BuffsMarket Project Report

Daniel Monteiro, Haeseo Jeong, Jaron Barrett,
Alex Paquier, Ethan Telang

Project Description: BuffsMarket is a marketplace for CU students for students to be able to sell and buy different items being sold safely. The website is built with 2-factor authentication to ensure only CU students can access the service. Students can create, edit, and delete listings and buyers can search for specific products with a search function. The CU Marketplace has a built-in chat feature that makes it easy for buyers and sellers to talk directly. They can use the chat to discuss prices, ask questions, and decide on a time and place to meet, like somewhere on campus or in a nearby area. This platform connects CU students in one community, making it simple to buy and sell items at affordable prices. It also helps promote sustainability by encouraging students to reuse and recycle things they no longer need. Whether you want to get rid of stuff or find great deals, the CU Marketplace is the perfect place for CU students.

Project Tracker: https://github.com/users/haejeg/projects/1

Video: https://youtu.be/TLN-2lgLV-8

VCS: https://github.com/haejeg/BuffsMarket/tree/main

Contributions:

Alex:

I worked on implementing the image upload and display features in listings, debugging the messaging system, and implementing the 2-factor authentication. For the image uploading I used Google Cloud API and created a bucket to hold our images for buffsmarket that could be accessed when we needed images for a specific listing. Additionally, I assisted Ethan in the debugging of the messaging system when he was creating it, eventually getting it to work. For the 2 factor authentication, I used nodemailer (node js) and a Gmail account I made to send the authentication code to the user. (I would put much more but only 100 words :(

Daniel:

I worked on creating the wireframes to set up the entire vision that we had for this project. I also implemented the login and registration pages as well as the authentication of the colorado.edu email. I helped set up the database and I implemented the necessary tools to be able

to launch the website on Render. The edit listing feature was implemented by me after Jaron set up the button I implemented being able to successfully edit the listing as well as being able to override the past images the users have posted, as well as implement the delete listing feature.

Haeseo:

I worked on implementing the database and some of the frontend designs (styles, CSS), as well as routes and requests through Express. I also helped set up render and worked on some scripting in index.js. I used CSS and HTML for the design, then Bootstrap and Handlebars for the UX interactions. I used Express.js and Node.js for the routing, as well as Render and Postgres for the back end. I also barely helped, but reworked some of how our tests were running using Mocha and Chai.

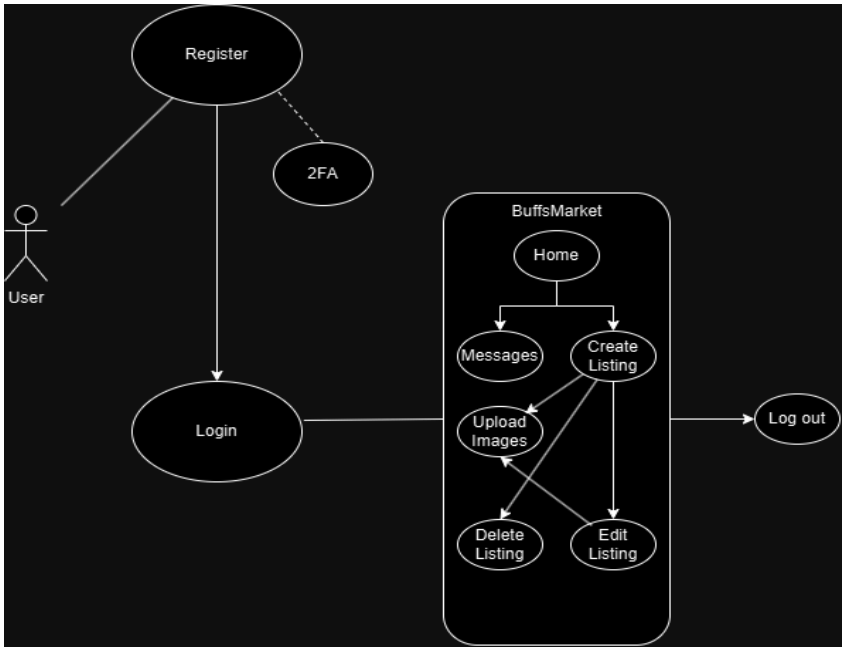
Jaron:

I worked on designing and implementing the database and the front-end and back-end for the home, listings, and my listings pages, in addition to the unit tests. For the database implementation, I used PostgreSQL. For the front end, I used CSS, Bootstrap, and Handlebars. For the back-end, I used Express.js and Node.js to implement API routes to interact with the database to display listings and render the pages. For testing, I used Mocha and Chai. I also attempted to implement live messaging with Socket.io but was unsuccessful.

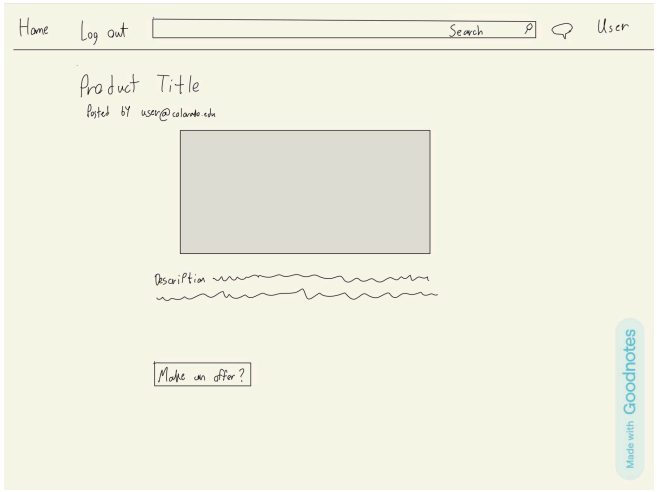
Ethan:

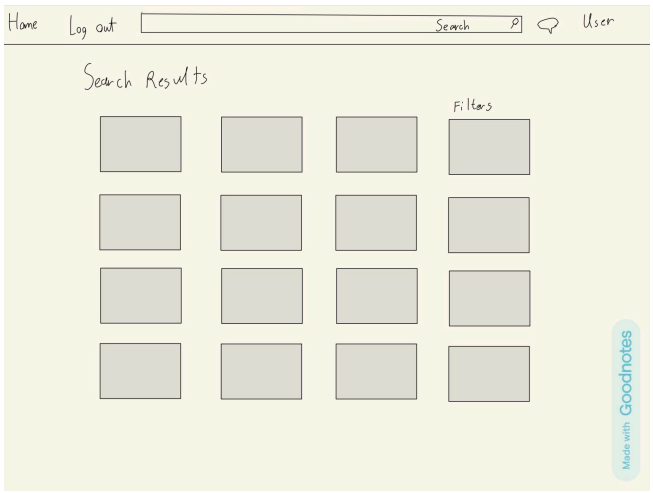
I primarily worked on debugging and implementing the messaging system. The messaging system was built through SQL by using functions that took in both the User ID and the properties of the messages such as Message ID, Content, Timestamp, etc. One of my favorite things that was implemented in our project was that if you clicked a listing there was a button that would start a chat with the person selling the item. For the front-end messaging system, we used handlebars and HTML to create a script where when they would start a chat it would create a whole new message box for that unique User ID. I also worked on debugging errors that would arise in our code. There would be multiple instances where different branches would fail to work after being pushed and I would help fix them. I would have to manually merge past github files with the modern code changes to then create a new branch that the team could work on. Other than those items the less important stuff I did was the ReadMe, Meeting Logs, and Milestones.

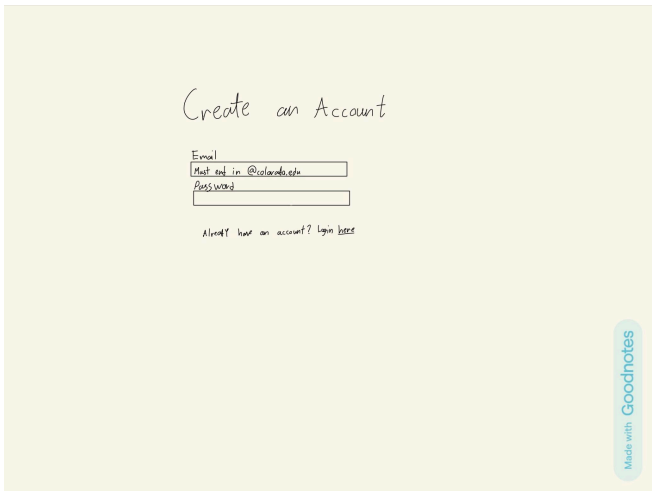
Use case Diagram:

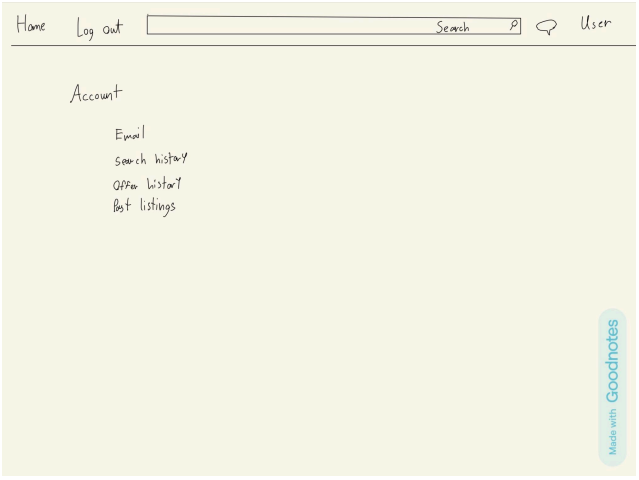


Wireframes:









Test Results/Observations

Our original User Acceptance Testing (UAT) focused on three primary features: successful registration of a new user, redirection to the login route after account registration, and rendering the login page with an HTML response. Initially, we tested these features using unit tests developed with Mocha and Chai. However, after redesigning our login system to include two-factor authentication restricted to valid @colorado.edu email addresses, we decided to test new features manually. This decision was based on the belief that manual testing would be significantly faster than extending automated tests for the new functionality. The inclusion of two-factor authentication also made hard-coded testing more challenging and time-consuming to implement.

For our updated testing, we identified four main feature test cases that reflect the core functionality of our application:

1. Registering and logging in with a valid @colorado.edu email address using two-factor authentication.
2. Posting a listing with up to ten images.
3. Editing an existing listing.
4. Messaging other users.

To ensure robust testing, we enlisted two CU students to evaluate the two-factor authentication and messaging features.

Findings and Observations

1. <u>Register and Login Functionality</u>
   Users successfully created accounts and completed the two-factor authentication process without needing additional guidance. This confirmed that the system was intuitive and user-friendly.

2. Listing Posting and Image Uploading
   Users found the listing creation process straightforward and intuitive. However, some users inquired about how to specify the category of their listing, a feature we had not yet implemented. If we were to continue developing this project, completing the implementation of listing categorization would be a high priority.
   Users also pointed out the absence of functionality to add or remove images after a listing was created, which was due to unresolved issues with our integration of Google Cloud Storage. This limitation, along with the inability to delete a listing, was frequently mentioned during testing. Based on this feedback, we concluded that implementing these features should be a top priority in future iterations.
3. Messaging System
   The UI and basic functionality of the messaging system received positive feedback. However, users expressed a strong preference for real-time messaging without the need to reload the page. While we attempted to develop a live chat feature in response to this feedback, we were unable to complete it due to time constraints.

Deployment: https://buffsmarket.onrender.com/