CSCI 201: Group 1 Documentation

Table of Contents

Project Proposal

Our project idea is to create a web app which will work like Facebook Marketplace but for USC students. Users can upload listings which are the items they want to sell, our web app will support user messaging and account creation.

High-Level Requirements

Users (USC students, faculty, alum) should be able to put up postings for items they would like to sell. Listings should allow for a price, photo, title, and description. It should also allow sellers to tag items with premade categories to reach customers quickly.

Users who would like to purchase items should be able to search for items based on the title. Users should also be able other users on the app regarding product details, payment processes, and meetup location.

There should be an authorization system for users to log into their specific account for listings or purchases they have made. Users should be able to give each other ratings based on their experience buying or selling from each other.

If users have not created an account, there should be a create an account page with users being able to link their USC email. They will then set up a password for their account. For users that have created an account, they should be able to login with their email and password they created.

Technical Specifications

Web Interface (20 hours)

- The first page users will be brought to is a login page. It will have an email/username field, password field, and Login button.
- Upon verification, users should be brought to a page with listings of potential items as tiles filling the screen, a search bar on the top to look up specific items, a button on the bottom to create a listing, and a menu icon to pull a side window with an account button and messaging icon.
- The account button will allow users to see their ratings, username, and email.
- To create a listing, users will fill out a form with fields for the title, price, description, thumbnail, product images, and category. Each user will have a profile page that displays their posted listings and transaction history. After completing a transaction, buyers can rate sellers based on experience, similar to applications such as Depop.

Messaging Feature (8-10 hours)

- Will record messages exchanged between users, tracking sender IDs, message content and timestamps.
- Messaging will be a large open chat room for users to ask questions about specific listings
- Users will be able to see all previous messages

RestAPI Backend Server (6 hours)

- Connects the Web Interface to the Database
- Makes use of clear API endpoints to allow authorized users (users with verified accounts) to perform CRUD operations for the following:
    - Product Listing (User and Admin)
    - Account creation/management (User and Admin)
    - Messaging API endpoints

Database (8 Hours)

- Our database will consist of four tables
- User table
    - User ID
    - User Name
    - User’s password (hashed)
    - Rating
    - User Email
- Product table
    - Product ID
    - Product name
    - Price
    - Product description
    - Thumbnail
    - Reference images
    - SellerID

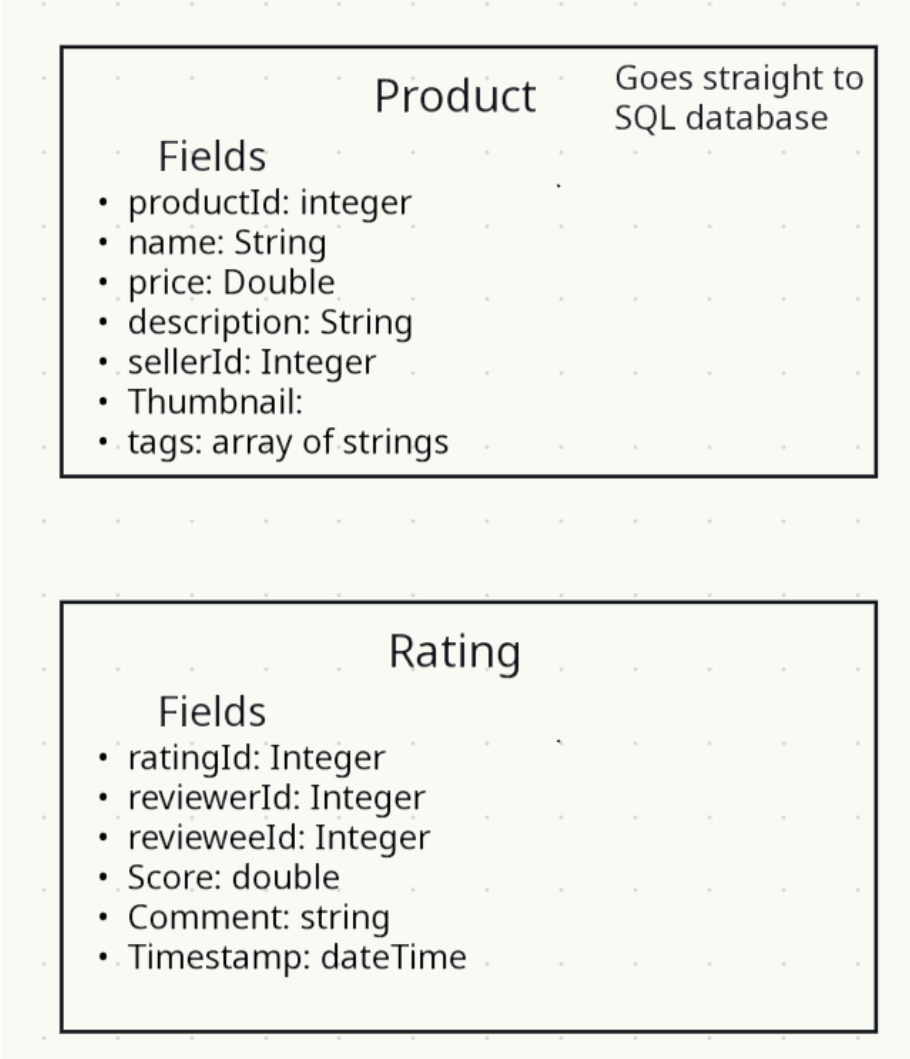
    - Tags/Categories
- Message table
    - ID
    - Sender ID
    - Message text
    - Timestamp
- Ratings Table
    - Id
    - buyerID
    - sellerID
    - Stars
    - Comment
    - createdAt

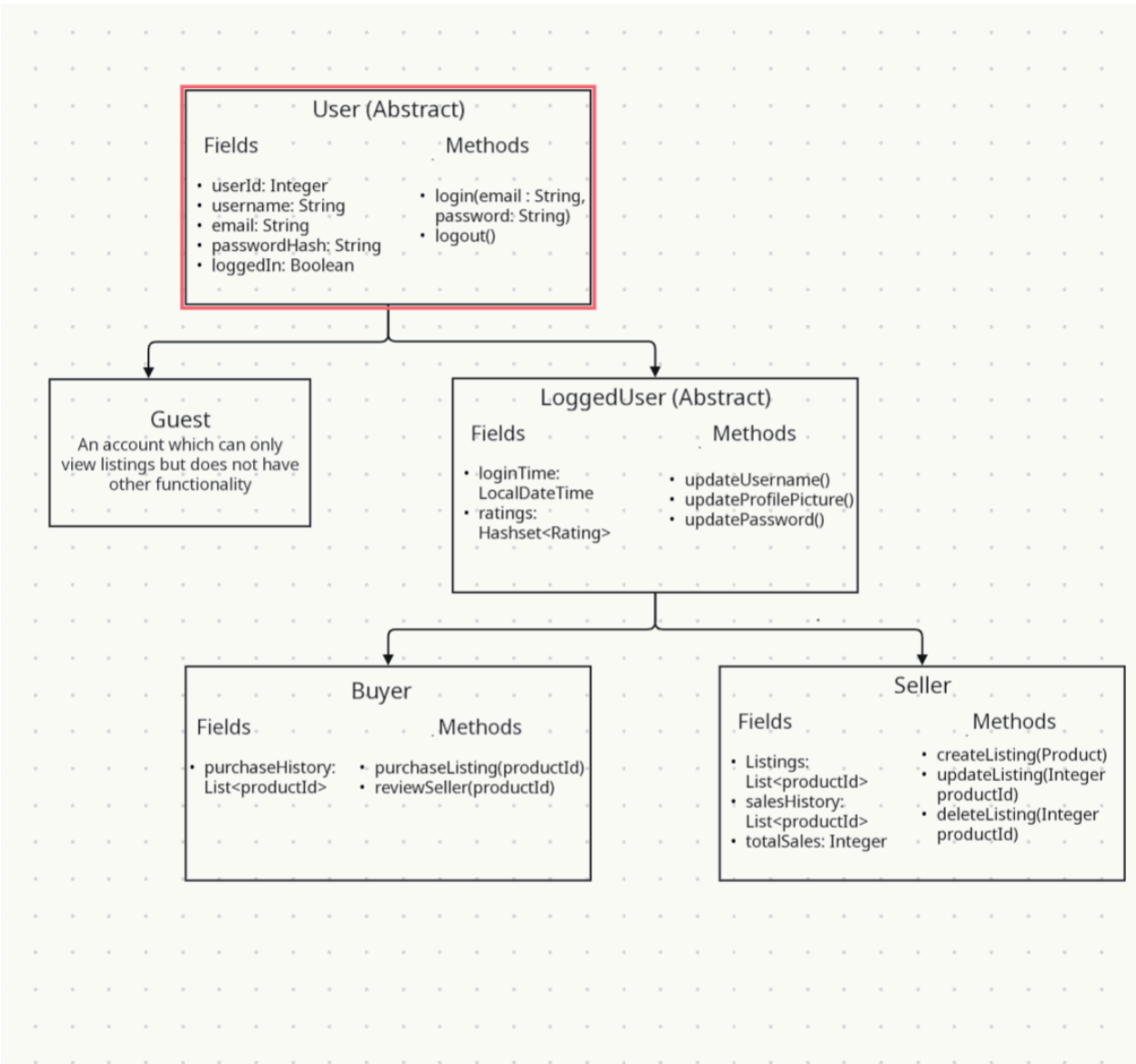
Detailed Design Document

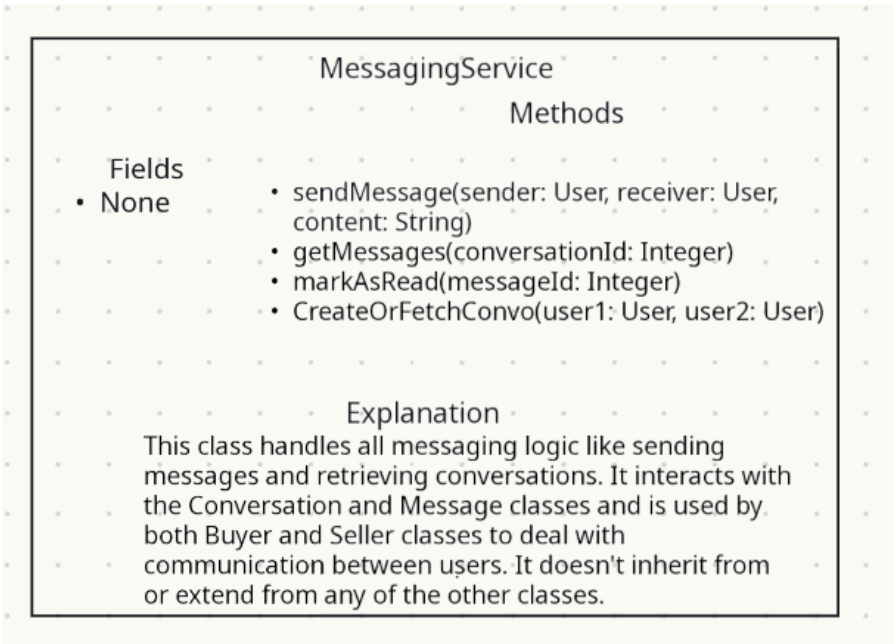
Class Diagram and Inheritance Hierarchy:

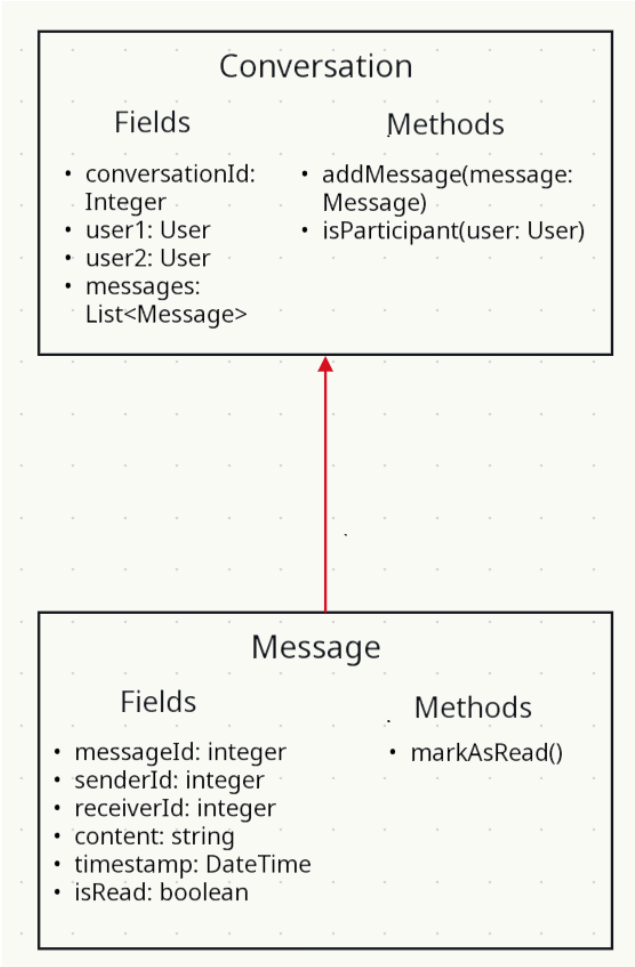
→ “is-a”/inheritance

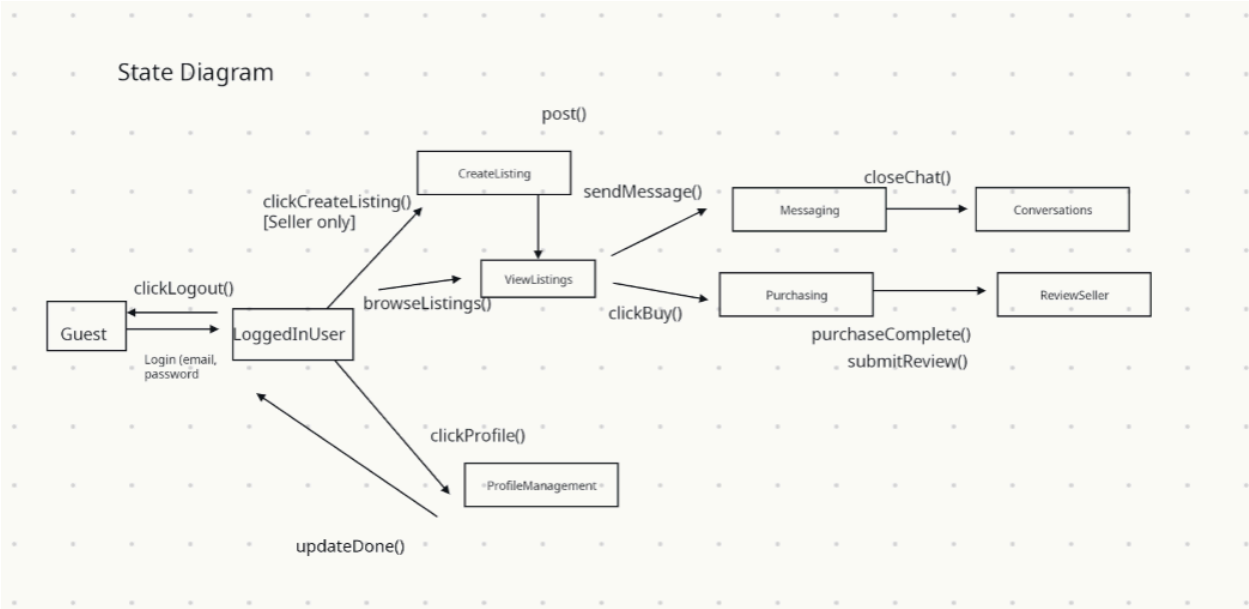
→ “has-a”/composition




User (Abstract)
Fields
userId: Integer
username: String
email: String
passwordHash: String
loggedIn: Boolean
Methods
login(email : String, password: String)
logout()
Guest
An account which can only view listings but does not have other functionality
LoggedUser (Abstract)
Fields
loginTime: LocalDateTime
ratings: Hashset<Rating>
Methods
updateUsername()
updateProfilePicture()
updatePassword()
Buyer
Fields
purchaseHistory: List<productId>
Methods
purchaseListing(productId)
reviewSeller(productId)
Seller
Fields
Listings: List<productId>
salesHistory: List<productId>
totalSales: Integer
Methods
createListing(Product)
updateListing(Integer productId)
deleteListing(Integer productId)


MessagingService
Methods
Fields
None
sendMessage(sender: User, receiver: User, content: String)
getMessages(conversationId: Integer)
markAsRead(messageId: Integer)
CreateOrFetchConvo(user1: User, user2: User)
Explanation
This class handles all messaging logic like sending messages and retrieving conversations. It interacts with the Conversation and Message classes and is used by both Buyer and Seller classes to deal with communication between users. It doesn't inherit from or extend from any of the other classes.


Conversation
Fields
conversationId: Integer
user1: User
user2: User
messages: List<Message>
Methods
addMessage(message: Message)
isParticipant(user: User)
Message
Fields
messageId: integer
senderId: integer
receiverId: integer
content: string
timestamp: DateTime
isRead: boolean
Methods
markAsRead()



Data Structures:

- Set: All MySQL queries are returned in result sets which are used because of their ability to have only unique entries.
- Heap: Used to sort messages that have already been sent, only need to pop the top item to start. Heapsort implementation for sorting.
- Queue: Messages are processed and displayed in the exact order (FIFO - First In, First Out) they were sent.
- Map: Maps were used for the ratings chart generation on the seller page. The number of stars was stored as keys (numbered 1 - 5, int) and the number of ratings that had the star count (long) were stored as the values.
- ArrayList: Listings are stored in an array for simple, indexed access and efficient iteration.

Multi-threading:

- All of our backend code will be multithreaded as all java servlets are inherently multithreaded. The messaging is also multithreaded as we are sending and receiving messages at the same time.

Networking:

- Sending messages, uploading products, and buying products all need networking functionality to work. Users will need to be able to send their messages through the internet in order for other users to receive them. Connections to the database will also require networking as the database is not local to any user's computer.

User Login Functionality:

- As denoted by the class diagram, there are two separate classes for Guests and LoggedUsers. This is decided upon in the login page. Guests have limited functionality

and are restricted to solely viewing listings, but always have the option to register to create an account. LoggedUsers can be either Buyers or Sellers, but all LoggedUsers can create conversations with other LoggedUsers and edit their profile picture and username. Buyers can give ratings to sellers and purchase listings. Sellers can put their listings on the market and make any changes to that listing if they desire – deletion is also a possibility.

Database Schema:

- User table
    - User ID
    - User Name
    - User's password (hashed)
    - Rating
    - User Email
- Product table
    - Product ID
    - Product name
    - Price
    - Product description
    - Thumbnail
    - Reference images
    - SellerID
    - Tags/Categories
- Message table
    - ID
    - Sender ID
    - Message text
    - Timestamp
- Ratings Table
    - Id
    - buyerID
    - sellerID
    - Stars
    - Comment
    - createdAt

Hardware and Software requirements:

Hardware:

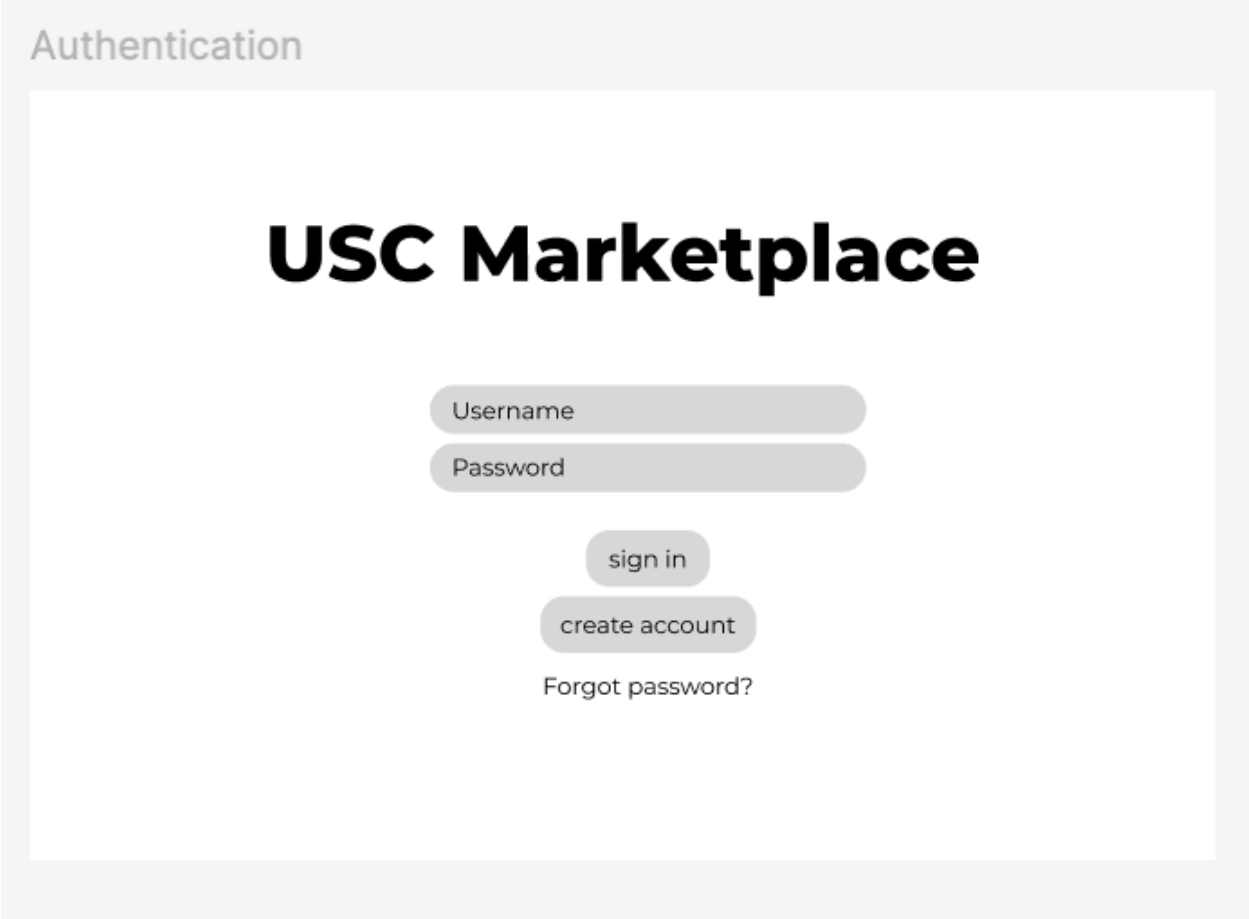
- Server
    - Local Host

Software:

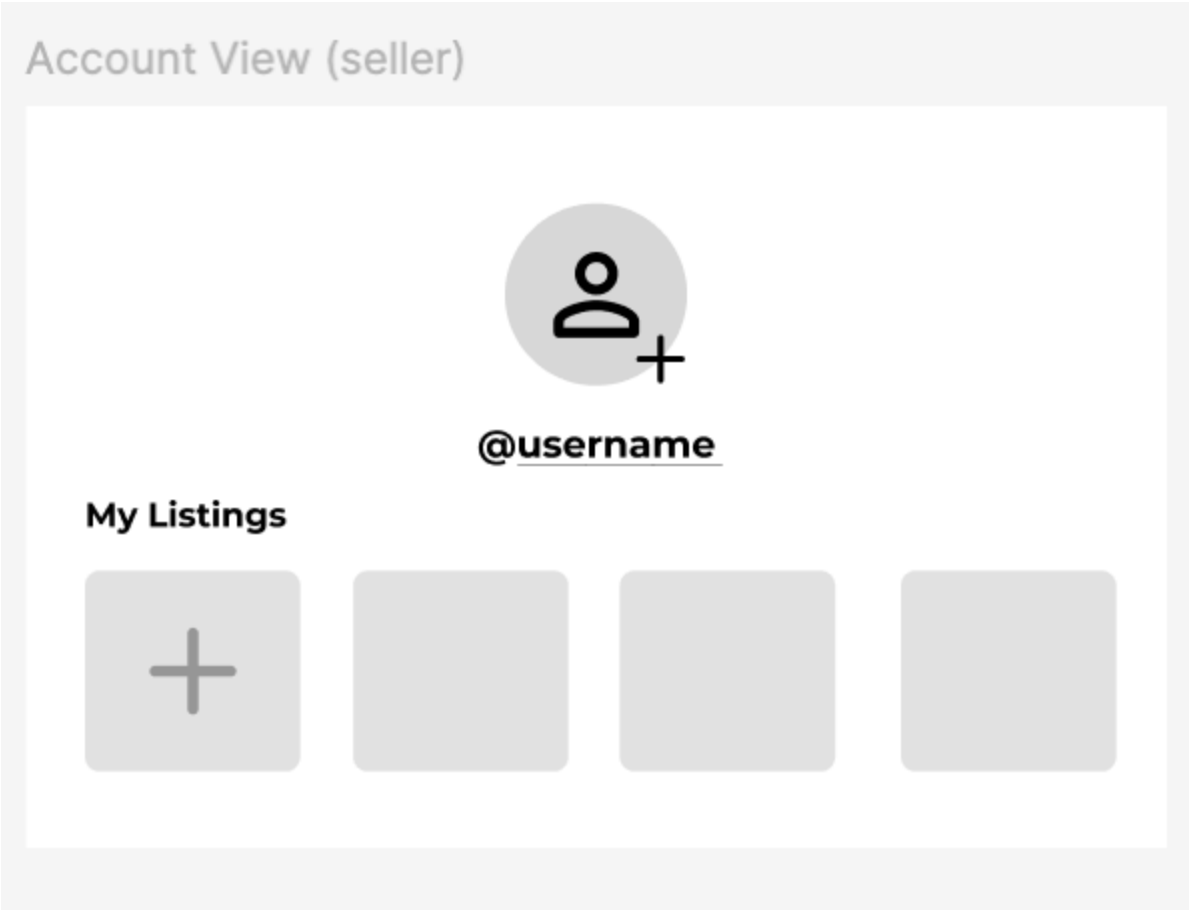
- Front-end
    - Languages: HTML, CSS, Javascript
    - Libraries Needed: React.js, SweetAlerts
    - Frameworks: Bootstrap, Tailwind CSS
- Back-end
    - Language: Java
    - Libraries: JDBC, Spring Boot, Java Mail API, JAF
- Database
    - MySQL
    - Language: SQL
    - DBMS: MySQL Workbench

GUI Design:

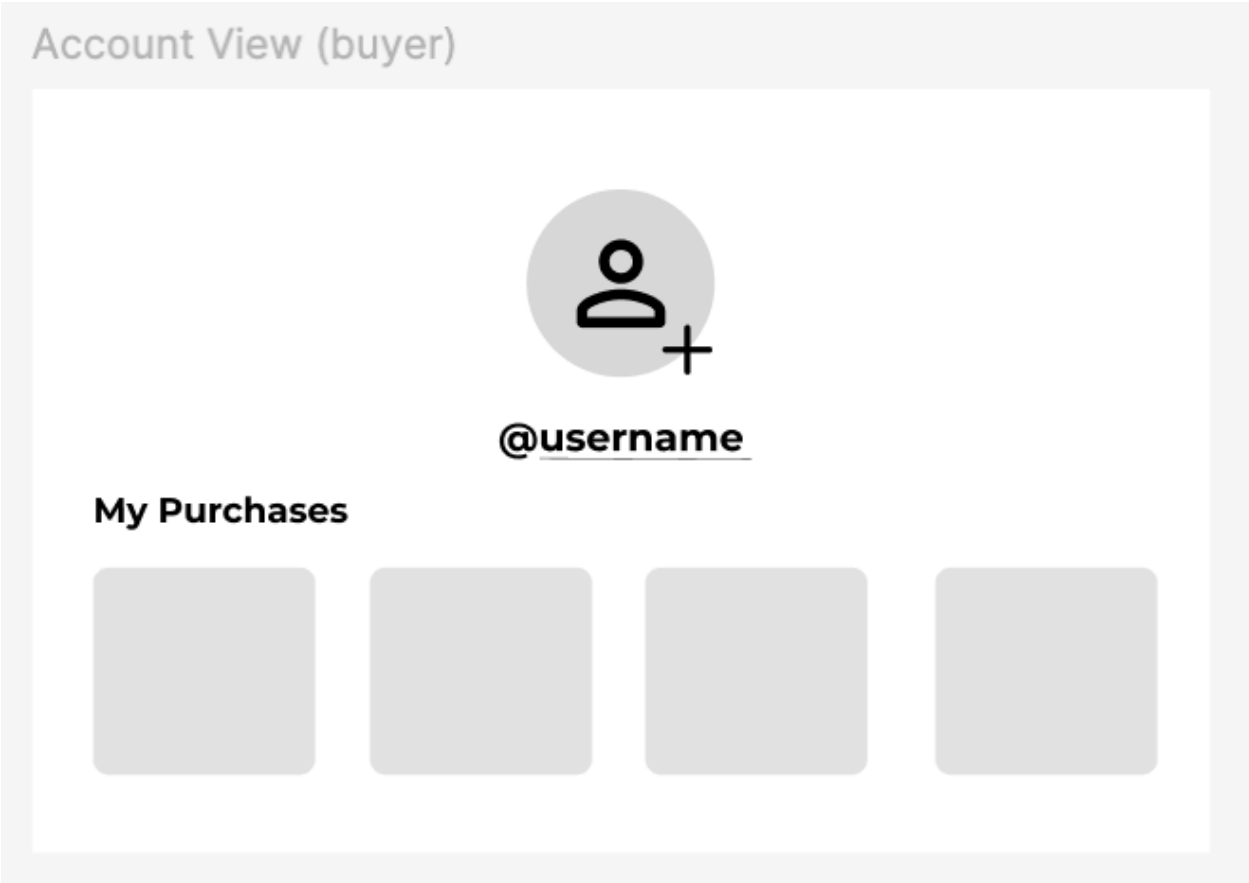
[Figma Design Link]


Authentication
USC Marketplace
Username
Password
sign in
create account
Forgot password?

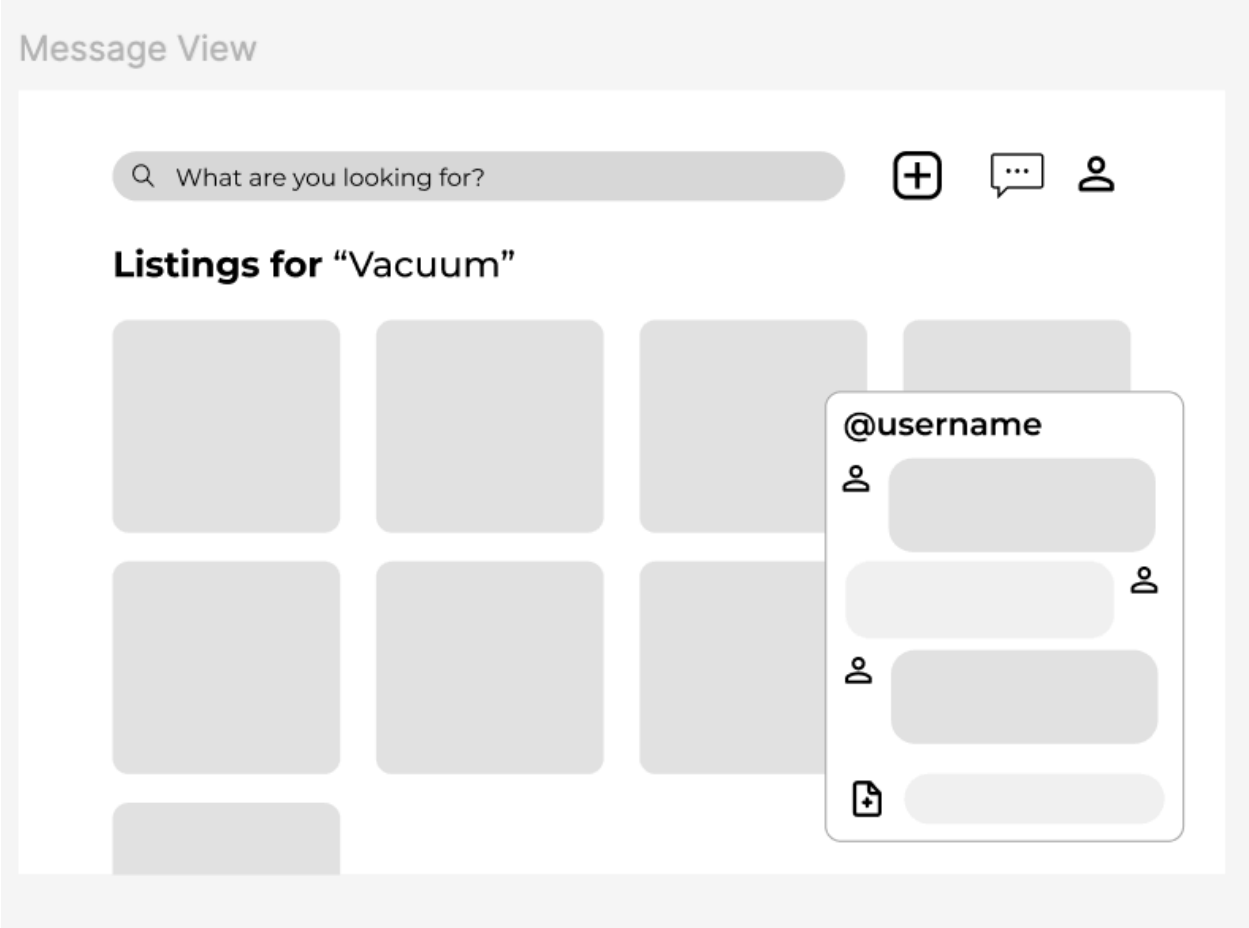
Login Page



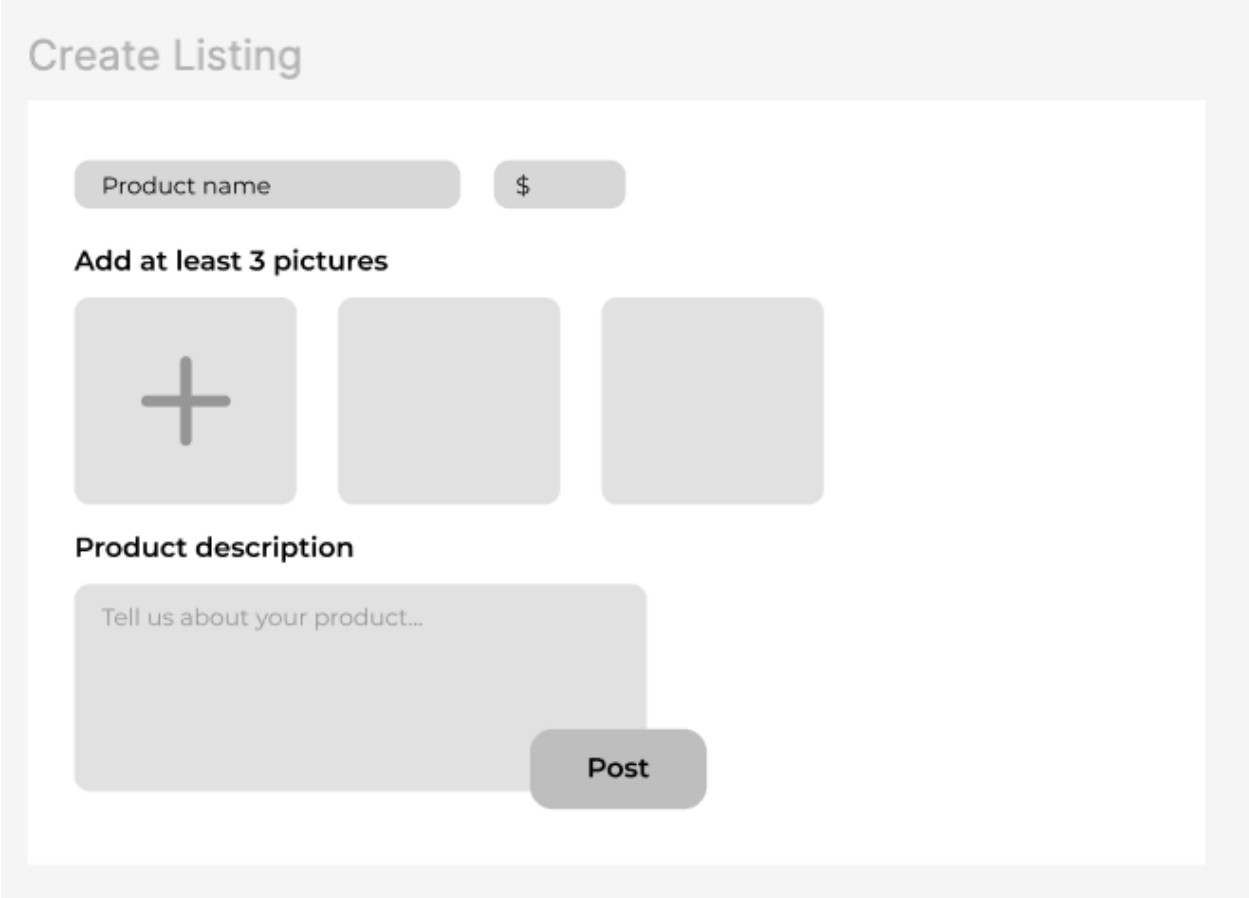
Account page (seller)



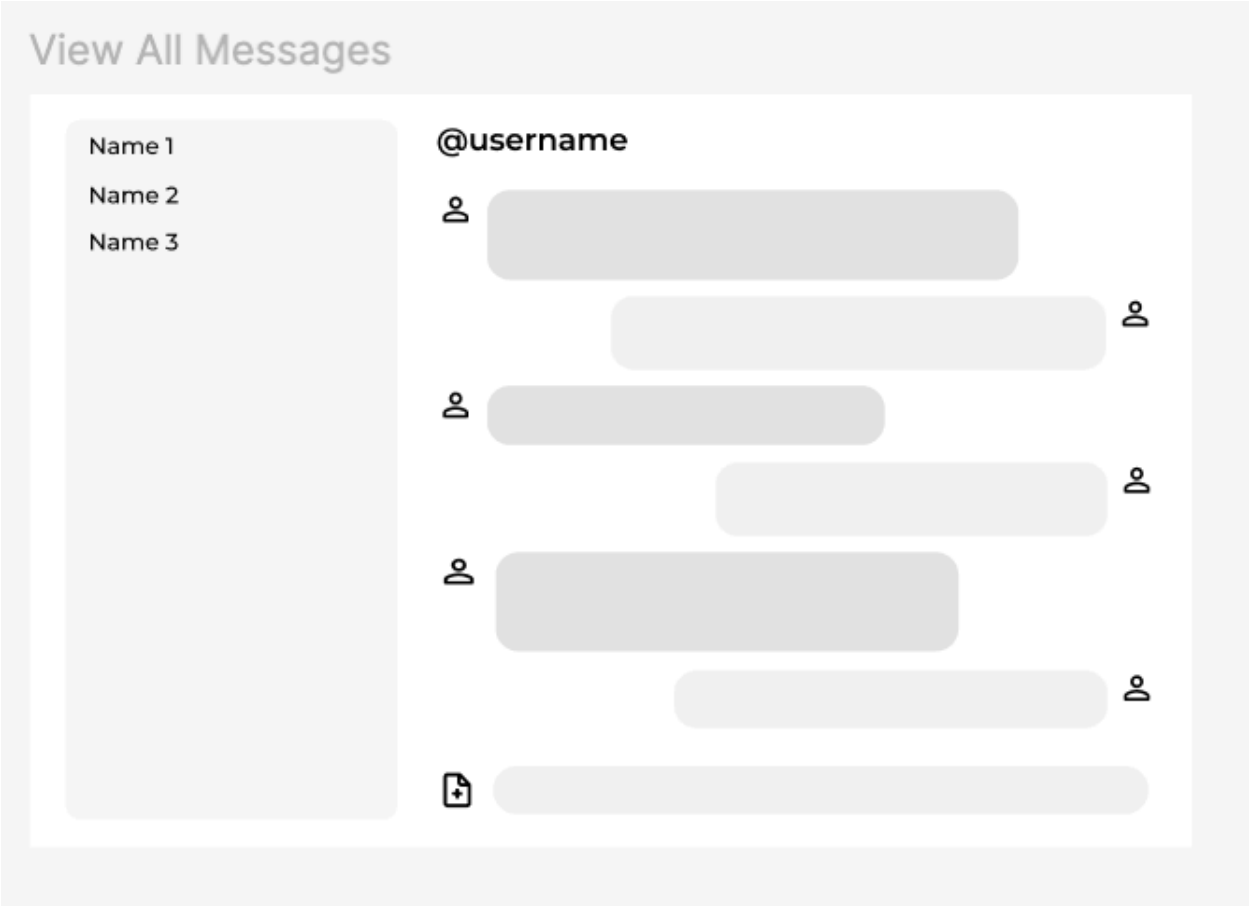
Account page (buyer)


Message View
What are you looking for?
Listings for “Vacuum”
@username

Home Page



Create Listing



Conversations Page

Testing Plan

Feature/Page: Login and Registration

Test Case	Input	Expected Output
Valid Login	Email: admin@gmail.com Password: admin12345	Users should be signed in and presented with the home page. Additionally, the home page views may different depending on whether the user is a seller or a buyer.
Invalid Login	Email: admin@gmail.com Password: 3jrlq3	Users should be alerted that the email and password combination they entered is invalid, i.e. a user with that email and password does not exist within the database.
Guest login	User clicks on the “Browse as guest” button.	Users can only browse listings. Messaging, buying, and posting are restricted.
Invalid Email Format	Email: admin.gmail.com Password: admin12345	Users should be alerted that what they entered is not a valid email format, and should be provided a template for a valid email.
Invalid Registration	Case 1: invalid email format Case 3: Username already exists	Show error: Username taken and must choose a different one.

Feature/Page: Conversation and Messaging

Test Case	Input	Expected Output
- Send valid message - Attempt to send empty	MessageContent: “Is this still available” Sender: “user1”	- The message stored in a database that will instantly appear in the conversation UI

Message (we will prompt the user to type something in before clicking send)	Receiver: “user2”	- The message will be appended to MessagesList - Message is stored in the Message table with proper timestamp and IDs
Upload media	MessageContent: media uploaded by either user (images and videos) Sender: “user1” Receiver: “user2”	The media should be stored in the messages lists between the user and should be visible in the conversation, with both users being able to enlarge and view the media.
Insert link	Sender: “user1” Receiver: “user2” MessageContent: Contains valid URL	- A warning should appear to warn the user that the link is taking them to an external site
Click SendMessage button (displayed on MessageBar) - Attempt to send empty Message (we will prompt the user to type something in before clicking send) - Send valid message	User: “buyer123” Seller: “Seller123” messageContent: “contains the User’s (buyer’s) message”	- If message content is empty, display error: "Please type a message before sending."

Feature/Page: Purchase a product (buyer side)

Test Case	Input	Expected Output
Only able to purchase available products	Attempt to purchase a product with status = "available" vs "sold out”	Purchase should only succeed if the product is available. It should display an error message if the product is out of stock.
Leave a Rating	Rate seller after purchase (ie. ⅘ stars)	Rating saved, reflected in seller profile.
Invalid Rating	Rate Seller with negative value of -1	Should reject rating and ask for another rating.

Feature/Page: Listings (Seller side)

Test Case	Input	Expected Output
Valid price (only numerical)	Only accepts numerical input for price, Strings are rejected. Has a minimum (> 99 cents) and maximum price ($1000).	If the user provides any invalid input, red text appears next to the “price” box explaining what is wrong with the user’s input. The “upload listing” button should be grayed out while the price is invalid.
Valid Listing Details	The listing's name is present. The listing has a description. The listing has a picture. The listing has a price.	There should be no errors shown to the user if all details (including price and images) have been uploaded. If the user forgets to enter a listing name or description, the boxes will be marked red, preventing the seller from publishing the listing.
Picture upload	A picture from the user’s files; a valid JPG or PNG file via fURL	The user will be able to click on one of the placeholder picture buttons to select an image from their files. The user must submit at least one image. Once the image has been chosen, it must replace the gray plus.The user should also be able to click on the image to change it to a different one.
Picture delete	Tapping an “x” in the right corner of the uploaded image.	The user should be able to tap on an “x” in the corner to delete the image, on which the image is replaced by the gray plus once again.
Delete Listing	Delete listing button should be pressed	The respective listing should be removed from the seller’s List of listings, and be removed from the frontend as well. All respective data in the database should also be removed.
“Upload listing” works	Pressing the “upload listing”	The button should be grayed

	button	out until all input is valid. The button should not result in any change even when tapped if the input is invalid. When all the input is valid, the “upload listing” button should become active and when pressed, it should update the listing details to the database and appear in the user’s profile as an active listing. The user should be redirected back to their profile page.

Feature/Page: Logged in User

Test Case	Input	Expected Output
Password without update	Same valid password	Same hash
Username already taken	Taken username	Username is not updated, error message for taken user
empty username update	empty username field	Username is not updated, error message for user
invalid password update	empty password field, invalid format, or same password entered	Password is not updated, error message for user

Feature/Page: Search and Filter

Test Case	Input	Expected Output
Search by title	Search bar: “microwave” Search bar: “MICROWAVE”	Listings with “microwave” in the title are shown (case-insensitive)
Sort by price	Drop down: Price low to high Price high to low	Listings appear in ascending or descending pricings based on whatever the user chooses
No matches	Title: “ZXCV123”	Show “No listings found”

		message

Deployment Document

Step 1: Deploying Back-end

1. Create an instance of a virtual machine on Oracle Cloud
2. Open the terminal or GitHub Desktop
3. git clone cmd
    a. I.e. git clone https://github.com/<your-team-username>/usc-marketplace.git
    cd usc-marketplace
4. Install dependencies via npm install or pip install
5. Set environment variables (.env file) to store sensitive data
6. Run server

Step 2: Deploying Front-End

1. Connect our github repo to Vercel
2. Customize URL to https://usc-marketplace.live/

Step 3: Set up MySQL Database

1. Ensure MySQL is installed up and running
2. Open MySQL Workbench or terminal and create a new database
    a. CREATE DATABASE usc_marketplace
3. Run the scripts from the database folder to set up tables and sample data info
4. Set up and configure database credentials

Step 4: Access the web-app to ensure successful deployment

1. Simply try to determine whether or not the front-end, back-end, and database were successfully deployed and set-up by trying to access the web application.
2. Once the web application is open, verify that the front-end looks as intended.

Step 5: Verify Front-end Back-end Communication

1. Manually enter listings via mysql workbench to see if the front-end displays said listings with the help of the back-end.
2. Perform actions in the front-end and check to see if network requests are sent, and whether or not the back-end appropriately returns information. Test the following:
    a. GET
    b. POST
    c. PUT
    d. DELETE

Step 6: Perform Regression Tests

1. Open browser and navigate to for example http://<server-ip>:3000
2. Using the earlier provided test cases in the testing plan document, test the application to ensure functionality
3. Use developer tools or Postman to ensure API responses are working as intended

Step 7: Notify Users

1. Email or share the live URL / application link with testers or end users
    a. Ie. https://usc-marketplace.live/login