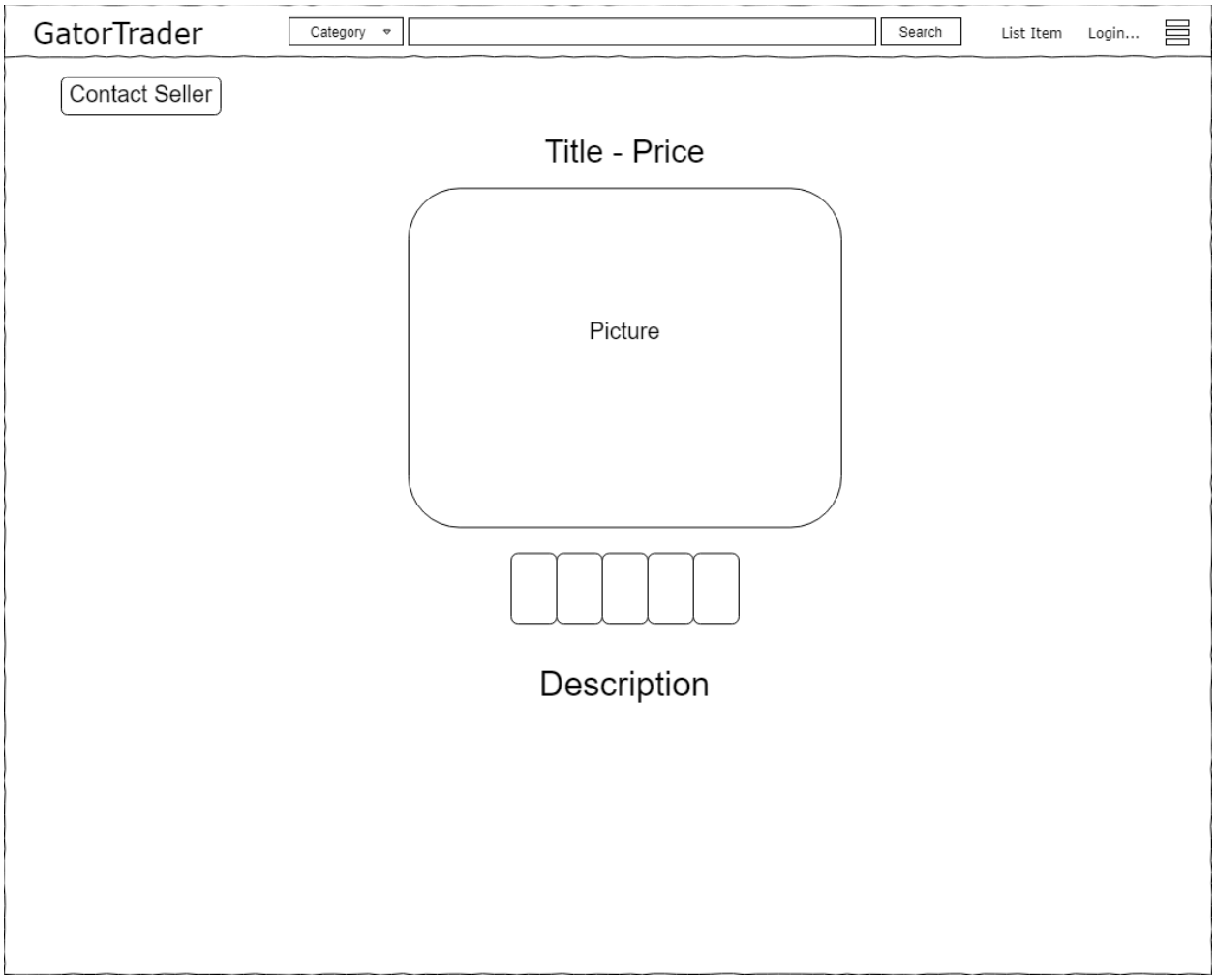
Mike signs onto the website.



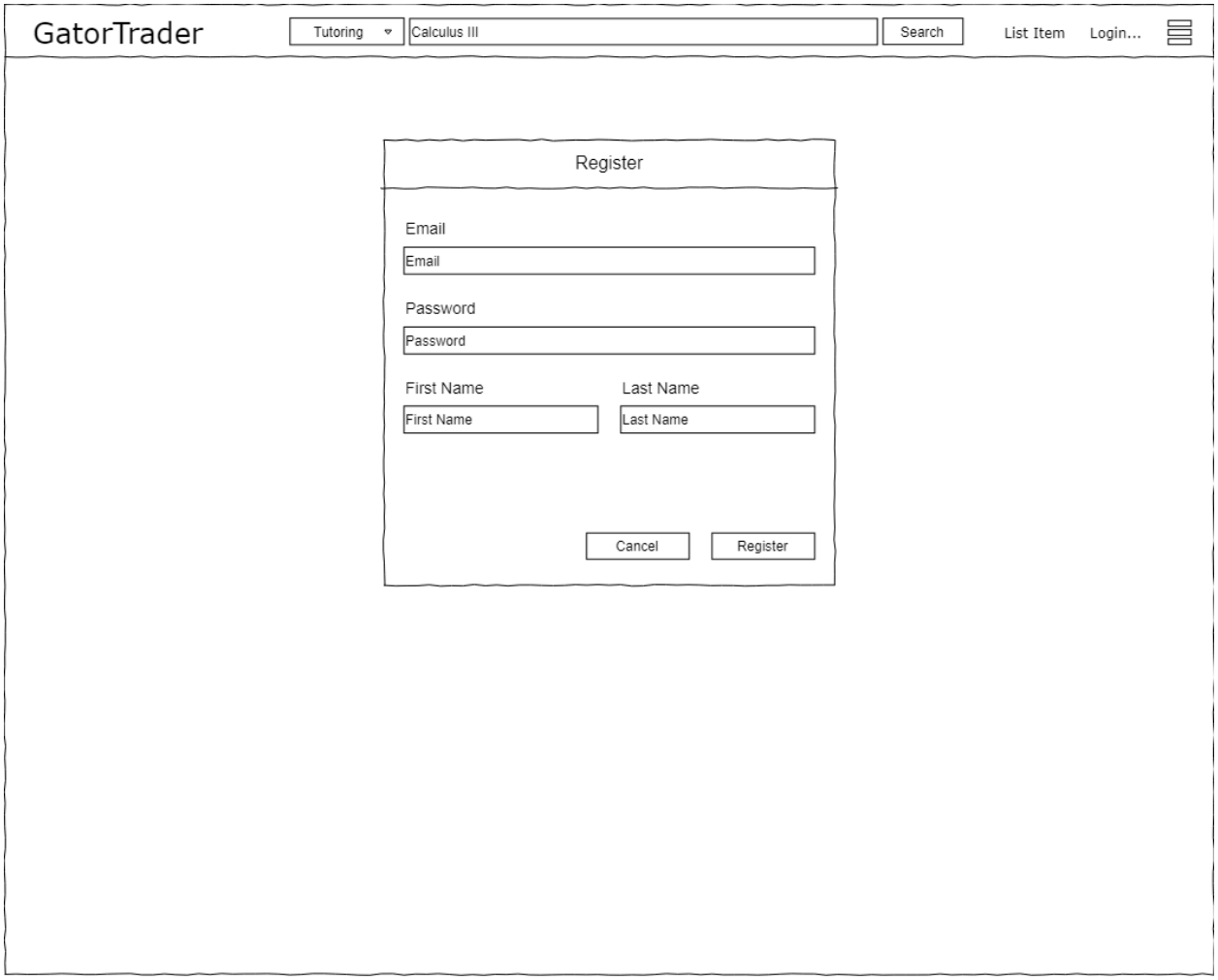
He then changes the category in the drop down next to the search bar and selects books. He proceeds to enter the first book that he needs for his classes and clicks search. He is able to sort by Price.



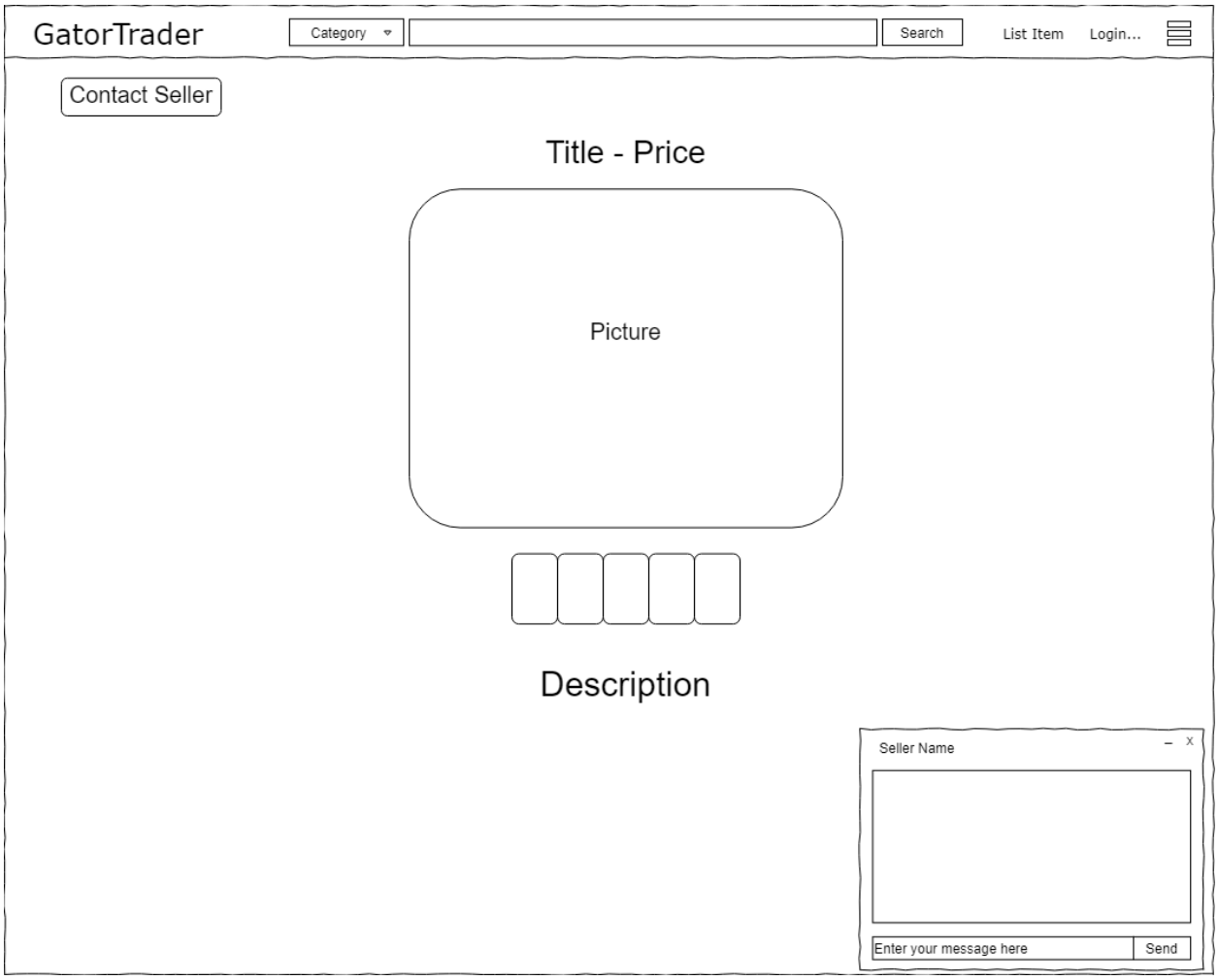
He finds a suitable book for a good price and clicks on the listing.



He reads the description and takes a look at the available pictures. Then decides to contact the seller. Since he does not have an account, he has to register first.



Once he is done registering, he is able to contact the seller.



5. High Level Architecture, Database Organization Summary:

The backend for our project consists of main tables to organize the main data items, as well as sub tables that are used for holding data of specific categories of items. We are using IDs for much of our database to relate different tables to one another. For example we use the userID to tie users to Listings as well as message Threads as well as individual messages. We are also using ListingIDs to relate it to our sub tables as well as the Categories table.

Main Tables:

User
username
userID
phone
email
password
listings
messageThreads
isAdmin

Categories
books
housing
services
household
electronics
automotive
games
beauty
outdoors

Listing
name
listingID
sellerID
timestamp
description
categories
photos
isApproved
isActive

Message
messageContents
messageID
threadID
senderID
timestamp

Message thread
threadID
messages
sellerID
buyerID

Sub Tables:

books
listingID
relevantClass

housing
listingID
streetAddress

automotive
listingID
year
make
model
odometer
title status
fuel

6. Key Risks:

Skill Risks:

- Some technical issues that can arise may be difficult for team members to troubleshoot in a timely manner.
    - Solution: Have good documentation of the project and well-managed version control to recover possible errors. Learn how to properly use relevant tools.
- Many of our members are learning the skills we use during the project for the first time. We may not have enough time to both learn the tools and use them in our project.
    - Solution: Distribute the workflow on the sections needed.

Scheduling Risks:

- Collaborating with a group may be difficult in terms of finding a meeting time that works with every team member. The risks include scheduling a regular meeting time, having every member participate in those meetings, and not meeting deadlines due to confusions about work distribution.
    - Solution: Use applications such as Discord to schedule meetings and to keep in touch with everyone. Team members can help their team catch up with anything that was missed in case they were not able to attend meetings.

Technical Risks:

- Issues can arise that may be out of team members' hands, such as their internet going out or malfunctioning hardware/software.
    - Solution: Have a back up plan in place. For example, if there is a problem with one's internet connection, have Zoom/Discord available on mobile so that they can still attend meetings.

Teamwork Risks:

- It can be difficult to track how each team member is doing on their part of the project without constantly checking up on them, which may hinder team chemistry and trust.

    - Solution: Set up regular checkpoints (weekly, bi-weekly, etc.) for the team to quickly chat or meet up to see how each team member is doing on their individual tasks.

Legal/Content Risks:

- Using software and utilities without proper licensing will pose a legal risk to the project and to the team members.
    - Solution: Make sure team members have the proper licensing and permission for all software and tools used for the project.

7. Project Management:

For project management we have decided to use Trello. It's free and has a convenient and user friendly GUI that makes it simple to assign tasks and track how everyone in the group is doing. Many of the default settings work perfectly for our project, and it makes you think about what a great job Trello did developing their web application. We have it set up so that our team lead gets an email whenever a card is moved, helping them keep track of the workflow even better. Trello should be a very useful tool that greatly contributes to the team's success moving forward.

M3:

Team # 4

Date of Milestone 3 review with Prof. D. Petkovic: 11/18/2020

Date of this summary and plan document: 11/18/2020

Summary of feedback and tasks to do:

1. UI

Home Page

- Move About/Post buttons more to the left
- Make overall font larger
- Reduce # of clicks – Add contact seller button to cards on homepage

Search Page

- Simplify filter – prefer sorting instead
  - o Sort by price (i.e., low to high, high to low, etc.)
    - To the right of “Showing 20 results…”, below nav bar
- Contact Seller – Prefill info from listing (name, date, ect) If not logged in, redirect to l login/register
- Make listing details page open in new tab

Registration Page

- Top right of register box: “All fields required”

Post Listing Page

- Add cancel button left of “Submit” button
  - o Under, add “may take up to 24 hours to approve of listing”
- Add “all fields required”

Dashboard

- Add “Welcome, Username/User ID” to page
- Add # of listings user has on Listings tab

2. Database

List Categories

- Categories must be foreign key

Message Database

- Add listing id

3. GitHub

Applications

- Create a frontend folder, just like backend directory
- Move images directory to backend

Branches & Commits

- Make commit messages more descriptive

Coding Style

- Add a basic header to each file
  - o Add 2-3 lines at the top for description
  - o Add name of owner of module

4. Miscellaneous

- Feature Freeze: do not add any extra features outside of those specified in M1
- Complete M3 summary

List of tasks the team chose to focus on and implement for final delivery:

All recommended changes are doable and will be implemented.
Additional features left to add:

- Messaging functionality
    - Notify users when they receive a message
    - Potentially add two way messaging instead of the buyer sending the seller a single message
- Flesh out user dashboard
    - User should be able to see their listings
- Class search
    - Might involve inheritance structure for listing table
- Listing expiration system (similar to craigslist)
- Miscellaneous Admin functionality (Low priority)

List of final product P1 functions agreed at the meeting

In our meeting with Dr. Petkovic, we never discussed P1 requirements. We will be working off our revised M2 P1 list:

1. Unregistered Users

1.1. Shall be able to search for items by item name, class, category
1.2. Shall be able to use their SFSU email to create an account

2. Registered Users

2.1. Shall be able to post item/service **listings**
2.2. Shall be able to send **messages** to sellers
2.3. Shall login with a username and password

3. Admins

3.1. Shall be able to remove **listings**
3.2 Shall be able to approve **listings** to go live

3.3. Shall be able to ban **user** account

M4:

SW Engineering CSC 648/848 Fall 2020
Team 4

Milestone 4
GatorTrader

Niall Healy (Team Lead, nhealy@mail.sfsu.edu)
Aaron Lander
Dale Armstrong
Lukas Pettersson
Joseph Babel
Vern Saeteurn

Date submitted	Date(s) Revised
December 8, 2020	

Table of Contents:

1. Product Summary:

GatorTrader is a unique website that allows San Francisco State University students to search for, purchase, sell, and trade essential materials for their courses. Our free-to-use website provides users with core functions that will allow them to make use of excellent features such as search by category or class, post item listings, and a messaging system in addition to a clean user interface. **GatorTrader**'s core functions include:

1. **Unregistered Users**
    1.1. Shall be able to search for items by item name, class, category
    1.2. Shall be able to use their SFSU email to create an account
2. **Registered Users**
    2.1. Shall be able to post item/service **listings**
    2.2. Shall be able to send **messages** to sellers
    2.3. Shall log in with a username and password

GatorTrader is unique due to the need for a medium that provides SFSU students the ability to make transactions with other students. With **GatorTrader**, students are able to message owners of post listings so that they have all the information they need before committing to a purchase.

URL: http://ec2-3-21-104-38.us-east-2.compute.amazonaws.com/

2. Usability Test Plan:

Test Objectives

We have decided to test the search capabilities of the GatorTrader website. Search was chosen because we feel it is one of the most important aspects of our website. Without search, GatorTrader is nearly useless.

Many criteria will be used to test search capabilities, including:

- Ease of finding and using the search bar
- The quickness of the results being loaded
- The satisfaction with the results received

With the feedback provided, we are hoping to gain insights into how we can provide the optimal experience for GatorTrader users.

Test Background and Setup

- System Setup:
    - Users will test the GatorTrader website using both Google Chrome and Firefox browsers.
- Starting Point:
    - The starting point of all tests will be on GatorTrader's home page.
- Intended Users:
    - Students and Faculty of San Francisco State University
- URL to be tested:
    - http://ec2-3-21-104-38.us-east-2.compute.amazonaws.com/html/index.html
- What is being tested:
    - Ease in finding and using the search bar
    - Quickness of results being returned
    - Receiving expected results from search

Usability Task Description

- Task
    - Search for a textbook on GatorTrader
- Success criteria
    - User finds an expected result from search query
- Benchmark:
    - Users are able to locate the search bar, enter a search query, and receive a result all within 10 seconds.

Effectiveness:

Effectiveness will be determined by:

- User's response to ease of finding and using search bar
- User's response to satisfaction of search results
- Any additional comments left by user

Efficiency:

Efficiency is measured by:

- Average time it takes for user to complete task
- User's response to results loading speed.
- Any additional comments left by user

Survey

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**	**Comment (Optional)**
The search bar was easy to find and use.						
The results loaded quickly.						
The results I received from my search were expected and satisfactory.						

3. QA Test Plan:

Test Objectives

Ensure the website displays the correct content based on the behavior of the search bar.

HW and SW Setup

Using Windows 10 with Google Chrome running version 87.0.4280.88, as well as Firefox running version 83.0.

Start with Google Chrome and go to the homepage. Each test should start on the homepage and after all page elements are finished loading. Once all tests are completed with Google Chrome, repeat the process with Firefox until the table is filled out.

The homepage is at the following URL:
http://ec2-3-21-104-38.us-east-2.compute.amazonaws.com/html/index.html

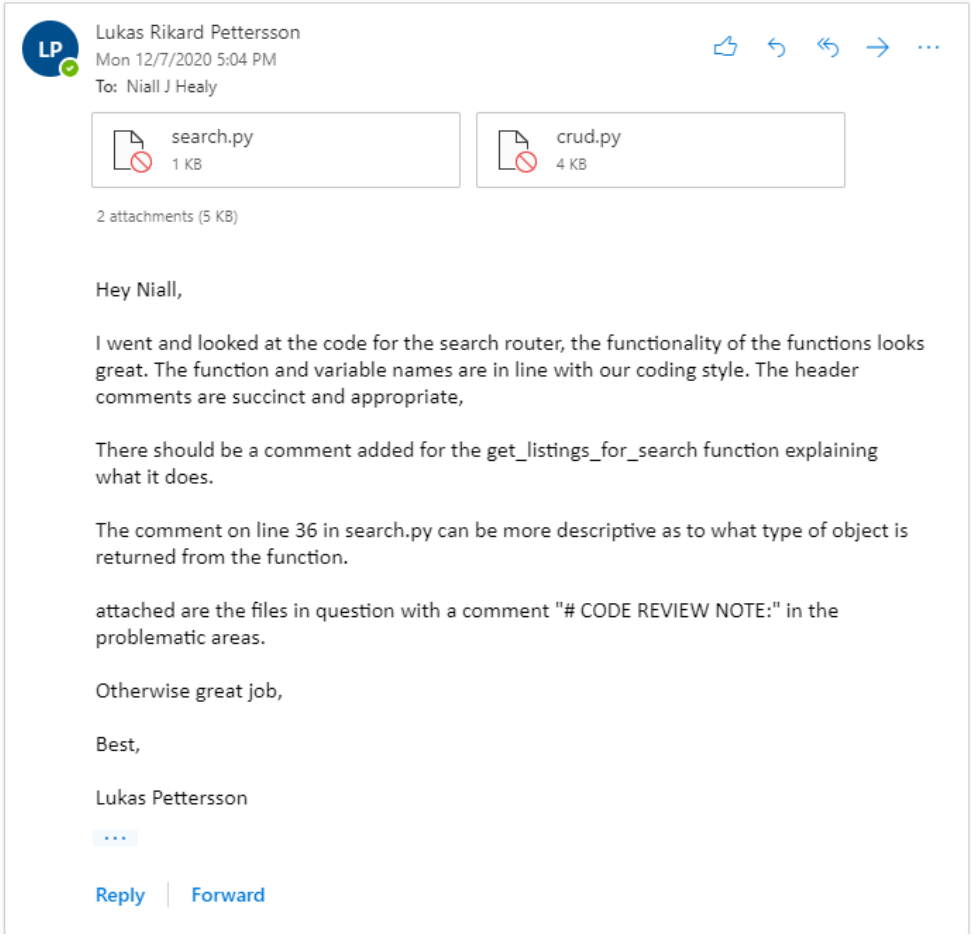
Testing Feature

The following test plan is to ensure expected behavior for the search bar.

QA Test Plan

Number	Description	Test Input	Expected Output	Chrome (PASS/FAIL)	Firefox (PASS/FAIL)
1	Test search persistence after a search	Type “book” in the search bar and click the search button	“book” will persist on the following results page	PASS	PASS
2	Test % like when searching by name	Ensure category is set to “Any” and type “calculus” into the search bar, then click the search button	1 result with the title: “precalculus with limits”	PASS	PASS
3	Test search by category	Click on category and set it to “Housing”. Ensure the search bar is clear and click on the search button	7 results showing all available housing	PASS	PASS

4. Code Review:



Lukas Rikard Pettersson
Mon 12/7/2020 5:04 PM
To: Niall J Healy

search.py
1 KB

crud.py
4 KB

2 attachments (5 KB)

Hey Niall,

I went and looked at the code for the search router, the functionality of the functions looks great. The function and variable names are in line with our coding style. The header comments are succinct and appropriate,

There should be a comment added for the get_listings_for_search function explaining what it does.

The comment on line 36 in search.py can be more descriptive as to what type of object is returned from the function.

attached are the files in question with a comment "# CODE REVIEW NOTE:" in the problematic areas.

Otherwise great job,

Best,

Lukas Pettersson

...

Reply | Forward

Figure 1: Code review email

```
26  # CODE REVIEW NOTE: ADD COMMENT HERE EXPLAINING WHAT THIS FUNCTION DOES AND RETURNS
27  def get_listings_for_search(db: Session, searchQuery: str, category: str):
28      if len(searchQuery) == 0:
29          if category == 'Any':
30              retVal = db.query(models.Listing).all()
31          else:
32              retVal = db.query(models.Listing).filter(models.Listing.category == category).all()
33      elif category != 'Any':
34          # Note: use like() for case sensitivity, ilike() for case insensitivity
35          retVal = db.query(models.Listing).filter(models.Listing.category == category,
36                                                   or_(models.Listing.name.ilike('%' + searchQuery + '%'),
37                                                       models.Listing.description.ilike('%' + searchQuery + '%'))).all()
38      else:
39          retVal = db.query(models.Listing).filter(or_(models.Listing.name.ilike('%' + searchQuery + '%'),
40                                                       models.Listing.description.ilike('%' + searchQuery + '%'))).all()
41
42      return retVal
43
```

Figure 2: Part of crud.py pertaining to search with comments on suggested fixes

```
"""
This file is used to route HTTP requests from the frontend to the appropriate place in the backend.
"""

# instantiates an APIRouter
router = APIRouter()


# returns a JSON formatted response of listings from the %like search
@router.get("/search/", response_model=List[schemas.Listing])
async def read_listings(keywords: str, category: str, db: Session = Depends(get_db)):
    return crud.get_listings_for_search(db, keywords, category)


# returns results html page
@router.get("/results/", response_class=HTMLResponse)
async def get_results_page():
    with open("/var/www/html/results.html") as f:
        html = f.read()

    return html


# returns list of categories
# CODE REVIEW NOTE: MORE DESCRIPTIVE COMMENT
@router.get("/categories/", response_model=List[schemas.CategoryReturn])
async def get_categories(db: Session = Depends(get_db)):
    return crud.get_all_categories(db)

```

Figure 3: search.py with comments on suggested fixes

5. Self-check on best practices for security:

Asset	Threat	Control
Users	Bad Actor uses XSS (Cross Site Scripting) to inject malicious code into a User's html.	1. Validate form inputs. Including character limit, size limit, file upload types, and limiting types of characters to only alphanumeric and dashes.
User Information	Bad Actor gains access to data through Buffer Overflow.	1. Validate all input fields. 2. Check and Limit the number of characters that the user may submit per input field.
User Information	Bad Actor gains access to a user account through brute force password.	1. Force a minimum number of characters for password to ensure the password is decently strong. 2. Include Captcha when signing in to limit bots capabilities to brute force.
User Information	Unauthorized users gain access to another user's data through a GET or POST request.	1. Use authorization tokens generated by JWT and a strong Secret Key to ensure that only the user with the correct token is able to access their data.
User Information	Unauthorized users gain access to another user's data through a man in the middle attack, listening to data being sent to and from the User and Database.	1. Will attempt to implement HTTPS utilizing a free certificate from the CA (Certificate Authority) Let's Encrypt.
User Passwords	If someone gains access to the database, they would now have the user's passwords that may be used on other sites.	1. Hash and Salt the passwords using bcrypt. One way encryption makes it harder to decrypt.
Database	Corruption through faulty data or mishandling	1. Create a backup and restore option. Backup every time the

		database is updated. 2. Protect the images folder
Database Integrity	Unauthorized users may attempt to use SQL Injection to manipulate the database or retrieve user's data.	1. SQLAlchemy classes and functions provide automatic escaping of special characters that may be used for SQL Injection.
Database Integrity	Spam bots create accounts to spam and upload faulty data.	1. Include Captcha when registering for an account to limit the ability for bots to create accounts.
Server	Bad Actor attempts to SSH into the server to gain access to information and manipulate the server files.	1. Utilize a strong key-pair to protect the server from unauthorized access. This key-pair is provided by Amazon.
Server	Bad Actors attempt to perform a DDoS (Distributed Denial of Service) attack to attempt to take down the server.	1. Utilize Amazon services to host the site. Amazon AWS Shield is automatically enabled for sites hosted on AWS. This provides protection against most common DDoS attacks.
Records Integrity	Users may upload malicious data or images to harm the reputation of the site.	1. Moderation - All posts must be pre approved before they can be seen by other users.

Password Encryption

- Stored user passwords are hashed and salted using bcrypt.

Input Validation

- All form input fields have a character limit along with limiting characters to alphanumeric and dashes. 40 max length for search is implemented.
    - maxlength and pattern attributes in html to control search input.
- Ensure the user emails start with proper names and end with @sfsu.edu or @mail.sfsu.edu
    - Regex in Javascript
- Ensure the email is not already used.

    - Query the database and check if the email is already in use.
- File upload - Number of files, file size, and file type limits.
    - Javascript to check all the variables
- Password 6 characters minimum and 40 characters maximum.
    - Javascript checks character length and notifies user
- Password confirmation matches.
    - Javascript to check if both passwords are the same

6. Self-check: Adherence to original Non-functional specs:

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO). **DONE**
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers. **DONE**
3. All or selected application functions must render well on mobile devices. **DONE**
4. Data shall be stored in the database on the team's deployment server. **DONE**
5. No more than 50 concurrent users shall be accessing the application at any time. **ON TRACK**
6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **DONE**
7. The language used shall be English (no localization needed). **DONE**
8. Application shall be very easy to use and intuitive. **DONE**
9. Application should follow established architecture patterns. **DONE**
10. Application code and its repository shall be easy to inspect and maintain. **DONE**
11. Google analytics shall be used. **ON TRACK**
12. No e-mail clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application. **DONE**
13. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI. **DONE**
14. Site security: basic best practices shall be applied (as covered in the class) for main data items. **DONE**
15. Media formats shall be standard as used in the market today. **DONE**
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. **DONE**
17. The application UI (WWW and mobile) shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2020. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). **DONE**

3. Product screenshots:

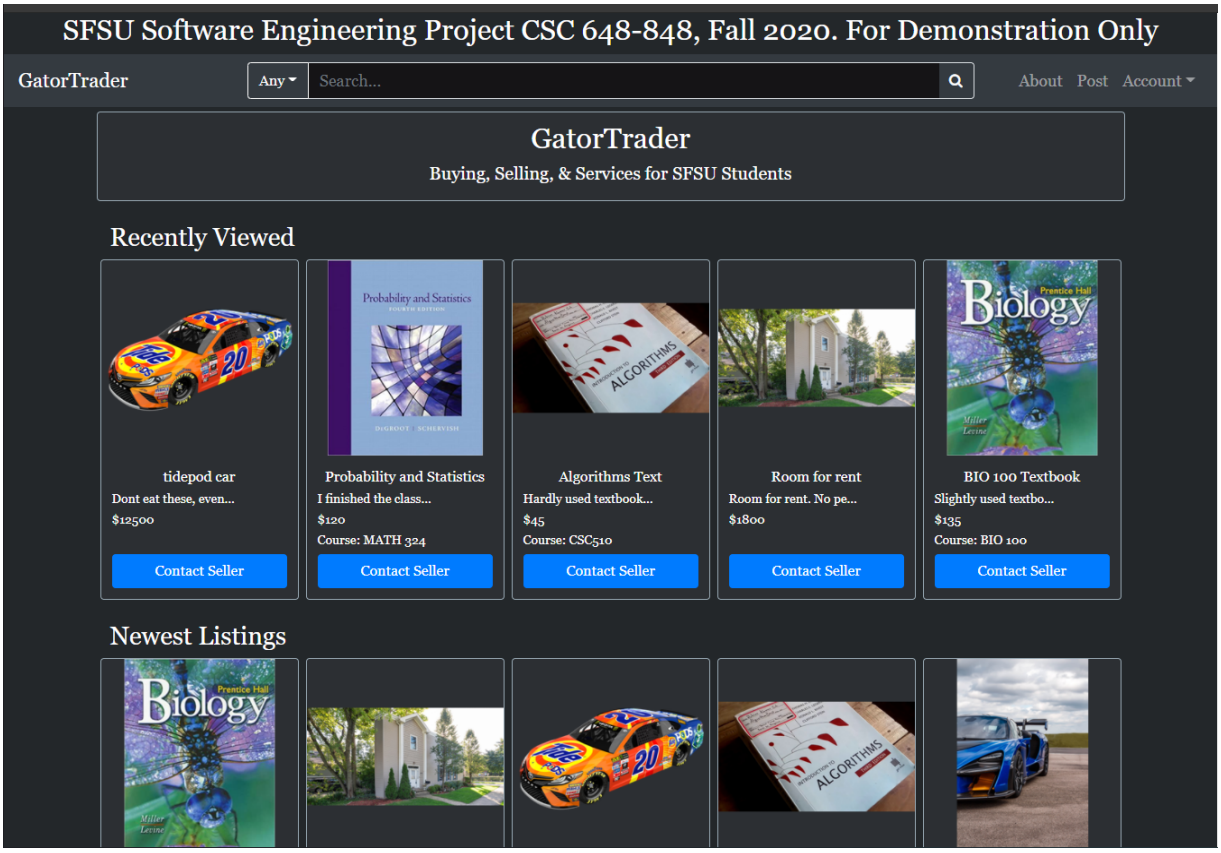


Figure 4: Home page

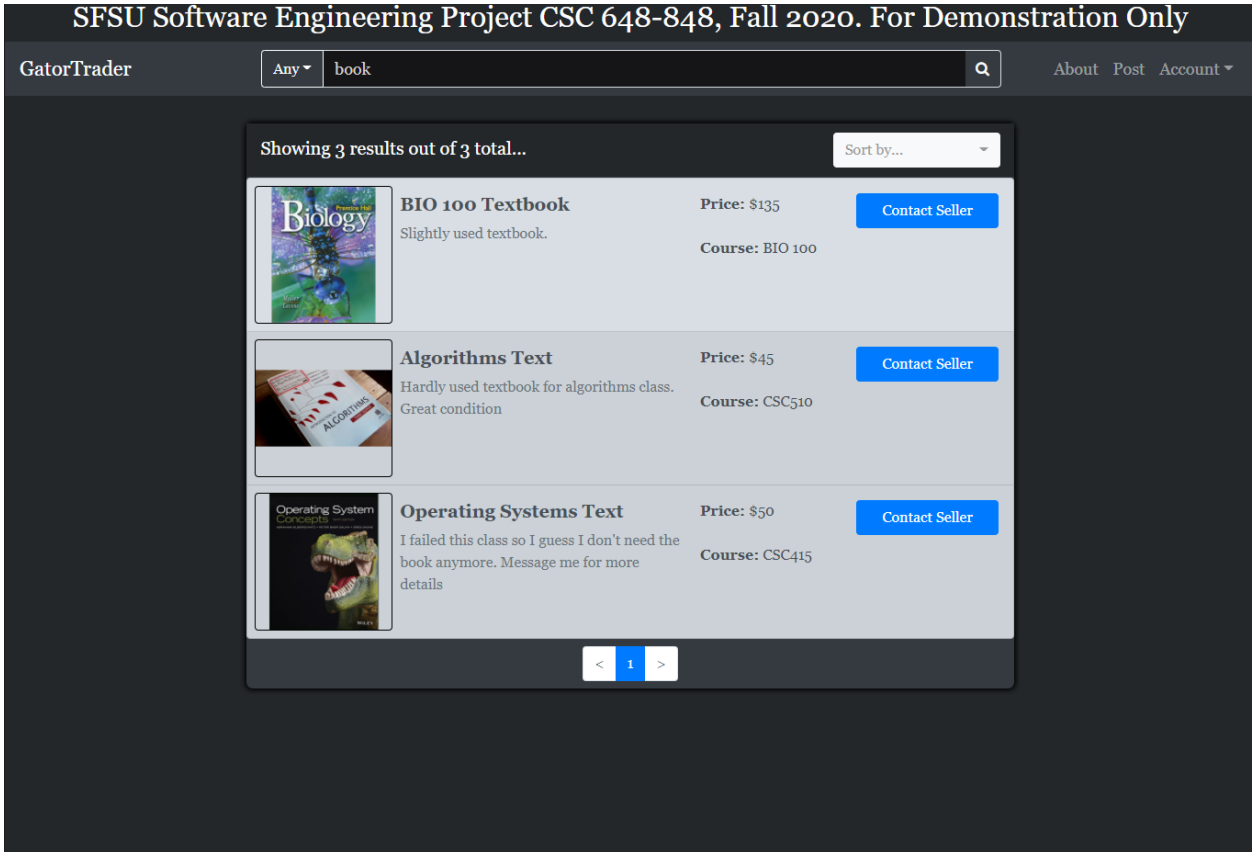


Figure 5: search result page

Figure 6: about page

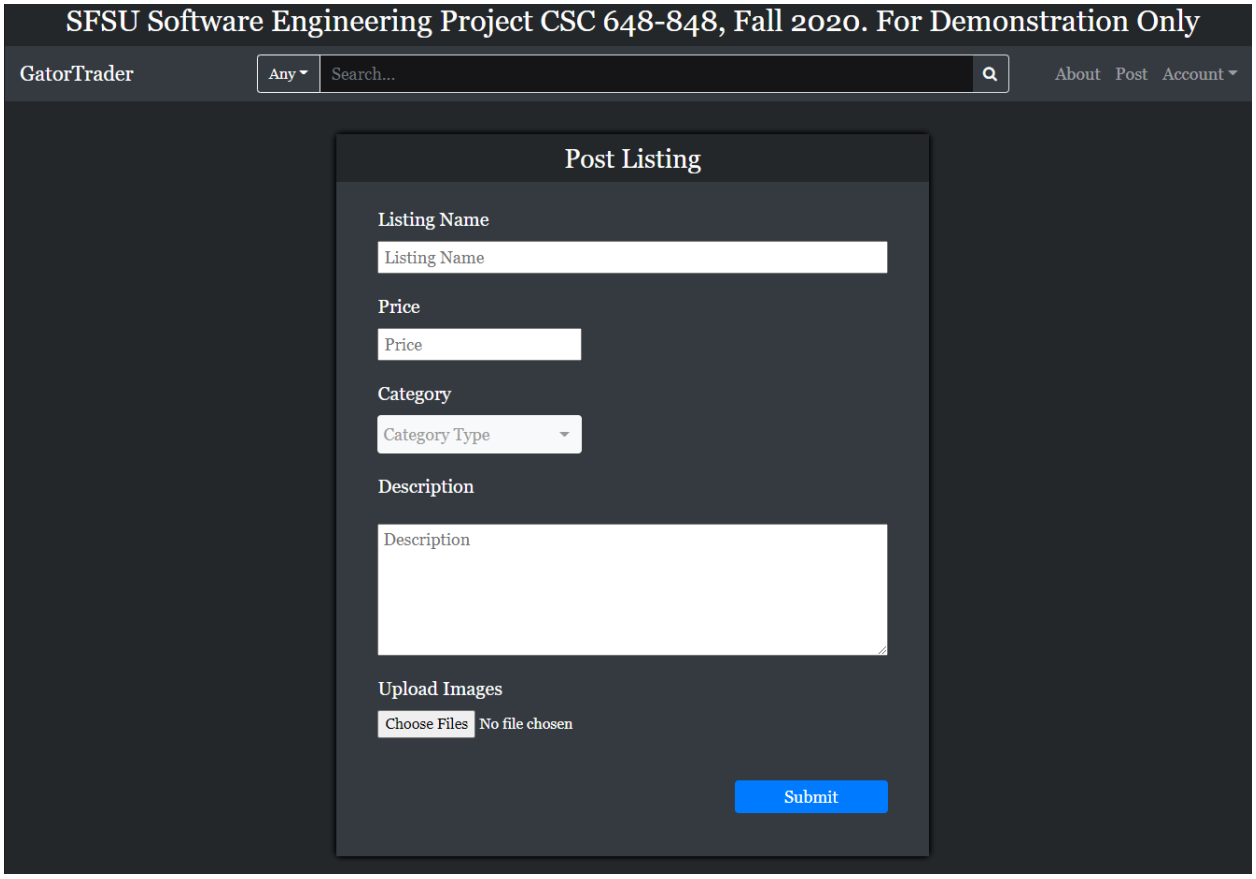


Figure 7: post listing

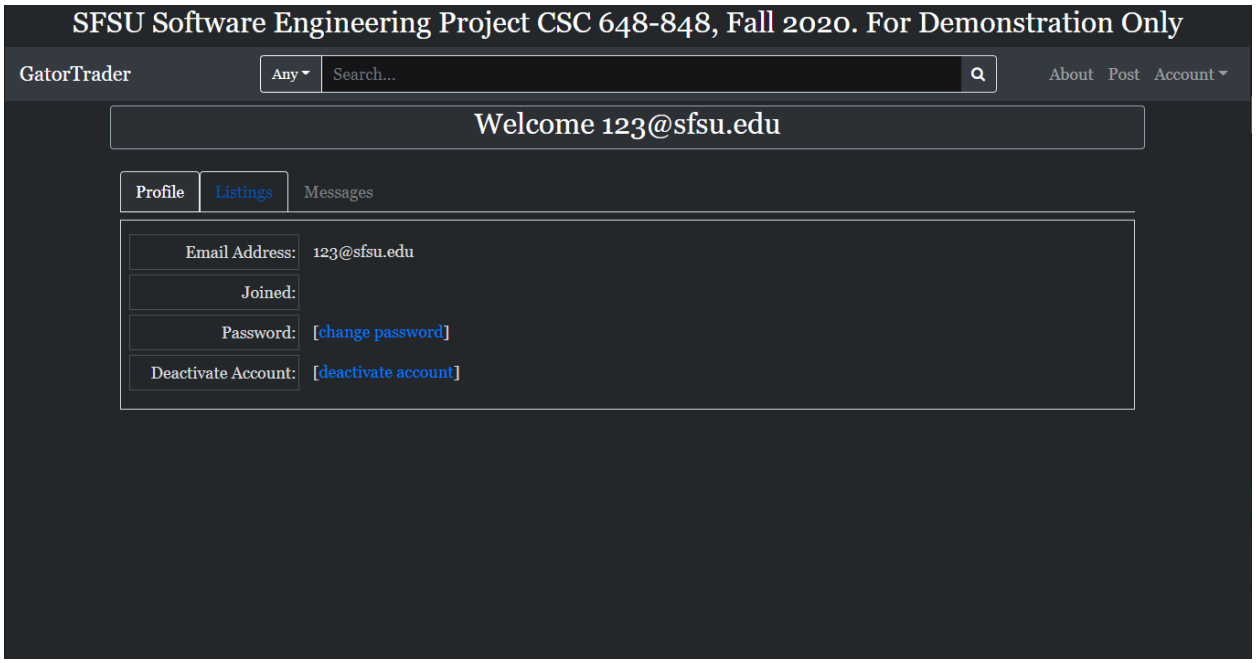


Figure 8: profile page

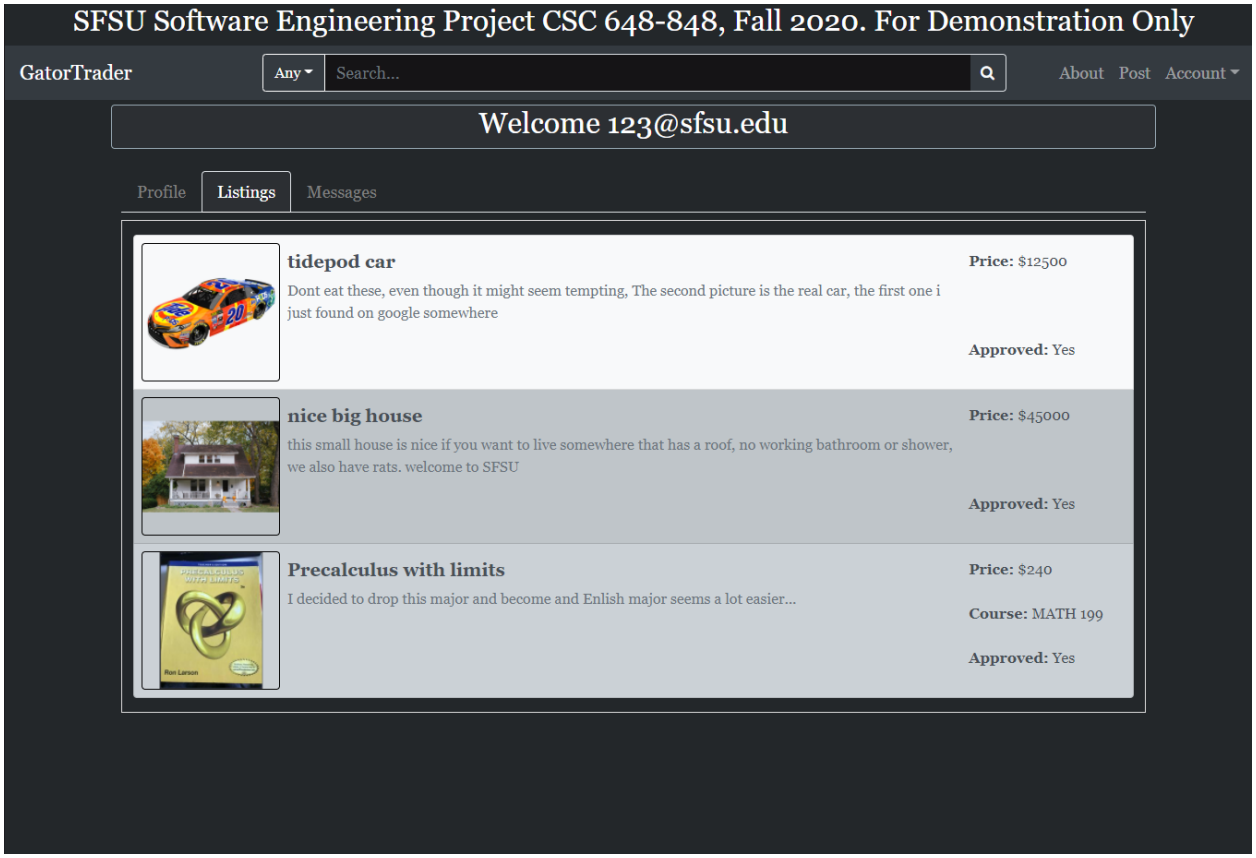


Figure 9: profile listings

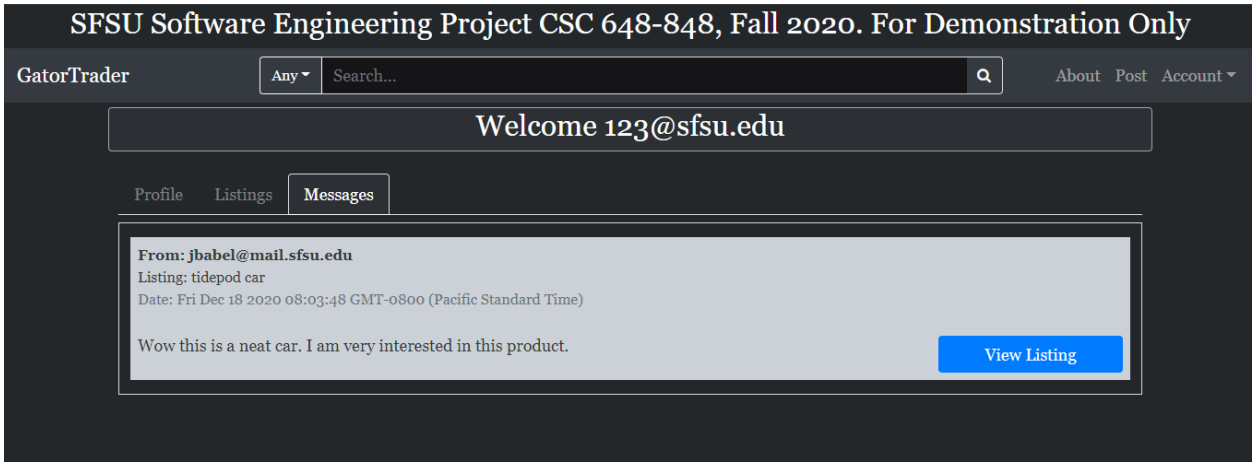


Figure 10: profile message

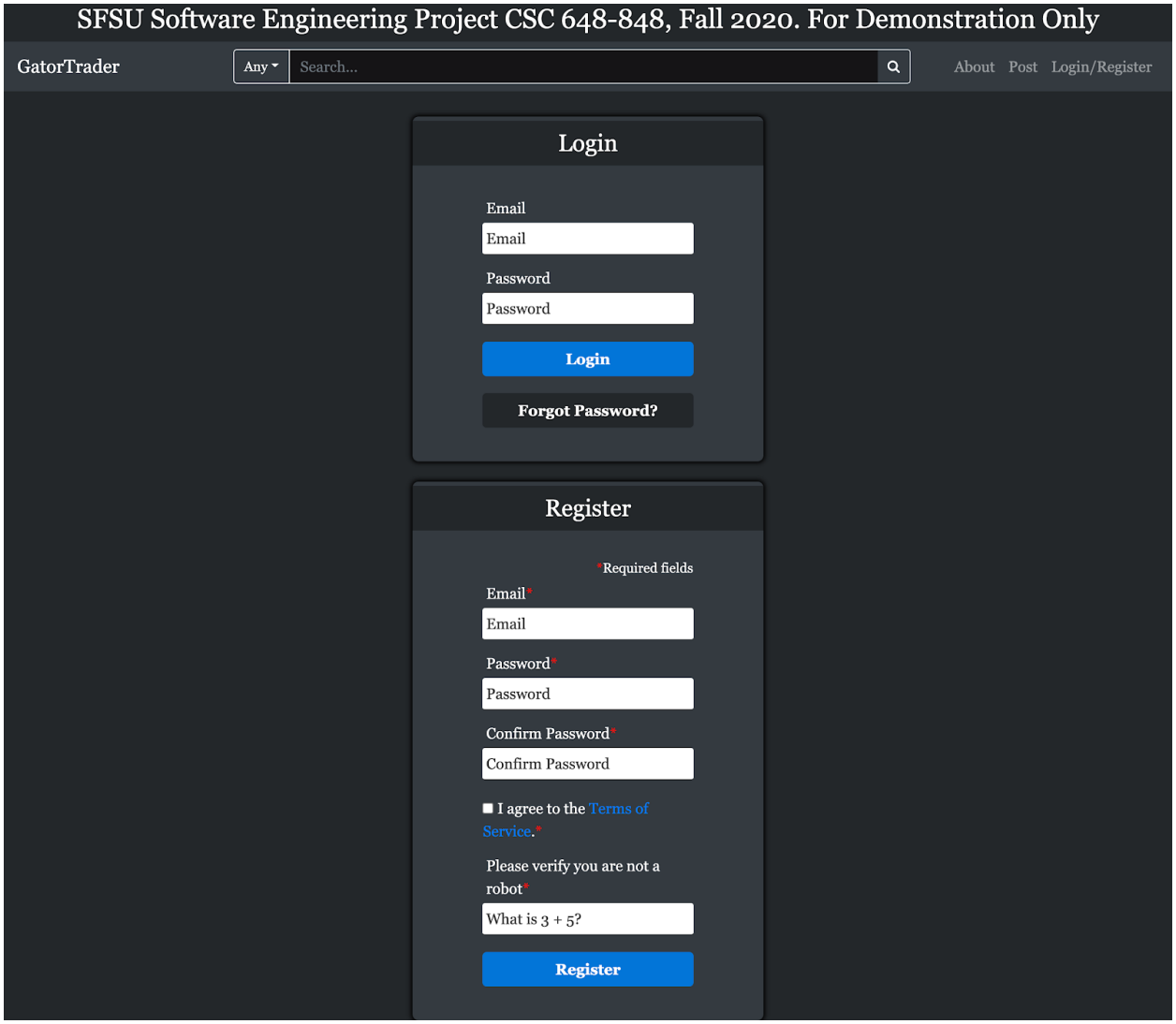


Figure 11: login/ register

4. Database organization:

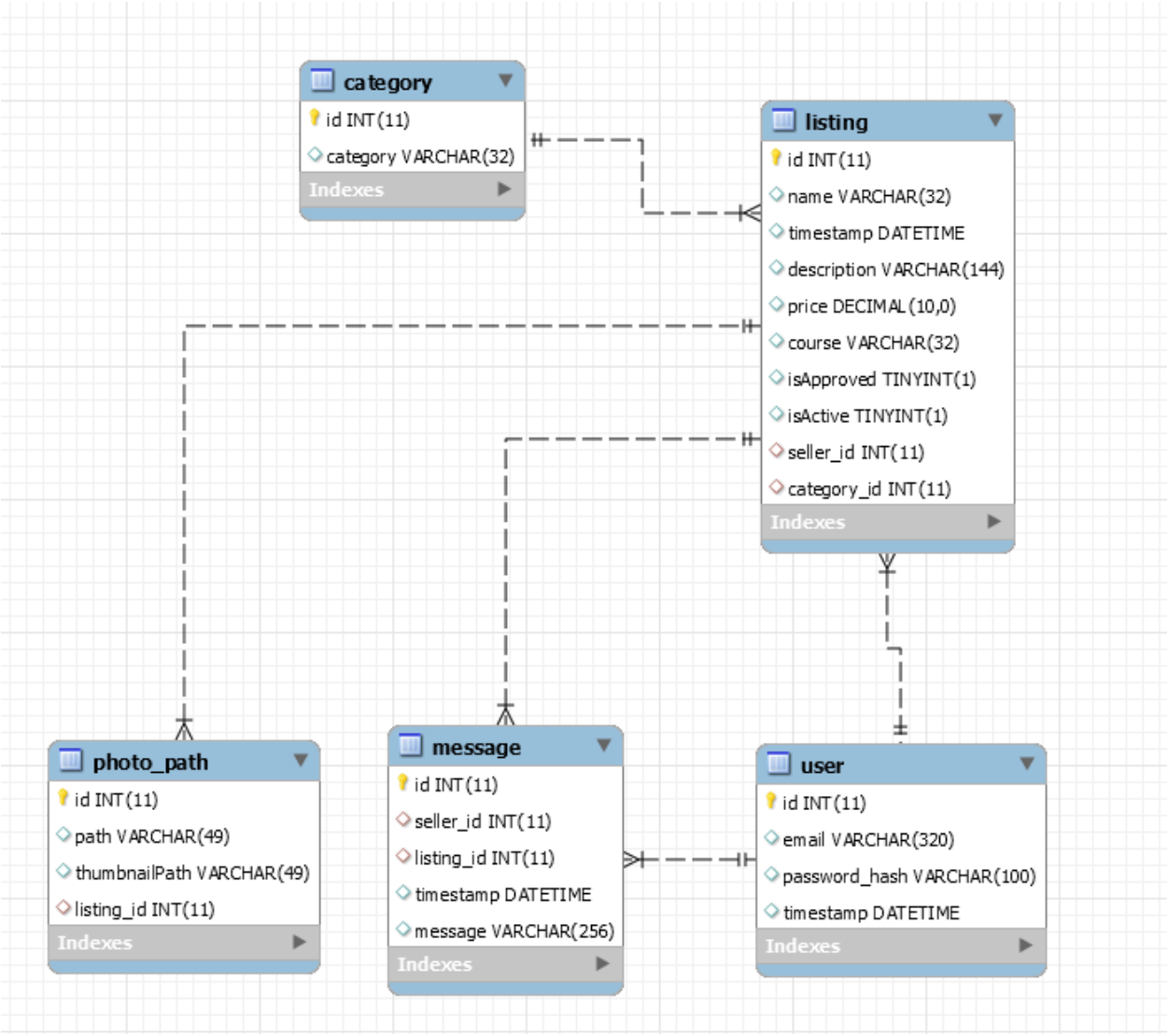


Figure 12: Database tables and relationships

Query 1 | user | listing | listing

Limit to 1000 rows

```
1 • select * from listing;
2 • select * from message;
3 • select * from photo_path;
4 • select * from user;
5
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	timestamp	description	price	course	isApproved	isActive	seller_id	category_id
1	BIO 100 Textbook	2020-12-18 07:24:10	Slightly used textbook.	135	BIO 100	1	0	2	2
2	Room for rent	2020-12-18 07:24:10	Room for rent. No pets allowed.	1800	NULL	1	0	2	3
3	tidepod car	2020-12-18 07:24:10	Dont eat these, even though it might seem tem...	12500	NULL	1	0	4	4
4	Algorithms Text	2020-12-18 07:24:10	Hardly used textbook for algorithms class. Grea...	45	CSC510	1	0	3	2
5	Cool expensive car	2020-12-18 07:24:10	Fuel inefficient but looks cool	75000	NULL	1	0	2	4
6	Room for Rent	2020-12-18 07:24:10	A walk in closet that isn't being used is available...	1000	NULL	1	0	1	3
7	nice big house	2020-12-18 07:24:10	this small house is nice if you want to live some...	45000	NULL	1	0	4	3
8	Strut tower bar	2020-12-18 07:24:10	Slightly used. It appears to have suffered some...	200	NULL	1	0	1	4
9	Lovely 1999 Honda Civic	2020-12-18 07:24:10	You know you want this beautiful Honda Civic	1000	NULL	1	0	3	4
10	Precalculus with limits	2020-12-18 07:24:10	I decided to drop this major and become and En...	240	MATH 199	1	0	4	2
11	2 bedroom apartment in...	2020-12-18 07:24:10	Great location and amenities. Message for addr...	3000	NULL	1	0	3	3
12	3 Bedroom House	2020-12-18 07:24:10	Selling a luxurious 3 bedroom 2 bath house. Op...	800000	NULL	1	0	5	3
13	Operating Systems Text	2020-12-18 07:24:10	I failed this class so I guess I don't need the bo...	50	CSC415	1	0	3	2
14	Probability and Statistics	2020-12-18 07:24:10	I finished the class and don't need this anymore.	120	MATH 324	1	0	5	2
15	Spacious Separate Garage	2020-12-18 07:24:10	We have a spare garage that is very spacious a...	600	NULL	1	0	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 13: Example of listing table in Workbench

5. Google Analytics:

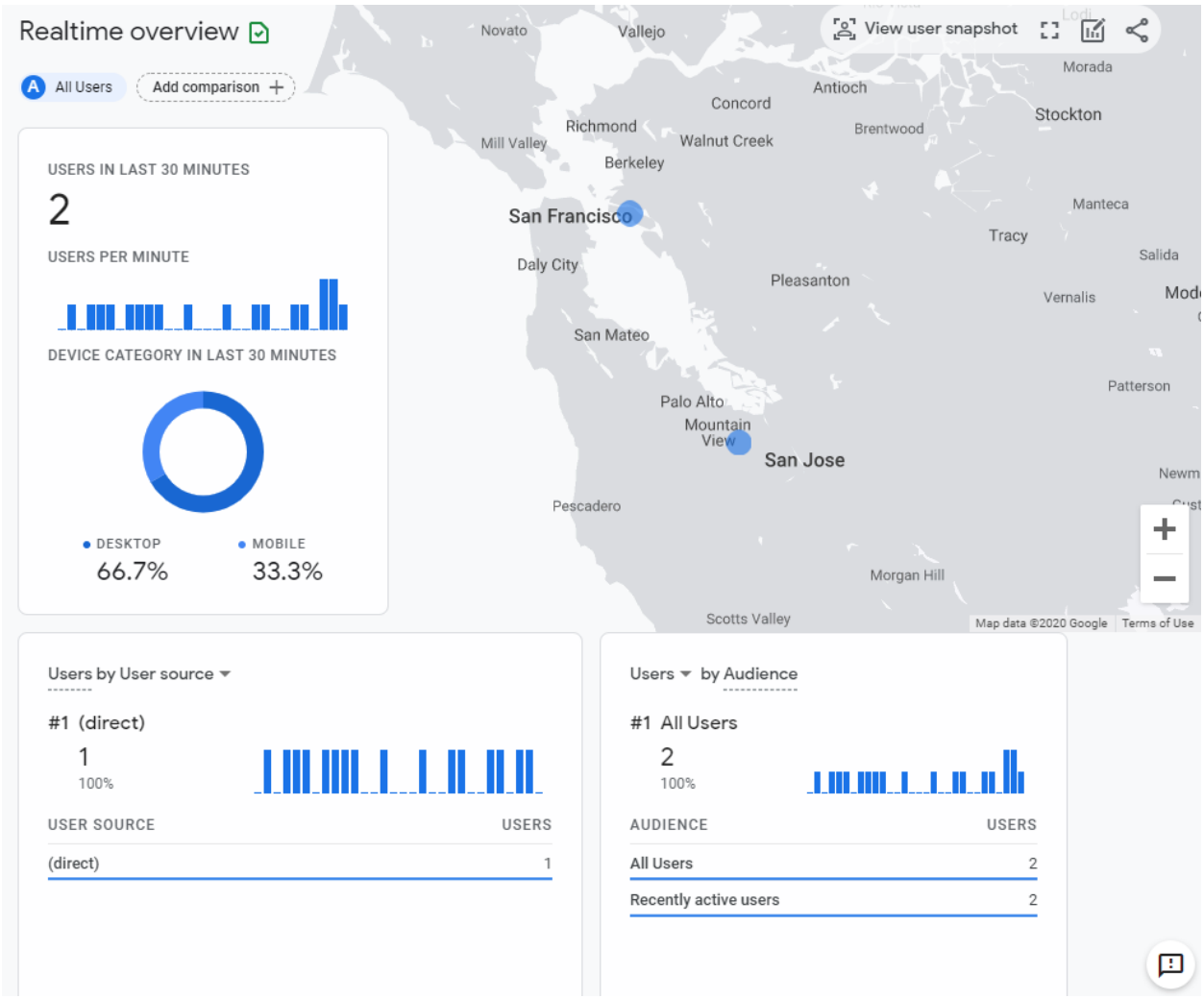


Figure 14: Google Analytics

6. Project Management:

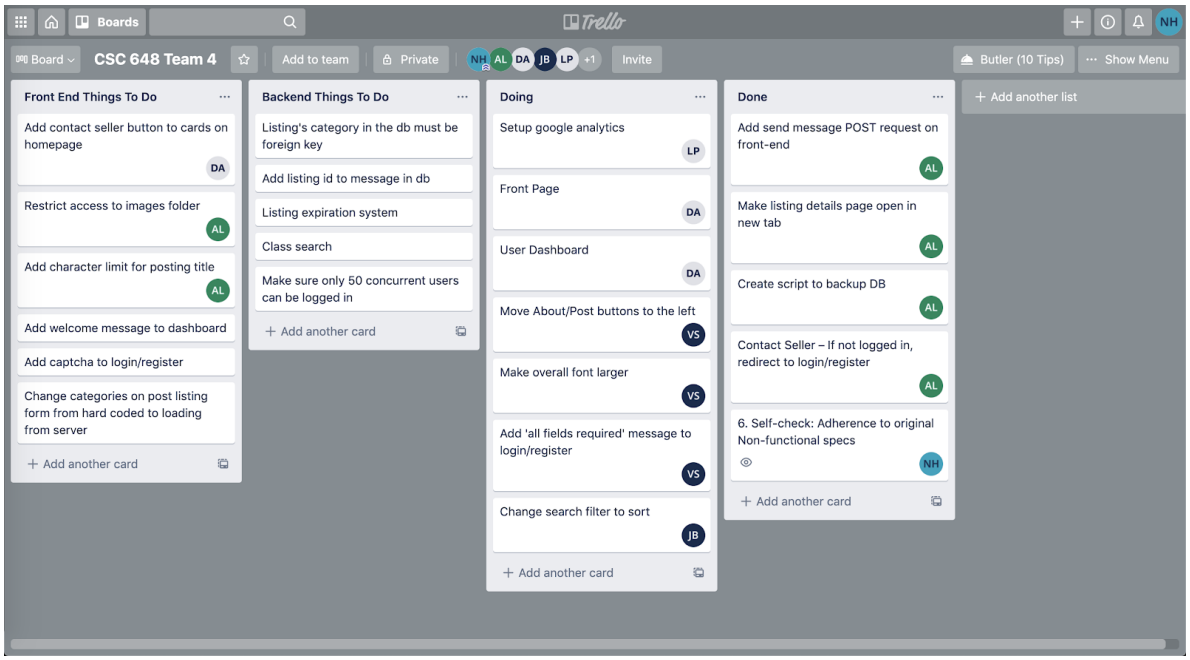


Figure 15: In Progress Trello screenshot

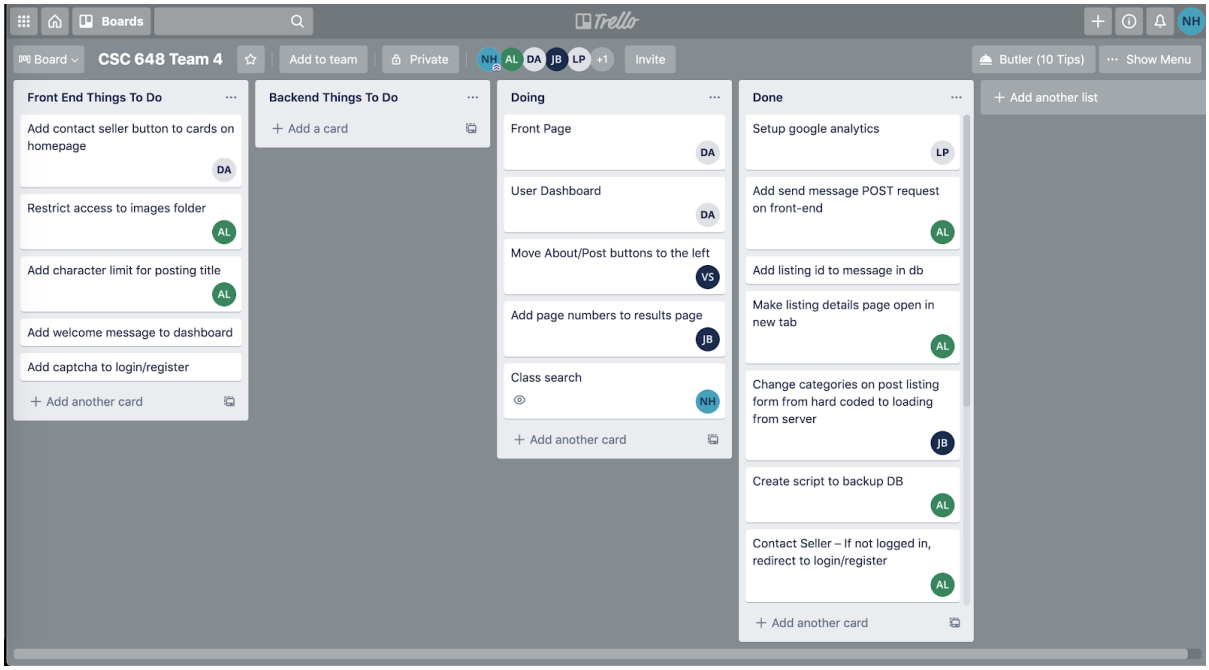


Figure 16: In Progress Trello screenshot

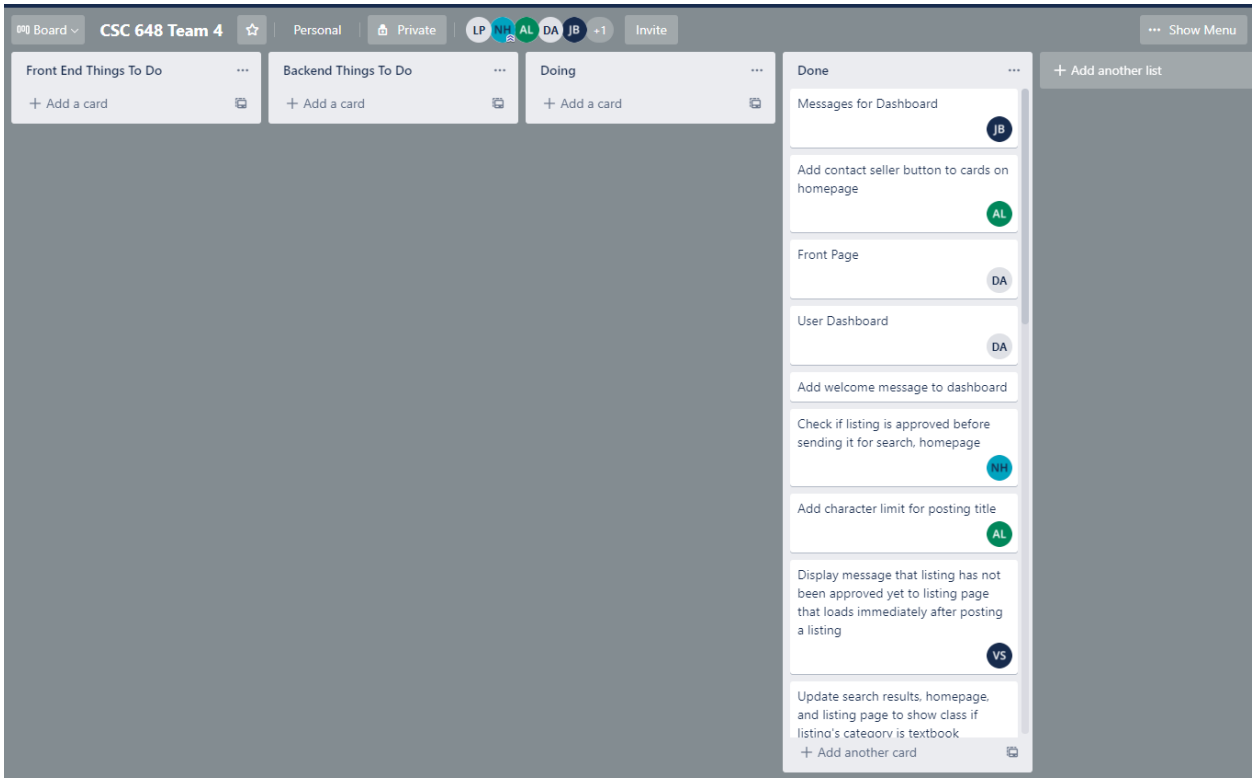


Figure 17: Final Trello screenshot

Figure 18: Commits over time from all members of team (Whal3man and LukasPettersson are both Lukas)

7. Team member self assessment and contributions:

NH

Niall J Healy
Thu 12/17/2020 11:57 AM
To: Lukas Rikard Pettersson; Aaron Lander; Joseph Babel **+2 others**

a) Contributions:

- Ran team meetings
- Managed Trello
- Search backend
- Login/Register backend & frontend functionality
- Document editing/formatting
- Various backend db code
- Miscellaneous bug fixes, testing
- Submitted Milestones
- Bothered team members to commit their code

b) Number of submissions (taken from output of 'git shortlog -s -n --all --no-merges'):

95 Commits

c) Main Challenges:

Being team lead with little prior leadership experience was a lot of work. It is quite a challenge to make sure that everyone gets their work done without being unpleasant to work with or overbearing. Keeping the team focused while encouraging everyone to get to know each other and discuss things outside of SE was also difficult. It was tough to choose and enforce a coding style (I could've done a better job).

d) What I'd do better:

I should've allocated work more evenly instead of taking on so much myself. I learned through trial and error how *not* to use tools like Trello and GitHub in a leadership position, and how to better use them in the future. I should've been more careful and tested my code more regularly. I pushed broken code a few times and wasted my teammates time in doing so. If I took this class again, I think I'd communicate with the professor more and ask more questions, as well as communicate more clearly what I expected and wanted from other team members.

Figure 19: Niall Healy

AL

Aaron Lander
Thu 12/17/2020 11:59 AM
To: Niall J Healy; Vern Chiem Chris Saeteurn; Dale Anthony Armstrong **+2 others**

Contributions:

- Managed the GitHub repo, ensuring conventions were being followed and protecting important branches.
- Participated in code reviews
- Created the listing page
- Created the contact seller pop-up window
- Added functionality for sending messages and retrieving messages
- Added functionality for resizing images
- Added form validation in login/register and post listing
- Created test server
- Created the MySQL server
- Created scripts to clear and populate the database, making testing much quicker

Number of commits to the dev branch: 64

Challenges:

Not many challenges, we all did a good job. If I had to come up with a challenge, I would say that we all could have communicated our progress a little bit better by utilizing Trello more (all of us, not just Niall). Other than that, it was an absolute pleasure.

Improvements for Next Time:

I think I've learned to have more trust in my teammates and to listen to their ideas more. Maybe it's PTSD from past group projects, but I've always just assumed the leadership position in groups and made most of the decisions. For this group, I learned to sit back and let others take over, and it was great because everyone contributed amazing ideas and I learned so much from you guys.

Figure 20: Aaron Lander

Dale Anthony Armstrong
Thu 12/17/2020 1:25 PM
To: Joseph Babel; Niall J Healy; Aaron Lander; Lukas Rikard Pettersson +1 other

a) Contributions:

- Created the initial about page template for the team to utilize
- Helped work on the mockups for what the pages may look like
- Create some personas along with the use cases
- Created the navbar and the associated JavaScript
- Created the initial landing page and the associated JavaScript
- Created the profile page and the associated JavaScript
- Created some python backend code to get some requests from the server
- Researched best practices for security and did a review of our current setup
- Helped edit the Milestones before delivery

b) Number of submissions (taken from output of 'git shortlog -s -n --all --no-merges'):
18 commits

c) Main Challenges:
I faced several challenges throughout this project, the main one was understanding how the whole system works, especially when communicating between the backend and the front end. The front end was a bit of a pain because getting the formatting and design the way you want it to look is complex. Even something as simple as designing and making the buttons/bars the same size and aligned correctly took a long time.

d) What I'd do better:
In the future I would break my commits up into smaller commits. My current way of designing was designing, creating, and finishing my portion then committing. I dislike dealing with merge conflicts but will work on it in the future. I now have a much better understanding of how the whole process of software engineering works and would spend more time in the initial phases detailing out the ideas. I felt like I spent more time creating the pages by piecemeal.

Regards,
Dale Armstrong

Figure 21: Dale Armstrong

Lukas Rikard Pettersson
Thu 12/17/2020 12:03 PM
To: Aaron Lander; Dale Anthony Armstrong; Joseph Babel **+2 others**

a) Contributions to team

- Google Analytics setup and implementation
- Live server setup and maintenance
- Message router and functionality
- Listing router
- Database table design
- Part of search router implementation
- Parts of various Milestone reports

b) submissions to the Dev Branch

- 39 total submissions to the dev branch

c) The biggest challenge that I faced in the project was my knowledge of the software stack and web development. Coming into this class I had no knowledge of how development worked on a technical level. Familiarizing myself and understanding how the backend and frontend talk to each other, how databases are built and maintained were all new concepts and I feel like I could have done more to learn how this worked to better assist my team.

d) For the next time that I work on a project such as this, I would spend more time on understanding the software stack and technologies that we are going to use in our project. As the backend lead I should have done more work and research on the backend of this project. I feel as though I let my team down in this regard and is something that I would like to improve on in the future.

Figure 22: Lukas Pettersson

JB

Joseph Babel
Thu 12/17/2020 12:05 PM

To: Niall J Healy; Dale Anthony Armstrong; Aaron Lander; Lukas Rikard Pettersson **+1 other**

a) Contributions:

- Several mockups of functional requirements
- Several personas, use cases, and storyboards
- QA Test plan
- Initial version of models and schemas
- Modified search router to allow for pages of listings
- Modified listings crud function to support sorting by price
- Overhaul of results page with sort by price and pages
- Creation of post listing form with basic fields and thumbnail preview
- Added functionality to display messages on the dashboard
- Fixed javascript and css links in header of HTML files
- Fixed general formatting issues in HTML and javascript files
- Helped teamwork through difficult technical issues
- Reviewed team members code on GitHub

b) Number of submissions (taken from output of 'git shortlog -s -n --all --no-merges'):
32 commits

c) Main Challenges:
I found working online more difficult than working in person, but it was still manageable. There were some challenging problems to solve and due to our chosen backend framework, we had some difficulty troubleshooting them with limited support from the web. This issue was offset by the overall ease of use of the framework, so the issues we ran into were not too frequent. There were also a few occasions we were pressed for time.

d) What I'd do better:
Next time I will aim for more frequent commits. I ended up putting too much work into individual commits that could have been broken up into smaller increments. There were also a few times I worked on low priority features when I should have been focusing on higher priority features. Next time I will have a better focus on what is important for the project.

Figure 23: Joseph Babel

Vern Chiem Chris Saeteurn
Thu 12/17/2020 12:31 PM

To: Lukas Rikard Pettersson; Joseph Babel; Aaron Lander; Niall J Healy; Dale Anthony Armstrong

a) Contributions

- M1: Competitive Analysis
- M2: Identify key risks
- M3: Project Status
- M4: Product Summary
- Front End: Login and register page
- Front End: Populate results page with server results

b) Number of submissions (taken from output of 'git shortlog -s -n --all --no-merges'):

- 37 commits

c) Main Challenges:

- As a team member that did not have a lead position (team lead, back end/front end lead, Github master), I did not have many personal challenges that I came up with. This is due to my other team members carrying most of the weight while I took on less challenging tasks. The only challenge I went through this semester was having an incident involving Covid-19, where I went through a brief period of heavy anxiety and unfortunately could not focus on any school-related work. Thankfully, my team members were kind enough to pick up my slack in order to meet milestone deadlines.

d) What I could do better:

- I believe that I could have taken the initiative of taking on more tasks to lessen the burden on my team members, even though I knew that I would struggle and take much longer to complete the work. I made repetitive mistakes when committing to Github, such as merging my code to the master branch instead of the current branch we were working on (ie. milestone-five branch). Also, I could have been more active on our Discord server as well as in our Zoom meetings, as I felt that I gave the least amount of input in our team.

Figure 24: Vern Saeteurn