Software Requirements Specification

for

Last Roll

Version 1.0

Prepared by:

Jack Cook	**jwc602**	**jackcook99@tuta.com**
James Harden	**jah1479**	**james.harden.jr@live.com**
Tess Harris	**emh585**	**tessharris413@protonmail.com**
Grayson Smith	**gbs105**	**gbaileysmith@gmail.com**

Instructor: Nisha Pillai
Course: CSE 6214

September 10, 2025

1. Introduction

1.1 Purpose

Last Roll is an online marketplace for tabletop role-playing game dice. Users will be able to buy and sell dice sets and sort listings based on color, size, material, etc. easily through a simple tag filtering system with limited and definite metrics. The goal is to design and implement a creative and immersive user experience that's still simple and safe to use.

This SRS will describe site functionality as a whole. In short, there will be functionality for logging in and creating accounts, creating and deleting listings, buying and selling dice, and seeing past orders and receipts (both incoming and outgoing). An overall description of the product will be in Section 2, a list and description of major system features in Section 3, and further nonfunctional requirements in Section 4.

1.2 Document Conventions

The terms Customer and User (interchangeably) refer to both Buyers and Sellers of dice, while Admin or Administrator refers to accounts with administrative permissions over both and are not included in the class of User.

1.3 Intended Audience and Reading

This document is intended for developers, project managers, and administrative staff. We aim to maintain a seamless user experience that does not require knowledge of software development to understand this document.

An overall description of the product will be in Section 2, a list and description of major system features in Section 3, and further nonfunctional requirements in Section 4. It is recommended for any project managers, developers, and documentation writers to read the document in its entirety and use it as reference during the lifespan of the project. Any marketing staff, potential users, and testers should not need to read beyond Section 2: Overall Description.

1.4 Product Scope

The site will be able to host buy and sell orders for dice based on tagged specifications. This will facilitate distribution of dice to buyers and the processing of transactions resulting from this. The website owner will take (roughly) a 10% commission from each order placed on the website, and the site itself will keep track of seller payouts, order receipts, and fulfillment, creating an easy platform for new and potential sellers.

1.5 References

We will be referencing articles from this guide toward making our project accessible to all users: https://consciousstyleguide.com/ability-disability/.

2. Overall Description

2.1 Product Perspective

Last Roll is a self-contained e-commerce marketplace that allows users to buy and sell TTRPG dice. Buyers are provided with the means to add unique and interesting dice to their collections. Sellers are provided with a centralized location to show and sell their products to an interested audience. This website will be managed by our new company (Last Roll, LLC), which will serve as facilitator and moderator of transactions. While sites exist to purchase TTRPG dice, no site provides a marketplace with a focus on collaboration and user-to-user transactions. With this in mind, it should be stated that Last Roll does not replace an existing system.

The website's core functionality is distributed to several different software components:

- The frontend faces the userbase, and is accessible through standard web browsers. "Buyer" users can view and purchase listings posted by "seller" users. Sellers can manage their listings via a seller dashboard. Administrators can monitor site usage through an administrative dashboard.
- The backend application server organizes user profiles, processes orders, and updates listing information.
- The database subsystem stores user and listing information, such as a user's credentials or a listing's properties.

A few external services provide full functionality:

- The payment gateway allows buyers to securely pay for their items, and sellers to securely receive the profit from sale of their products.
- The shipping API generates shipping labels for sellers to use and allows buyers and sellers to track the shipping progress and delivery of their order..
- The email notification service provides updates to users, such as shipment notifications or updates to a particular listing. It additionally enables a second layer of authentication, to be used for security functions and password resets.
- An administrative dashboard allows user accounts and listings to be managed by Last Roll staff.

2.2 Product Functions

Buyer Functions

All users are "buyers" by default. Therefore, all users will be able to:

- Create and manage their user account.
- Browse for listings.
- Search for a listing based on descriptive tags, such as "Blue", "Resin", or "D20".
- View listing details, such as description, price, and availability.
- Add items to a "shopping cart" and update cart as needed.
- Place orders and complete checkout.
- Make secure payments via a payment gateway.
- View their order history.
- Track packages.
- Receive notifications through email.

Seller Functions

Users approved by Last Roll administrators to sell items will be given access to a seller dashboard. Here, users will be able to:

- Create listings, including a product photo and the appropriate descriptive tag.
- Modify or delete listings.
- View, update, fulfill, or cancel orders.
- Securely deposit their earnings to a connected bank account.

Administrator Functions
Last Roll administrators will be able to:

- Manage registered users.
- Approve Seller Accounts
- Remove listings (to prevent spam and fraud).
- Manage and edit tags on a product
- Monitor buyer, seller, and admin activity.

System Functions
The system will maintain confidentiality and accuracy by:

- Securely authenticating registered users.
- Updating listing database.
- Integrating with shipping APIs for label generation and tracking.
- Integrating with external payment systems for item purchases and seller deposits.
- Sending automated email notifications.
- Ensuring data backup and recovery.
- Enforcing security measures such as encryption and access control.

2.3 User Classes and Characteristics

Buyer *(Primary)*

- Role: purchases products, manages personal account.
- Frequency: occasional to frequent.
- Functions Used: account creation, login/logout, browsing catalog, cart management, checkout, payment, order tracking.
- Technical Expertise: basic Internet and shopping familiarity.
- Privilege Level: access to only their account data and order history.
- Importance: High

Seller *(Primary)*

- Role: sells products, manages personal account.
- Frequency: occasional to frequent.
- Functions Used: account creation, login/logout, listing creation, order management, deposit, shipping label generation.
- Technical Expertise: basic Internet and selling familiarity.
- Privilege Level: access to only their listings and outstanding orders.
- Importance: High

Guest *(Secondary)*

- Role: browses catalog without logging in.
- Frequency: occasional.
- Functions Used: view products.
- Technical Expertise: Minimal to none.

- Privilege Level: None
- Importance: Medium (helps attract new customers).

Administrator *(Primary)*
- Role: manages user accounts and listings.
- Frequency: daily.
- Functions Used: user account management, listing management.
- Technical Expertise: requires training.
- Privilege Level: full access to user accounts and listings.
- Importance: Critical

2.4 Operating Environment

The website's servers and services will run on an on-premises Ubuntu 24.04 LTS server (IP: 70.250.151.1) with sufficient CPU, memory, and storage to handle expected traffic. The HTTP server will use the Apache framework. The backend will be written in Python/Django, with a MySQL relational database system. The frontend will be primarily written in HTML/CSS, with some Java as deemed necessary for accessibility and aesthetics.

The payment gateway will emulate PayPal. The shipping API will emulate Easypost. Email notifications will be handled through SMTP.

End users will access the system through personal devices, including desktop computers, laptops, tablets, and smart phones. Although the user's device must be capable of running a modern Internet browser, the application is platform-independent.

2.5 Design and Implementation Constraints

We have not secured permission to operate yet, which limits our abilities to fully integrate the external services. We will emulate PayPal and Easypost's APIs to streamline deployment of these services. For now, users will be able load their accounts via gift certificates.

The development team's previous experience necessitates the use of specific programming environments to ensure a quality product is delivered on time. Therefore, no changes will be made to the framework unless absolutely necessary.

Our company is self-hosting the site, and we will be responsible for maintaining the software.

3. System Features

3.1 Buyer Account Features

The following features will be available for all user accounts. Every account is a buyer account by default.

3.1.1 Create a new buyer account. (HIGH)

1. User navigates to the top right of the “Home” page and clicks to open the navigation submenu.
2. In the submenu, user clicks the “Create Account” button.
3. Application server loads the “Account Creation” page.
4. User enters a valid desired username into the “Username” textbox.
    - 4.1. Valid username contains only capital and lowercase letters and digits 0-9.
        - 4.1.1. If user enters invalid username, an error message is displayed: “Username can only contain letters and numbers.”
    - 4.2. Application server queries database to check if desired username is already registered.
        - 4.2.1. If username is unique, a green check is displayed.
        - 4.2.2. If username is already registered, a red X is displayed with the error message “Username taken. Please choose another username.”
5. User enters a valid desired password into the “Password” text box.
    - 5.1. The password text is redacted.
    - 5.2. Valid password is at least 8 characters long and contains both a capital letter and a number.
        - 5.2.1. If user enters a password with an invalid length, an error message is displayed: “Password must be at least 8 characters long.”
        - 5.2.2. If user enters a password without a capital letter and a number, an error message is displayed: “Password must contain a capital letter and a number.”
6. User enters a valid email address into the “Email Address” text box.
    - 6.1. Valid email address follows convention [a-zA-z0-9]@[a-zA-z0-9].[a-z]
        - 6.1.1. If email address is invalid, an error message is displayed: “Invalid email address.”
7. User enters the same valid email address into the “Confirm Email Address” text box.
    - 7.1. If email address does not match the email address in the “Email Address” text box, an error message is displayed: “Email addresses do not match.”
8. User clicks the “Sign Up” button.
    - 8.1. Application server sends account credentials to an Account Staging database, where they are temporarily stored (~72hrs) until email confirmation.
    - 8.2. The application server generates and hosts a unique confirmation link.
    - 8.3. The email API sends a message containing the confirmation link to the entered email address.
    - 8.4. A message is displayed: “Please confirm your account through the link sent to your email within 72 hours.”
9. The user navigates to their personal email account, opens the email sent by the email API, and clicks the confirmation link.
    - 9.1. The application server records the date and time the account was verified.

9.2. The verification date and time is appended to the account's entry in the Account Staging database.
9.3. The application server moves the full account credentials to the Accounts database.
9.4. A message is displayed: "Thank you for confirming your account."
10. The application server redirects the user to the "Log In" page.

3.1.2 Log into an existing account. (HIGH)

1. User has an existing account (via 3.1.1).
2. User navigates to the top right of the "Home" page and clicks to open the navigation submenu.
3. User clicks the "Log In" button.
4. Application server loads the "Log In" page.
5. User enters their username into the "Username" field.
6. User enters their password into the "Password" field.
   6.1. The password text is redacted.
7. User clicks the "Log in" button.
8. Application server queries the Accounts database and checks if the provided password matches the stored password for the provided username.
   8.1. If passwords match, user is redirected to the "Home" page in the "logged in" state.
   8.2. If passwords do not match, an error message is displayed: "Invalid credentials."

3.1.3 Log out. (HIGH)

1. User is logged in (via 3.1.2).
2. User navigates to the top right of the "Home" page and clicks to open the navigation submenu.
3. User clicks the "Log Out" button.
4. Application server sets the user's status in the user database to "logged out".
5. A message is displayed: "You have been logged out successfully."
   5.1. The user is redirected to the "Home" page in the "logged out" state.

3.1.4 View account details. (MEDIUM)

1. User is logged in (via 3.1.2).
2. User navigates to the top right of the "Home" page and clicks to open the navigation submenu.
3. In the submenu, user clicks the "My Account" button.
4. Application server loads the "My Account" page for the user (i.e, in its "state").
5. The user's username, date of account creation, and outstanding orders are displayed.

3.1.5 Delete account. (MEDIUM)

1. User views account details (via 3.1.4).
2. User clicks the "Delete Account" link.
3. Application server loads the "Delete Account" page.
4. User enters their password into the "Password" field.
   4.1. The password text is redacted.
5. User clicks the agreement check box, which is displayed with a message: "I agree to permanently delete my account, and I understand this action is irreversible."
6. User clicks the "Delete my account" button.

6.1. If agreement check box is not selected, an error message is displayed: “Please agree to the statement.”
6.2. Application server queries Accounts database.
6.2.1. If entered password does not match stored password, an error message is displayed: “Password is incorrect.”
6.2.2. If passwords match, application server removes user account information from Accounts database.
7. A message is displayed in a pop-up window: “Your account has been deleted successfully.”
8. User is redirected to “Home” page in the “logged out” state.

3.1.6 Browse listings. (HIGH)

1. User is logged in (via 3.1.2).
2. User clicks the “Shop” button on the “Home” page.
3. Application server loads the “Shop” page in the “recent listings” state.
3.1. Application server queries the Listings database and retrieves the 20 most recently published listings.
3.2. The listings are displayed in card format.

3.1.7 View listing. (HIGH)

1. User views “Shop” page (via 3.1.6).
2. User clicks on a item’s image.
3. The application server loads the “Product” page for the desired item (i.e, in its “state”).
3.1. The application server queries the Listings database for the item’s full data.
3.2. The product image is retrieved from cache.
4. The item’s details are displayed.

3.1.8 Add listing to cart. (HIGH)

1. User views item listing (via 3.1.7).
2. User enters the desired quantity in the “Quantity” field.
3. User clicks the “Add to Cart” button.
4. The application server adds the product ID to the user’s shopping cart vector.
4.1. The shopping cart vector persists client-side via cookies.
5. A message is displayed: “Item added successfully.”

3.1.9 View cart. (HIGH)

1. User is logged in (via 3.1.2).
2. User navigates to the top right of the “Home” page and clicks the “Cart” button.
3. Application server uses shopping cart vector to query Listings database.
3.1. Application server loads the “Cart” page for the specified items (i.e, in its “state”).
3.2. Cart contents, quantity, and subtotal are displayed.

3.1.1- Remove item from cart. (HIGH)

1. User views their cart (via 3.1.8).
2. User clicks the “Remove from Cart” button next to the desired item.
3. Application server modifies user’s shopping cart vector to reflect changes.
3.1. Application server reloads “Cart” page according to new shopping cart vector.

3.1.11 Check out. (HIGH)

1. User views their cart (via 3.1.8).
2. User clicks the “Check Out” button.
3. Application server loads the “Check Out” page.
   - 3.1. Application server uses shopping cart vector to query Listings database.
   - 3.2. Cart contents are displayed.
   - 3.3. Total cost of cart contents, plus appropriate tax, is calculated and displayed.
   - 3.4. Application server queries the Accounts database for saved address information.
     - 3.4.1. If user has a saved address, information is auto-filled into the appropriate fields.
4. User enters their first name in the “First Name” field.
5. User enters their last name in the “Last Name” field.
6. User enters their telephone number in the “Phone Number” field.
7. User enters the first line of a valid shipping address in the “Shipping Address Line 1” field.
   - 7.1. A valid shipping address begins with a number 0-9.
   - 7.2. User enters second line in shipping address in the “Shipping Address Line 2” field, if necessary.
   - 7.3. If user provides an invalid shipping address, or clicks away from field without entering an address, an error message is displayed: “Must provide a valid shipping address.”
8. User enters a valid city in the “Shipping Address City” field.
   - 8.1. A valid city contains only capital and lowercase letters.
   - 8.2. If user provides an invalid city, an error message is displayed: “Must provide a valid city.”
9. User chooses a state from the “Shipping Address State” dropdown menu.
10. If user’s shipping address is their billing address, user clicks “Use shipping address as billing address” checkbox.
    - 10.1. Otherwise, user enters their billing address into the “Billing Address Street Line 1”, “Billing Address Street Line 2” Billing Address City”, and “Billing Address State” fields as described in 3.1.7-3.1.9.
11. If desired, user checks “Save My Address Information” box.
    - 11.1. Upon submission, the application server hashes the address information and sends it to the Accounts database.
12. User enters their credit card number in the “Credit Card Number” field.
13. User enters a valid first name in the “First Name On Card” field.
    - 13.1. A valid name contains only capital and lowercase letters.
    - 13.2. If user provides invalid name, an error message is displayed: “Must provide a valid name.”
14. User enters a valid last name in the “Last Name On Card” field.
    - 14.1. A valid name contains only capital and lowercase letters.
    - 14.2. If user provides invalid name, an error message is displayed: “Must provide a valid name.”
15. User enters three-digit security code in the “Security Code” field.
    - 15.1. If security code is invalid, an error message is displayed: “Must provide a valid security code.”
16. User clicks the “Check Out” button.
17. If “Save Address Information” button is checked, application server hashes user address and adds it to the Accounts database.
18. Application server removes item listings from Listings database.
19. Application server queries payment gateway with appropriate payment information.

20. Application server adds order to Orders database.
21. Application server loads “Thank you for your order” page.
22. Email API sends purchase notification email to buyer.
23. Email API sends sale notification email to seller.

3.1.12 Sort listings based on tags. (MEDIUM)

1. User browses listings (via 3.1.6).
2. User selects the desired descriptive tag from the “Tags” list on the sidebar.
3. The application server refreshes the “Shop” page with matching items (i.e, in its “state”).
    3.1. The application queries the Listings database for the 20 most recently listed items matching the descriptive tag.
    3.2. Additional tags can be selected, which queries the Listings database for the 20 most recently listed items matching all desired tags.

3.1.13 View order history. (MEDIUM)

1. User views their account details (via 3.1.4).
2. User clicks the “Order History” button.
3. Application server loads the “Order History” page for the user (i.e, in its “state”).
    3.1. Application server queries the Orders database for all orders placed by that user.

3.1.14 View tracking information. (MEDIUM)

1. User views their order history (via 3.1.12).
2. User clicks the “Tracking” button next to the desired order.
3. Application server redirects user to the tracking link.
    3.1. Application server queries the shipping API for the tracking link.

3.1.15 Request seller account. (HIGH)

1. User views their account details (via 3.1.4).
2. User clicks the “Request Seller Account” button.
3. Application server queries Accounts database and modifies the user’s account to the “Seller Pending” state.
4. A message is displayed: “An administrator will approve your seller account request.”

3.2 Seller Account Features

Seller accounts inherit all features described in 3.1.

3.2.1 View seller dashboard. (HIGH)

1. User is logged in (via 3.1.2).
2. User navigates to the top right of the “Home” page and clicks to open the navigation submenu.
3. In the submenu, user clicks the “Seller Dashboard” button.
4. The application server loads the “Seller Dashboard” page for the user (i.e, in its “state”).
5. A summary of the seller’s recent sales is displayed.

3.2.2 View listings. (HIGH)

1. User views seller dashboard (via 3.2.1).
2. Users clicks the “My Listings” button.
3. Application server loads the “My Listings” page for the user (i.e, in its “state”).
    3.1. Application server queries the Listings database for the user’s listings.
    3.2. Application server queries the Listing Staging database for the user’s pending listings.
    3.3. The user’s active and pending listings are displayed.

3.2.3 Create listing. (HIGH)

1. User views listings (via 3.2.2).
2. User clicks the “Create Listing” button.
3. Application server loads the “Create Listing” page.
4. User enters the product name in the “Name” field.
5. User enters the product description in the “Description” field.
6. User chooses the appropriate dice type tag from the “Dice Type” dropdown menu.
    6.1. Sample dice types include “D20”, “D4”, “D6”, “D100”, or “Complete Set.”
7. User chooses the appropriate material tag from the “Material” dropdown menu.
    7.1. Sample materials include “wood”, “metal”, “resin”, or “plastic”.
8. User chooses the appropriate descriptive tag from the “Color” dropdown menu.
    8.1. Sample colors include “holographic purple”, “green”, “pewter”, or “natural”.
9. User enters a valid price in the “Price” field.
    9.1. A valid price includes numbers 0-9 and .
10. User clicks the “Upload Photo” button.
    10.1. User’s file explorer is opened.
    10.2. When user submits the file, it is cached.
    10.3. A thumbnail of the photo is displayed.
11. User clicks the “Confirm” button.
    11.1. Application server moves listing information to Listing Staging database.
    11.2. A message is displayed: “Your listing will be posted once it has been approved by an administrator.”
12. Application server redirects user to the “My Listings” page.

3.2.4 Modify listing. (MEDIUM)

1. User views their listings (via 3.2.2).
2. User clicks the “Edit Listing” button next to the desired listing.
3. Application server loads the “Create Listing” page in the “edit” state.
    3.1. Application server queries the Listings database and autofills the fields with the appropriate information.
4. User modifies the listing information (according to rules in 3.2.3.4-3.2.4.10).
5. User clicks the “Confirm” button.
    5.1. Application server stages listing for approval (according to process in 3.2.3.11).
6. Application server redirects user to the “My Listings” page.

3.2.5 Delete listing. (HIGH)

1. User views their listings (via 3.2.2).
2. User clicks the “Delete Listing” button next to the desired listing.
3. A message is displayed in a popup window: “Are you sure you would like to delete this listing?”

3.1. If user clicks the “No” button, the window is dismissed with no effect.
3.2. If user clicks the “Yes” button, the application server queries the Listings database to remove the listing.
3.2.1. A message is displayed: “Listing deleted successfully.”

3.2.6 Add deposit method. (HIGH)

1. User views the seller dashboard (via 3.2.1).
2. User clicks the “Add Deposit Method” button.
3. The application server queries the payment gateway’s OAuth link.
4. The application server redirects the user to the payment gateway’s OAuth page.

3.2.7 View order history. (HIGH)

1. User views the seller dashboard (via 3.2.1).
2. User clicks the “Order History” button.
3. Application server loads the “Seller Order History” page for the user (i.e, in its “state”).
   3.1. Application server queries the Orders database for all orders owned by the user.

3.2.8 View unfulfilled orders. (HIGH)

1. User views their order history (via 3.2.7).
2. User clicks the “Show Only Unfulfilled Orders” button.
3. Application server queries the Orders database for all orders owned by the user with the “unfulfilled” tag.
4. Application server reloads the “Seller Order History” page for the user (i.e, in its “state”).

3.2.9 Print shipping label for order. (LOW)

1. User views unfulfilled orders (via 3.2.8).
2. User clicks the “Print Label” button next to the desired order.
3. The application server calls the shipping API.
   3.1. The shipping API generates a label and returns the file as a PDF to the application server.
   3.2. The application server opens the label in the browser’s print preview window.

3.2.10 Mark order as shipped. (HIGH)

1. User views unfulfilled orders (via 3.2.8).
2. User clicks the “Mark As Shipped” button next to the desired order.
3. The application server queries the Orders database and removes the “unfulfilled” tag from the order.

3.2.11 View sales information. (LOW)

1. User views seller dashboard (via 3.2.1).
2. User clicks the “Sales Summary” button.
3. Application server loads the “Sales Summary” page for the user (i.e, in its “state”)
   3.1. Application server queries the Orders database for orders owned by the user.
   3.2. Application server aggregates statistics.
   3.3. Statistics are displayed.

3.3 Administrator Account Features

Administrator accounts are created as needed from an existing administrator account. The initial administrator account's credentials will be manually loaded into the Accounts database.

3.3.1 Log in. (HIGH)

1. User navigates to the dedicated admin portal URL.
2. Application server loads the "Admin Log In" page.
3. User enters their username into the "Username" field.
4. User enters their password into the "Password" field.
    4.1. The password text is redacted.
5. User clicks the "Log In" button.
6. Application server queries Accounts database and checks if the provided password matches the stored password for the provided username.
    6.1. If passwords match, user is redirected to the "Admin Dashboard" page.
    6.2. If passwords do not match, an error message is displayed: "Invalid credentials."

3.3.2 Log out. (HIGH)

1. User is logged in (via 3.3.1).
2. User navigates to the top right of the "Admin Dashboard" page and clicks the "Log Out" button.
3. Application server sets the user's status in the user database to "logged out".
4. A message is displayed: "You have been logged out successfully."
    a. The user is redirected to the "Admin Log In" page in the "logged out" state.

3.3.3 View admin dashboard. (HIGH)

1. User is logged in (via 3.3.1).
2. User navigates to the top right of the "Home" page and clicks the "Admin Dashboard" button.
3. Application server loads the "Admin Dashboard" page.

3.3.4 Create new administrator account. (MEDIUM)

1. User views admin dashboard (via 3.3.3)
2. User clicks the "Create Admin Account" button.
3. Application server loads the "Create Admin Account" page.
4. User enters a valid desired username into the "Username" textbox.
    a. Valid username contains only capital and lowercase letters and digits 0-9.
        i. If user enters invalid username, an error message is displayed: "Username can only contain letters and numbers."
    b. Application server queries database to check if desired username is already registered.
        i. If username is unique, a green check is displayed.
        ii. If username is already registered, a red X is displayed with the error message "Username taken. Please choose another username."
5. User enters a valid desired password into the "Password" text box.
    a. Valid password is at least 8 characters long and contains both a capital letter and a number.

        i. If user enters a password with an invalid length, an error message is displayed: “Password must be at least 8 characters long.”
        ii. If user enters a password without a capital letter and a number, an error message is displayed: “Password must contain a capital letter and a number.”
6. User enters a valid email address into the “Email Address” text box.
    a. Valid email address follows convention [a-zA-z0-9]@[a-zA-z0-9].[a-z]
        i. If email address is invalid, an error message is displayed: “Invalid email address.”
7. User enters the same valid email address into the “Confirm Email Address” text box.
    a. If email address does not match the email address entered above, an error message is displayed: “Email addresses do not match.”
8. User clicks the “Sign Up” button.
    a. Application server sends account credentials to an Account Staging database, where they are temporarily stored (~72hrs) until email confirmation.
    b. The application server generates and hosts a unique confirmation link.
    c. The email API sends a message containing the confirmation link to the entered email address.
    d. A message is displayed: “Please confirm your account through the link sent to your email within 72 hours.”
9. The user navigates to their personal email account, opens the email sent by the email API, and clicks the confirmation link.
    a. The application server records the date and time the account was verified.
    b. The verification date and time is appended to the account’s entry in the Account Staging database.
    c. The application server moves the full account credentials to the Accounts database.
    d. A message is displayed: “Thank you for confirming your account.”
10. The user is redirected to the “Admin Dashboard” page.

3.3.5 Remove a listing. (MEDIUM)

1. User views admin dashboard (via 3.3.3).
2. User clicks the “Delete A Listing” button.
3. Application server loads the “Delete Listings” page.
4. User enters listing ID in the “Listing ID” field.
5. User clicks the “Confirm” button.
6. Application server queries Listings database and removes the listing with the matching ID.
7. The user is redirected to the “Admin Dashboard” page.

3.3.6 Remove user accounts. (MEDIUM)

1. User views admin dashboard (via 3.3.3).
2. User clicks the “Delete A User Account” button.
3. Application server loads the “Delete User Accounts” page.
4. User enters username in the “Username” field.
5. User clicks the “Confirm” button.
6. Application server queries Accounts database and removes the account with the matching username.
7. The user is redirected to the “Admin Dashboard” page.

3.3.7 Approve new seller account.

1. User views admin dashboard (via 3.3.3).
2. User clicks the “Verify Sellers” button.
3. Application server loads the “Verify Sellers” page.
    3.1. Application server queries Accounts database for all accounts with “Seller Pending” status.
    3.2. All account usernames are displayed.
4. User clicks the “Verify” button next to the desired username.
5. Application server queries Accounts database and changes user status from “Seller Pending” to “Seller”.

4. Other Nonfunctional Requirements

4.1 Performance Requirements

To ensure a satisfactory buyer and seller experience, we must adhere to specific performance requirements.

4.1.1 The website will be robust to moderate traffic (~100 concurrent users) without experiencing outages.

4.1.2 Should the website experience too much traffic, it will mitigate user frustration through use of graceful redirects.

4.1.3 All pages will load in less than 15 seconds on a 25 Mbps wireless connection.

4.1.4 Moderation actions, such as banning a user or removing a listing, will take no longer than 1 minute to occur.

4.1.5 Customers will be notified via email of maintenance periods.

4.2 Safety Requirements

4.2.1 Users cannot purchase something they cannot afford.

4.2.2 Sellers should have a confirm listing button with the listing price displayed.

4.2.3 Admins should have a way to confirm listings before they are published to make sure they’re appropriate.

4.2.4 Admins should be able to flag and remove/approve listings.

4.2.5 Admins should be able to remove reviews if they violate rules.

4.3 Security Requirements

4.3.1 The Website should be tested to make sure there aren't any bypasses.

4.2.2 Inputs should be sanitized.

4.2.3 Too many invalid attempts at logging into an account should cause a temporary pause on the user’s ability to log in.

4.2.4 The website needs to have other account protections in place so that accounts are safe.

4.4 Software Quality Attributes

- **Availability:** We want the platform to be available at all times.
- **Flexibility:** We want to make sure the platform can handle all sorts of situations.
- **Maintainability:** We want to make sure that once complete the website will be easy to maintain.
- **Reliability:** We want the website to be functional at all times.
- **Useability:** We want to make sure the website is easy to use since a variety of people are going to be using it.

Appendix A: Glossary

TTRPG	tabletop role playing game