Software Requirements Specification

Gardener’s Exchange Mobile Application

Team Struggle Bus

Chris Arpin

Brandon George

Adam Hursey

Noah Lindsey

Tim Ours

Mitch Whedon

Table of Contents

[Introduction 3](#_Toc500688773)

[Glossary 4](#_Toc500688774)

[User Roles 5](#_Toc500688775)

[Collaborating Systems/Software 6](#_Toc500688776)

[Schedule 7](#_Toc500688777)

[Functional System Requirements 8](#_Toc500688778)

[System Response Table 8](#_Toc500688779)

[Use Cases 11](#_Toc500688780)

[Listing 11](#_Toc500688781)

[Customer 21](#_Toc500688782)

[Trading 25](#_Toc500688783)

[Seller 28](#_Toc500688784)

[Cross-Functional Map 36](#_Toc500688785)

[Logical Data Model 37](#_Toc500688786)

[Logical Process Model 38](#_Toc500688787)

[Data Dictionary 56](#_Toc500688788)

[User Experience Diagrams 58](#_Toc500688789)

[General 58](#_Toc500688790)

[Customer 59](#_Toc500688791)

[Sell 60](#_Toc500688792)

[Wireframes 61](#_Toc500688793)

[Non-Functional User Requirements 85](#_Toc500688794)

[Usability 85](#_Toc500688795)

[Accessibility 85](#_Toc500688796)

[Availability 85](#_Toc500688797)

[Documentation and Training 85](#_Toc500688798)

[Non-Functional System Requirements 86](#_Toc500688799)

[Performance 86](#_Toc500688800)

[Capacity 86](#_Toc500688801)

[Security 86](#_Toc500688802)

[Longevity 87](#_Toc500688803)

## Introduction

This document serves as the description of the functional and nonfunctional requirements of the Gardner’s Exchange mobile phone application. This document will outline all of the designs necessary to assist in the development of the application by Taylor University students.

The main functions of the application will be to provide a platform for local peoples to buy and sell home-grown goods and to develop gardening communities.

## Glossary

**Associative Table** - A table in a database needed to connect tables that have a many-to-many relationship like students and teachers.

**Database** - An organized collection of data.

**Entity** - A table in a database.

**FK, Foreign Key** - A column in a table that is used as a unique identifier for a row in another table.

**Functional System Requirements** - A series of documents that describe the requirements of the behavior of the system itself.

**GB** - Gigabytes

**GET Request** - A method for requesting data from a web source.

**Home Feed** - The main page in the application where all of the listings are located.

**Listing** - A certain produce that a user has put up for sale.

**Non-Functional System Requirements** - A series of documents that describe the requirements of the operation of the system itself.

**Order** - An instance of a user buying one or more items.

**PK, Primary Key** - A unique identifier for a row in a table.

**PUT Request** - A method for sending data to a web source.

**SQL** - A database querying language used for retrieving, updating, and removing data from a database as well as developing the database itself.

**SQL Injection** - A security attack where unsanitized user input is exploited to run malicious SQL statements.

**Trade** - An instance of a user trading one or more items for another set of items.

**Wireframe** - A mockup design of a specific page in an application.

## User Roles

These are the users who will interact with the system and will be referred to throughout this document:

**User**

A user is any users of the system. Users have many states including guest, customer, and seller.

**Unregistered User/Guests**

An unregistered user is anyone using the system that is not logged in. Unregistered users can only view items.

**Customer**

A customer is the state of a user that is explicitly looking to purchase produce.

**Seller**

A seller is the state of a user that is explicitly looking to sell produce.

## Collaborating Systems/Software

* PayPal Express Checkout
  + This would be the integrated payment system that will use PayPal systems to send the payment from Buyer to the Seller.
* Amazon Web Services(AWS) S3
  + This system will host all of the servers that will be needed hold the user content and serve up the content to the many users.

## Schedule

|  |  |
| --- | --- |
| **Date** | **Milestone** |
| January 30th | Start product development |
| February 15th | Backend complete, app development begin |
| March 15th | External systems integrated |
| March 30th | Start testing |
| April 20th | Polish product |
| May 5th | Documentation and preparation for app release |
| May 17th | App released on App Store |

# Functional System Requirements

## System Response Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event ID** | **Source** | **Trigger** | **Response** | **Major Outputs** | **External Destinations** |
| L1 | Guest | Enters the app | Shows a list of items within 25 miles | Home feed | None |
| L2 | Guest | Presses the account button | Shows login page | Login page | None |
| L3 | User | Login to user account | Validation from the server | 1. User logged in  2. Error | None |
| L4 | Guest | Selects to create an account | Creates a form to allowed to enter user information | 1. New account sent into the home feed  2. Error | Project database |
| L5 | Guest | Submitting the create account form | Sent to the server for validation | 1. User added to the database and sent to the home feed  2. Error | Project database |
| L6 | User | Selects to delete an account | Server asks if customer is certain that they want their account information removed | Updates account status to “inactive” | Project database |
| L7 | User | Updating user account | Updates the information in the user profile | 1. Update account information  2. Error | Project database |
| L8 | User | Pressing the Distance Button | Pop up box where customer can edit the distance that the app looks for items | Changes searching distance to the set amount | None |
| L9 | User | Get user update account form | Displays a page with fillable fields to update account information | Account update form | None |
| L10 | User | Get Purchase History Page | Displays a page with previous orders and trades | Purchases Page | None |
| C1 | Customer | Enters a quantity | Calculates price | Price | None |
| C2 | Customer | Buys items | Adds an order to the database and notifies the seller | Order data, Email to seller, Home Feed | Project database, seller email |
| C3 | User | Presses listing tab | User sent to the home feed | Home feed | None |
| C4 | User | Views the details of a listing | Generates page with more information about the item | Item page | None |
| T1 | Customer | Offers Trade | Adds a trade to the database, notifies seller | Trade data, email to seller | Project database, seller email |
| T2 | Seller | Accepted Trade | Update trade data to close trade and email offering customer | Trade data, email to offering customer | Project database, offering customer email |
| T3 | Seller | Cancel Trade Offer | Update the status of the trade to ‘declined’ | The trade is canceled | Project database |
| S1 | Seller | Lists an item for sale | Adds new listing to database | Listing data | Project database |
| S2 | Seller | Wants to create a new post | Goes to a new post form | Displays the new post form | None |
| S3 | Seller | Remove listing | Sets listing to inactive | Listing data | Project database |
| S4 | Seller | Clicks button to view trade offers | Sent to trade offers page | Trade offers page | None |
| S5 | Seller | Clicks on button to view account listings | Sent to account listings page | Account listings page | None |
| S6 | Seller | Clicks to edit listing | Sent to edit listing page | Edit listing page | None |
| S7 | Seller | Confirms edits to listing | Edits listing in database | Listing data | Project database |

## Use Cases

### Listing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L1 | | | |
| NAME | Enters the App | | | |
| Primary Actor | Guest | | | |
| Other Actors | None | | | |
| Description | Guest enters app without a login | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Guest opens app on smartphone | |  | |
|  | | 2. System opens app to the home feed | |
| Alternate Event Flows | 1. Guest opens app on smartphone | |  | |
|  | | 2. System crashes on opening the app | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Enters the App | Preconditions | App installed on smartphone |
| After event | Conclusion | Shows listing of items within 25 miles | Postconditions | Display the home feed |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L2 | | | |
| NAME | Presses Account Button | | | |
| Primary Actor | Guest | | | |
| Other Actors | None | | | |
| Description | Presses Account Button to access login page | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Guest presses the account button | |  | |
|  | | 2. System produces a login page | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Presses the account button | Preconditions | Application running, user not logged in |
| After event | Conclusion | Shows login page | Postconditions | Display login page |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L3 | | | |
| NAME | Login | | | |
| Primary Actor | User | | | |
| Other Actors | None | | | |
| Description | User attempts to log in with valid information | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. User enters correct login information | |  | |
|  | | 2. System validates login information against database | |
|  | | 3. System logs user in | |
| Alternate Event Flows | 1. User enters incorrect login information | |  | |
|  | | 2. System validates login information against database | |
|  | | 3. System does not log user in | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Login to user account | Preconditions | Application running, user not logged in |
| After event | Conclusion | Validation from the server | Postconditions | User logged in |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L4 | | | |
| NAME | Create Account | | | |
| Primary Actor | Guest | | | |
| Other Actors | None | | | |
| Description | User selects to create an account | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Guest selects Account tab | |  | |
|  | | 2. System generates Create Account page | |
|  | | 3. Returns page | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Selects to create an account | Preconditions | User is not logged in |
| After event | Conclusion | Creates a form allowed to enter user information | Postconditions | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L5 | | | |
| NAME | Submit Create Account Form | | | |
| Primary Actor | Guest | | | |
| Other Actors | None | | | |
| Description | User submits filled out account form | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Guest submits Create Account form | |  | |
|  | | 2. System verifies account information, no duplicate account name | |
|  | | 3. New account created in database | |
| Alternate Event Flows | 1. Guest submits Create Account form | |  | |
|  | | 2. System verifies account information, finds duplicate account name | |
|  | | 3. System return duplicate account error | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Submitting the create account form | Preconditions | Guest has no account |
| After event | Conclusion | Sent to the server for validation | Postconditions | New user registered in database |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L6 | | | |
| NAME | Delete Account | | | |
| Primary Actor | User | | | |
| Other Actors | None | | | |
| Description | User selects to delete account | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. User selects to delete their account | |  | |
|  | | 2. System asks to confirm account deletion | |
| 3. User confirms account deletion | |  | |
|  | | 4. System set account status to “inactive” | |
| Alternate Event Flows | 1. User selects to delete their account | |  | |
|  | | 2. System asks to confirm account deletion | |
| 3. User does not confirm account deletion | |  | |
|  | | 4. System does not set account to “inactive” | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Selects to delete account | Preconditions | User logged in |
| After event | Conclusion | Server asks if customer is certain that they want their account information removed | Postconditions | User account set to “inactive” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L7 | | | |
| NAME | Update Account | | | |
| Primary Actor | User | | | |
| Other Actors | None | | | |
| Description | User selects to update account | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. User presses Update Account button | |  | |
| 2. User fills out new account information | |  | |
|  | | 3. System verifies updated account information | |
|  | | 4. System updates account information in the database | |
| Alternate Event Flows | 1. User presses Update Account button | |  | |
| 2. User fills out new account information | |  | |
|  | | 3. System verifies updated account information | |
|  | | 4. System detects duplicate account name, returns error | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Updating user account | Preconditions | User logged in |
| After event | Conclusion | Updates the information in the user profile | Postconditions | Updated account information |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L8 | | | |
| NAME | Search by distance | | | |
| Primary Actor | Guest | | | |
| Other Actors | None | | | |
| Description | User changes the distance parameter | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. User presses the Distance button | |  | |
| 2. User enters a new distance | |  | |
|  | | 3. System refreshes Home Feed with new listings | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Pressing the Distance button | Preconditions | Application running |
| After event | Conclusion | Pop up box where user can edit the distance that the app looks for items | Postconditions | Home feed updated with new distance parameter |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L9 | | | |
| NAME | Update user account | | | |
| Primary Actor | User | | | |
| Other Actors | None | | | |
| Description | User requests to update their account | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. User presses Update Account button | |  | |
|  | | 2. System generates Update Account page | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Get user update account form | Preconditions | User logged in |
| After event | Conclusion | Displays a page with fillable fields to update account information | Postconditions | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | L10 | | | |
| NAME | View purchase history | | | |
| Primary Actor | User | | | |
| Other Actors | None | | | |
| Description | User views transaction history | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. User presses Purchase History button | |  | |
|  | | 2. System generates Purchase History page | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Viewing purchase history | Preconditions | User logged in |
| After event | Conclusion | Shows a list of previous purchases | Postconditions | None |

### Customer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | C1 | | | |
| NAME | Enter quantity | | | |
| Primary Actor | Customer | | | |
| Other Actors | None | | | |
| Description | Customer enters a quantity for an item | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Customer presses item listing for details | |  | |
| 2. Customer enters quantity for item | |  | |
|  | | 3. System calculates price for items | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Enters a quantity | Preconditions | Customer logged in |
| After event | Conclusion | Calculates price | Postconditions | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | C2 | | | |
| NAME | Buying items | | | |
| Primary Actor | Customer | | | |
| Other Actors | None | | | |
| Description | Customer buys the items | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Customer presses the Buy button | |  | |
|  | | 2. System asks for confirmation | |
| 3. Customer confirms buy order | |  | |
|  | | 4. System create a new order entry in database | |
|  | | 5. System notifies seller of buy order | |
| Alternate Event Flows | 1. Customer presses the Buy button | |  | |
|  | | 2. System asks for confirmation | |
| 3. Customer cancels buy order | |  | |
|  | | 4. System loads item details page | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Buys items | Preconditions | Customer logged in |
| After event | Conclusion | Adds an order to the database and notifies the seller | Postconditions | New order entry in database |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | C3 | | | |
| NAME | Home Feed | | | |
| Primary Actor | User | | | |
| Other Actors | None | | | |
| Description | Customer selects to return to Home Feed | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Customer presses the Home Feed tab button | |  | |
|  | | 2. System loads the Home Feed | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Presses Home Feed tab | Preconditions | Customer logged in |
| After event | Conclusion | Customer sent to the home feed | Postconditions | Home Feed displayed |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | C4 | | | |
| NAME | View listing details | | | |
| Primary Actor | Customer | | | |
| Other Actors | None | | | |
| Description | Customer views the details of a particular listing | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Customer presses on a listing | |  | |
|  | | 2. System loads the item details | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Views the details of a listing | Preconditions | Customer logged in |
| After event | Conclusion | Generates page with more information about the item | Postconditions | Item details displayed |

### Trading

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | T1 | | | |
| NAME | Offer a trade | | | |
| Primary Actor | Customer | | | |
| Other Actors | Seller | | | |
| Description | Customer offers to trade their item for seller’s item | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Customer presses the Trade button | |  | |
| 2. Customer enters items to trade | |  | |
|  | | 3. System confirms trade offer, adds trade entry to database | |
|  | | 4. System notifies seller of trade offer | |
| Alternate Event Flows | 1. Customer presses the Trade button | |  | |
| 2. Customer enters items to trade | |  | |
| 3. Customer cancels trade offer | |  | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Offers trade | Preconditions | Customer logged in |
| After event | Conclusion | Adds a trade to the database | Postconditions | Trade entry added to database |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | T2 | | | |
| NAME | Accept trade | | | |
| Primary Actor | Seller | | | |
| Other Actors | Customer | | | |
| Description | Seller accepts Customer’s trade offer | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller accepts a trade offer | |  | |
|  | | 2. System confirms trade entry | |
|  | | 3. System marks trade entry as completed | |
|  | | 4. System notifies Customer of payment information | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Accepted trade | Preconditions | Seller logged in |
| After event | Conclusion | Update trade data to close trade and email offering customer | Postconditions | Trade entry completed |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | T3 | | | |
| NAME | Decline trade | | | |
| Primary Actor | Seller | | | |
| Other Actors | Customer | | | |
| Description | Seller declines Customer’s trade offer | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller declines a trade offer | |  | |
|  | | 2. System marks trade entry as declined | |
|  | | 3. System notifies Customer trade decline | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Declined Trade Offer | Preconditions | Seller logged in |
| After event | Conclusion | Update the status of the trade to ‘declined’ | Postconditions | Trade entry set to declined |

### Seller

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | S1 | | | |
| NAME | List an item | | | |
| Primary Actor | Seller | | | |
| Other Actors | None | | | |
| Description | Seller posts an item for sale | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller submits form with item information | |  | |
|  | | 2. System adds item to database | |
|  | | 3. System generates Home Feed | |
|  | | 4. Returns to Home Feed | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Lists an item for sale | Preconditions | Seller logged in |
| After event | Conclusion | Adds new listing to database | Postconditions | Item listed |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | S2 | | | |
| NAME | Change listing sale time | | | |
| Primary Actor | Seller | | | |
| Other Actors | None | | | |
| Description | Seller changes the time that the listing is for sale | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller presses the view/edit button and selects a post to edit | |  | |
|  | | 2. Post details are displayed in editable form fields | |
| 3. Seller presses the change time button | |  | |
|  | | 4. Popup with a change time widget shows up | |
| 5. The seller puts the time and presses the update item button | |  | |
|  | | 6. Time is changed in the project database | |
| Alternate Event Flows | 1. Seller presses the view/edit button and selects a post to edit | |  | |
|  | | 2. Post details are displayed in editable form fields | |
| 3. Seller presses the change time button | |  | |
|  | | 4. Popup with a change time widget shows up | |
| 5. The seller puts the time and presses the update item button | |  | |
|  | | 6. The time is invalid and an error is displayed to the seller | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Changes time that listing is for sale | Preconditions | Logged in and on posts listings |
| After event | Conclusion | Updates expiration date of listing | Postconditions | Time is updated |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | S3 | | | |
| NAME | Removing a listing | | | |
| Primary Actor | Seller | | | |
| Other Actors | None | | | |
| Description | Seller removes a listing | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller presses the view/edit post button and selects a post | |  | |
|  | | 2. The posting details are listed to the screen | |
| 3. Seller presses the delete item button | |  | |
|  | | 4. Confirmation pops up | |
| 5. Seller presses Ok button | |  | |
|  | | 6. Listing is marked as inactive in the project database | |
| Alternate Event Flows | 1. Seller presses the view/edit post button and selects a post | |  | |
|  | | 2. The posting details are listed to the screen | |
| 3. Seller presses the delete item button | |  | |
|  | | 4. Confirmation pops up | |
| 5. Seller presses Cancel button | |  | |
|  | | 6. Returned to the listing details page | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Remove listing | Preconditions | Logged in and on postings tab |
| After event | Conclusion | Sets listing to inactive | Postconditions | Listing is marked as inactive |

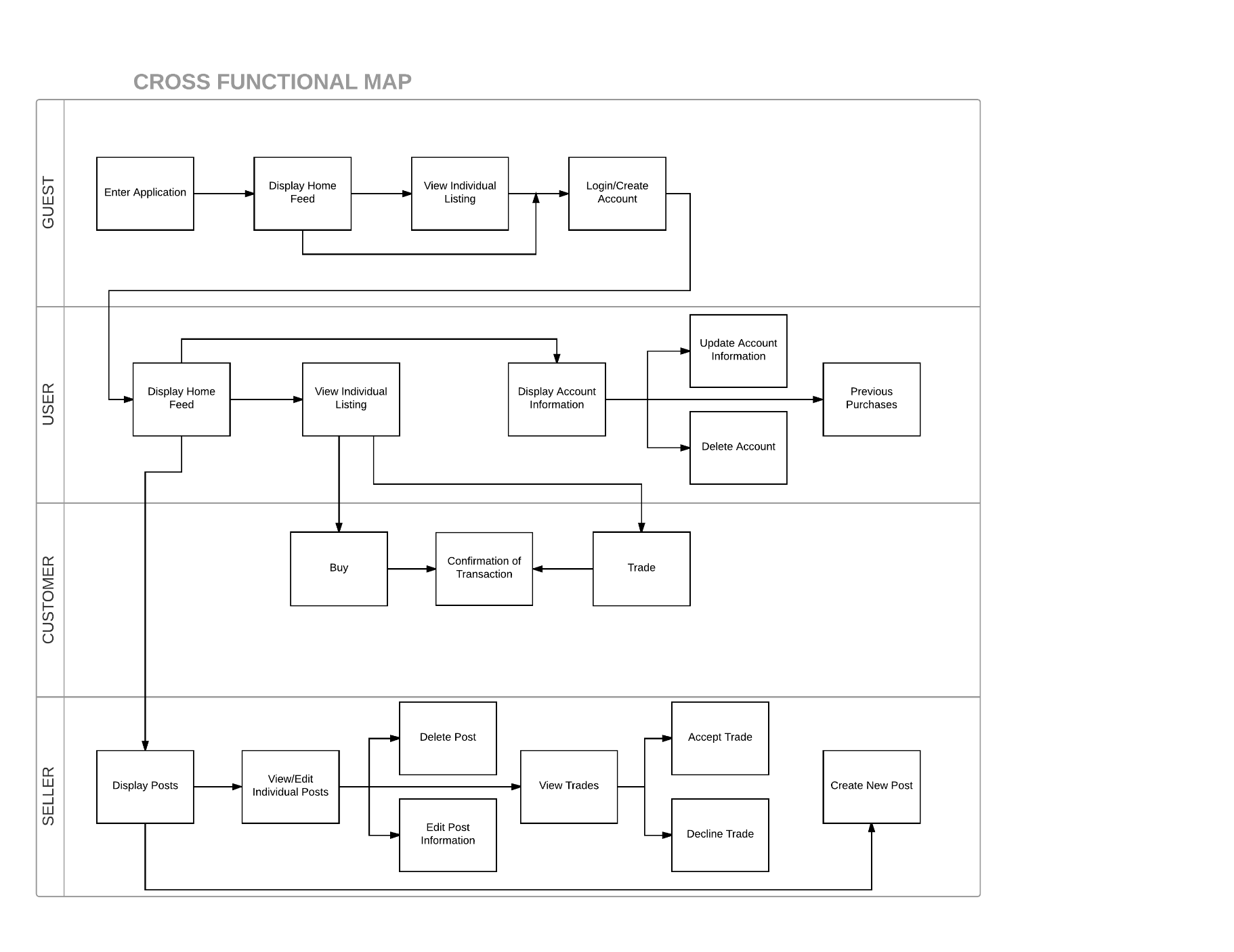
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | S4 | | | |
| NAME | View trade offers | | | |
| Primary Actor | Seller | | | |
| Other Actors | None | | | |
| Description | Seller views current trade offers from customers | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller presses the Post tab | |  | |
|  | | 2. System loads the Post page | |
| 3. Seller presses the View Trades button | |  | |
|  | | 4. System loads trade offers | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Clicks button to view trade offers | Preconditions | Seller logged in |
| After event | Conclusion | Sent to trade offers page | Postconditions | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | S5 | | | |
| NAME | Viewing account listings | | | |
| Primary Actor | Seller | | | |
| Other Actors | None | | | |
| Description | Seller views all of their listings | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller presses the Post tab | |  | |
|  | | 2. System loads the Post page | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Clicks on button to view account listings | Preconditions | Seller logged in |
| After event | Conclusion | Sent to account listings page | Postconditions | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | S6 | | | |
| NAME | Edit listing | | | |
| Primary Actor | Seller | | | |
| Other Actors | None | | | |
| Description | Seller edits the listing details | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller selects Edit on their listing | |  | |
|  | | 2. System generates Edit Listing form | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Clicks to edit listing | Preconditions | Seller logged in |
| After event | Conclusion | Sent to edit listing page | Postconditions | None |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | S7 | | | |
| NAME | Confirm edit to listing | | | |
| Primary Actor | Seller | | | |
| Other Actors | None | | | |
| Description | Seller confirms their edits to a listing | | | |
| Actor Action | | | System Response | |
| Typical Event Flow | 1. Seller edits their listing | |  | |
|  | | 2. System asks for confirmation | |
| 3. Seller confirms changes to listing | |  | |
|  | | 4. System updates item entry in database | |
| Alternate Event Flows | 1. Seller edits their listing | |  | |
|  | | 2. System asks for confirmation | |
| 3. Seller cancels edits to listing | |  | |
|  | | 4. System cancels changes | |
| Business Focus | | | System Focus | |
| Before event | Trigger | Confirms edits to listing | Preconditions | Seller logged in |
| After event | Conclusion | Edits listing in database | Postconditions | Updated item entry in database |

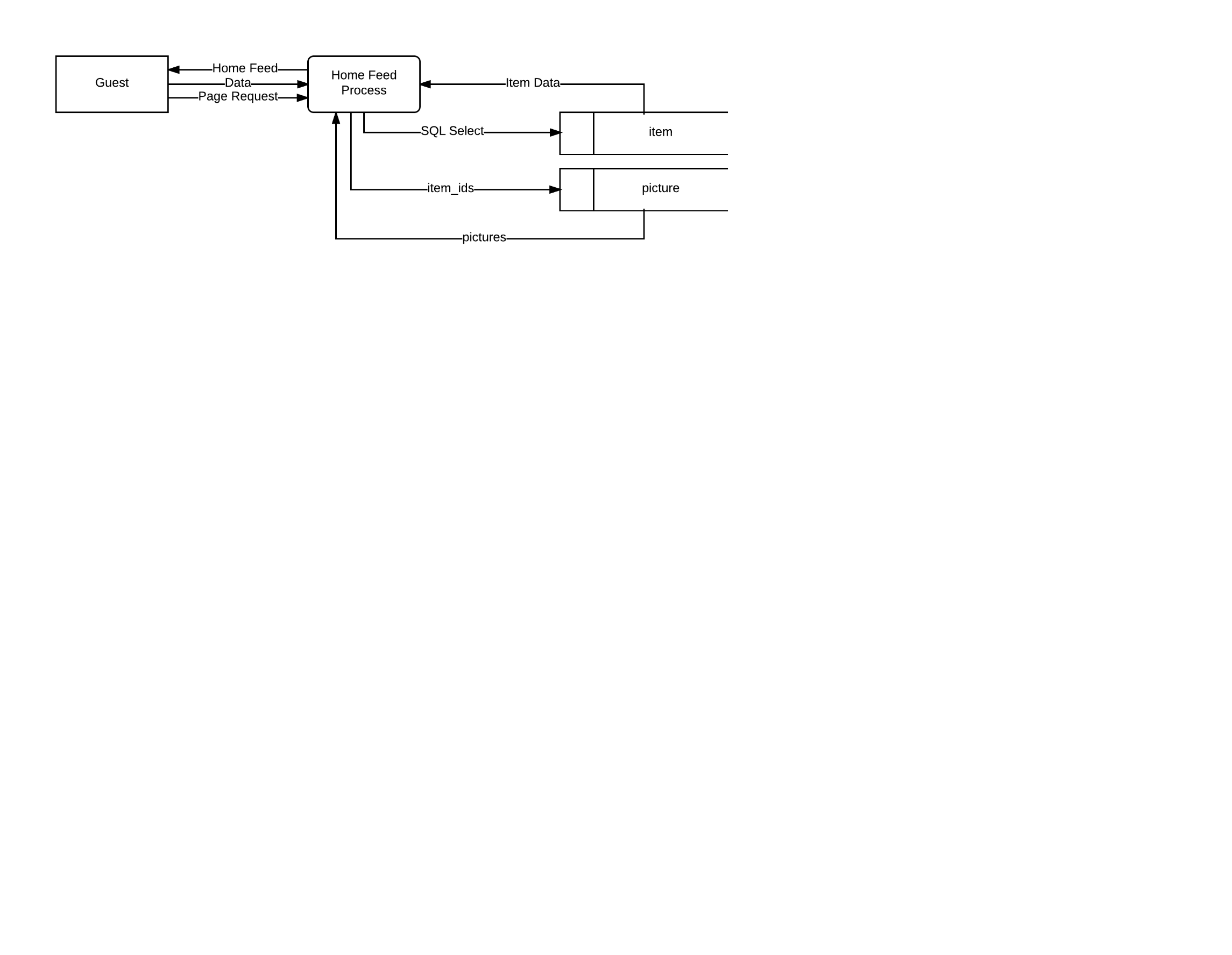
## Cross-Functional Map



## Logical Data Model

## Logical Process Model

**L1 - Entering the App**



Data = distance, {user\_id}

SQL Select = status, {user\_id}

Item Data = item\_id, title, price, quantity

#Enter App

Request Home Feed

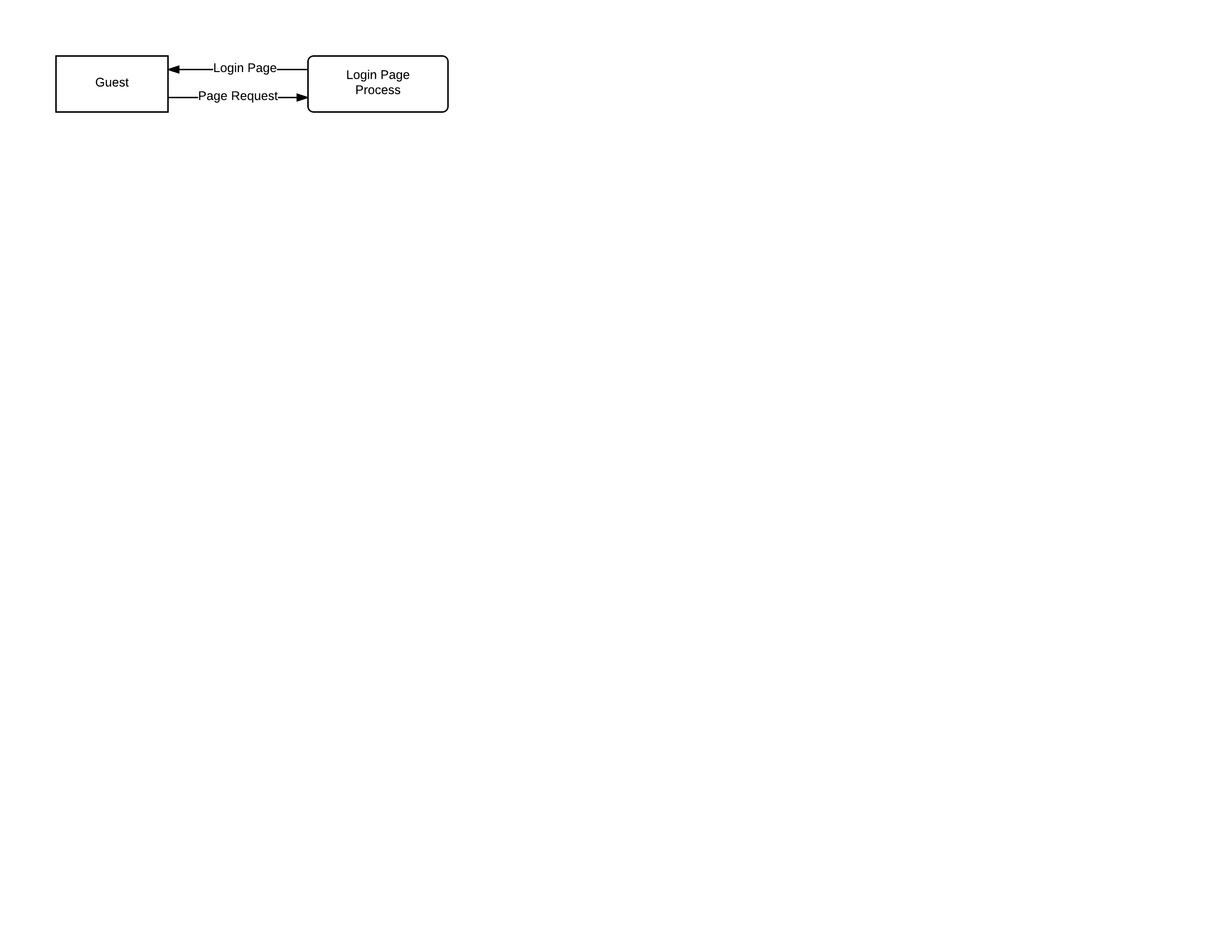
Get items within distance

Get item pictures

Generate page

Return page

**L2 - Viewing Login Page**

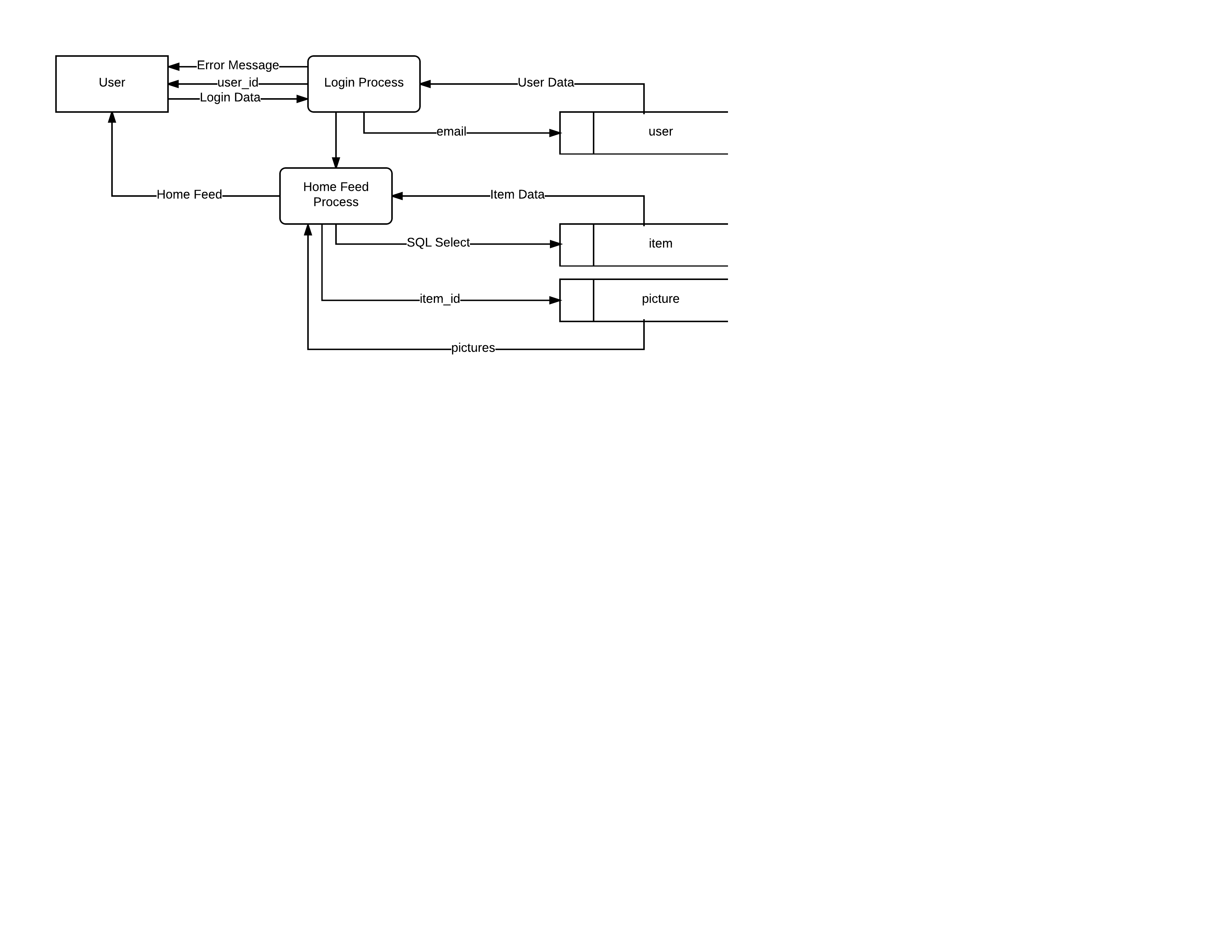
****

#Login Page

Request Login Page

Return Login Page

**L3 - Logging In**



Login Data = email, password

User Data = user\_id, password\_hash

Item Data = item\_id, title, price, quantity

SQL Select = user\_id, status

#Logging in

Get login information

Get user\_id and password for compare

if(info is valid):

Log user in

Generate Home Feed

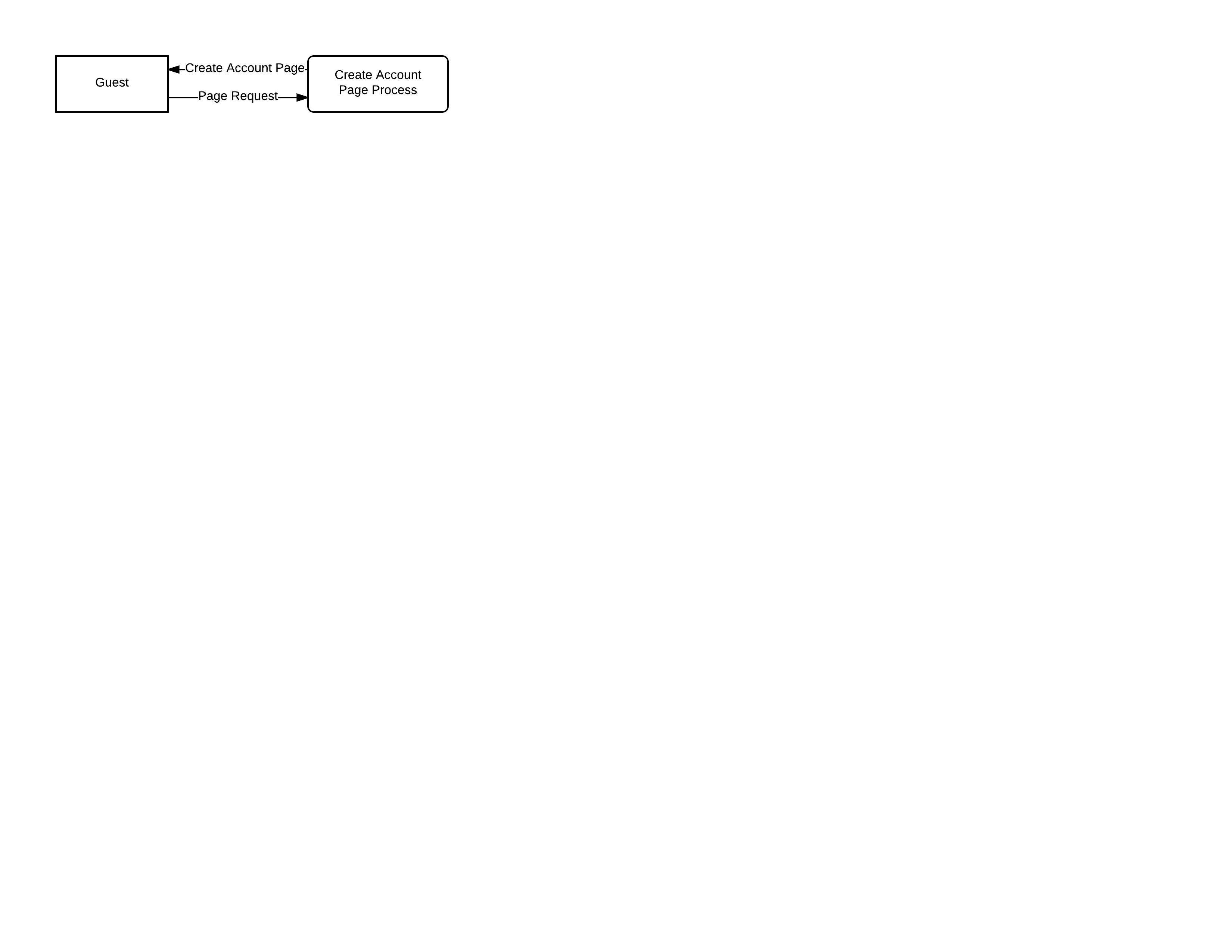
Return Home Feed

else:

Return error message

Request valid login

**L4 - Viewing Create Account Page**

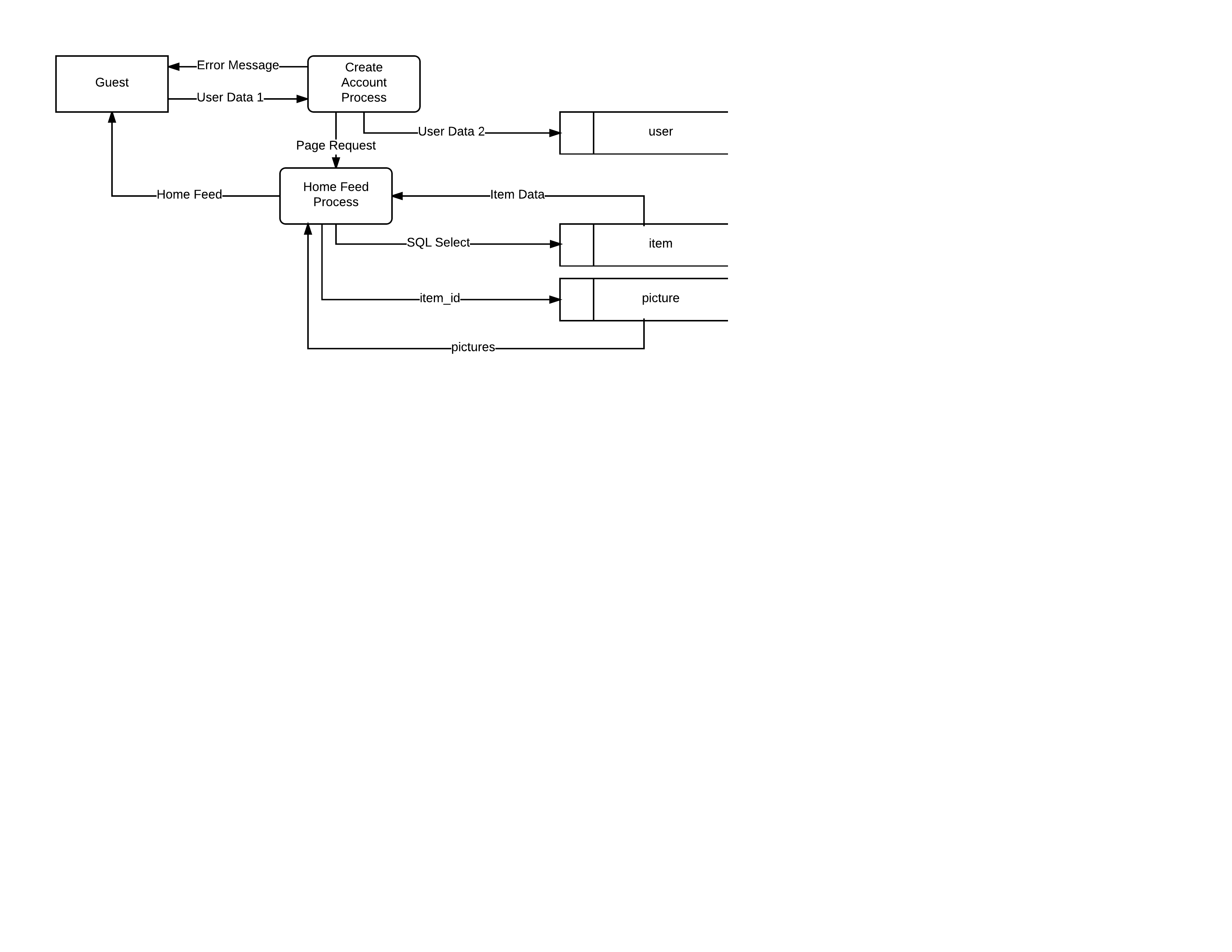


#Create Account Page

Request Create Account Page

Return Create Account Page

**L5 - Creating Account**



User Data 1 = first\_name, last\_name, email, password, zip

User Data 2 = first\_name, last\_name, email, password\_hash, zip

SQL Select = status, {user\_id}

Item Data = item\_id, title, price, quantity

#Account Creation

Get info for new account

if(info is valid):

Save data to user table

Log new user in

Generate Home Feed

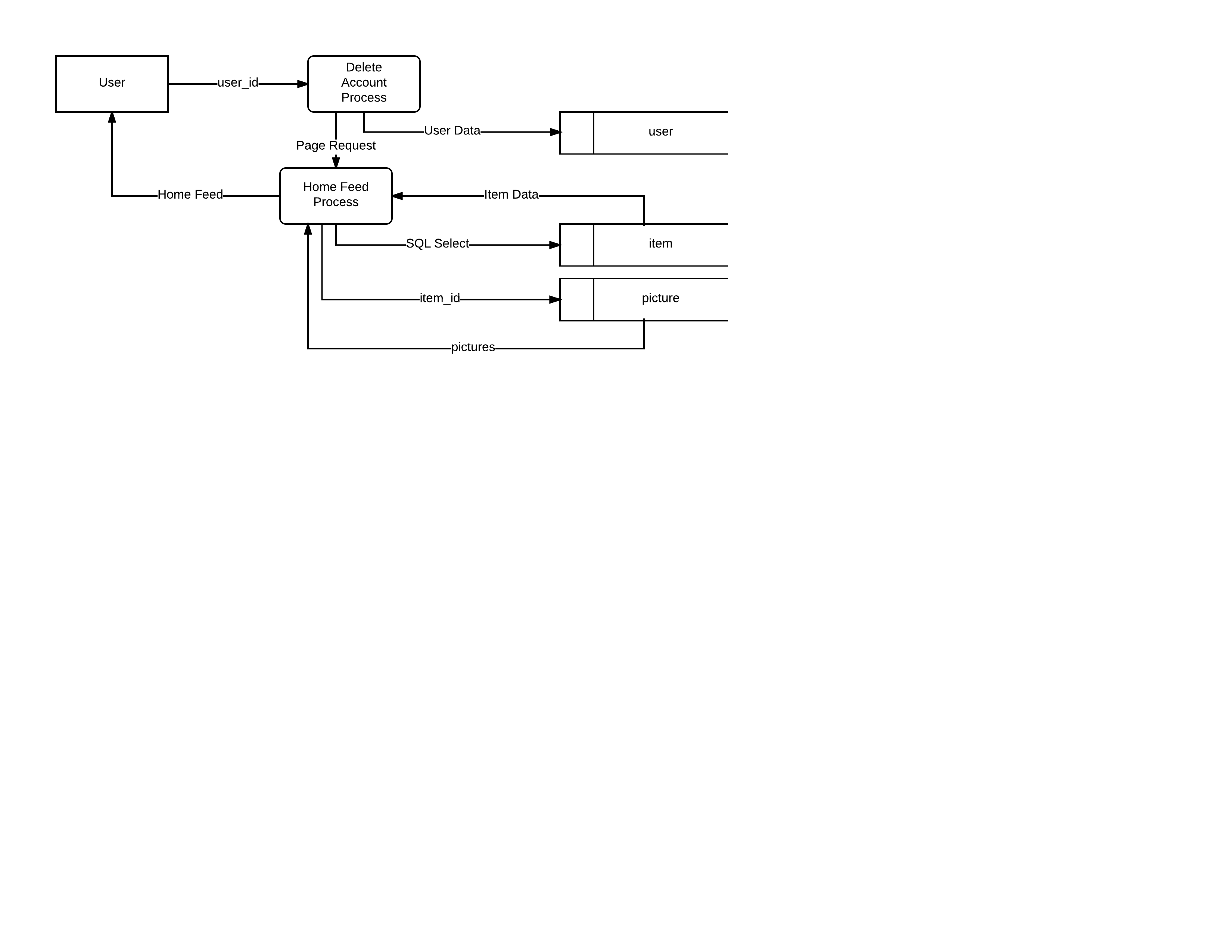
Return Home Feed

else:

Return error message

Request valid information

**L6 - Deleting Account**



User Data = user\_id, status

SQL Select = user\_id, status

Item Data = item\_id, title, price, quantity

#Account Deletion

Get user\_id

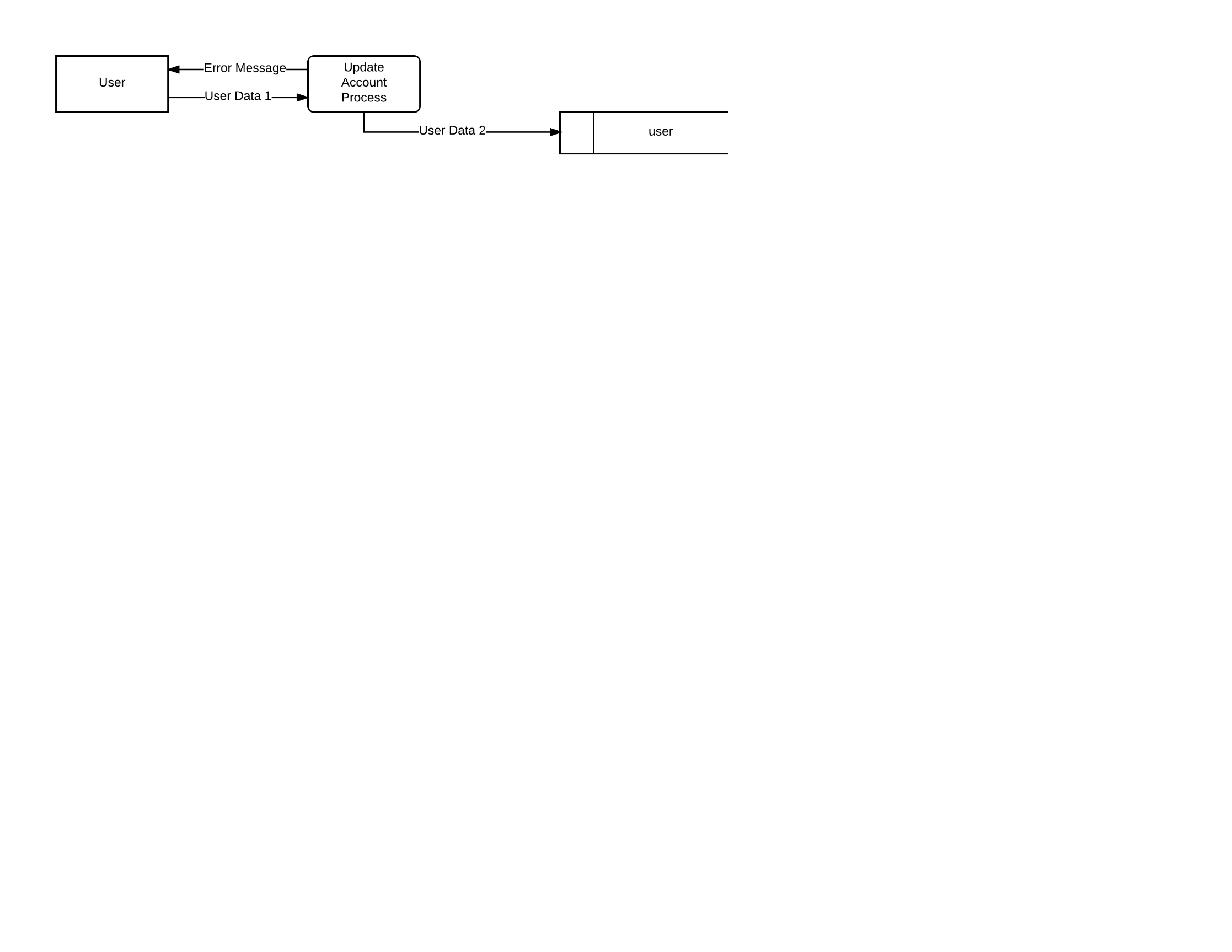
Set account status for user\_id to inactive in user table

Log user out

Generate Home Feed

Return Home Feed

**L7 - Updating Account**



User Data 1 = first\_name, last\_name, email, password, zip

User Data 2 = first\_name, last\_name, email, password\_hash, zip

#Updating Account

Get update data

if(data is valid):

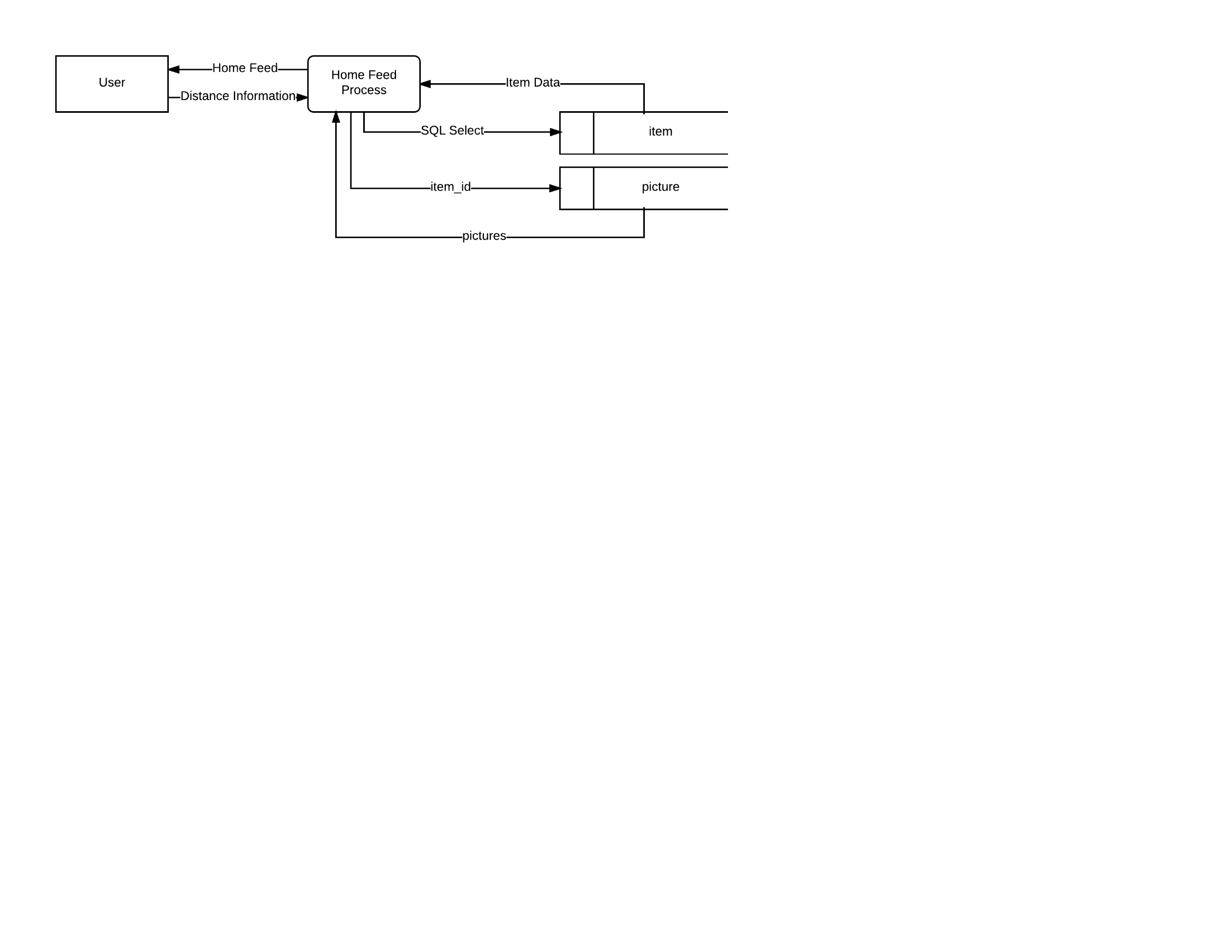
Update user in user table

else:

Return error message

Request valid information

**L8 - Changing Distance**



Distance Data = user\_id, distance

SQL Select = status, {user\_id}

Item Data = item\_id, title, price, quantity

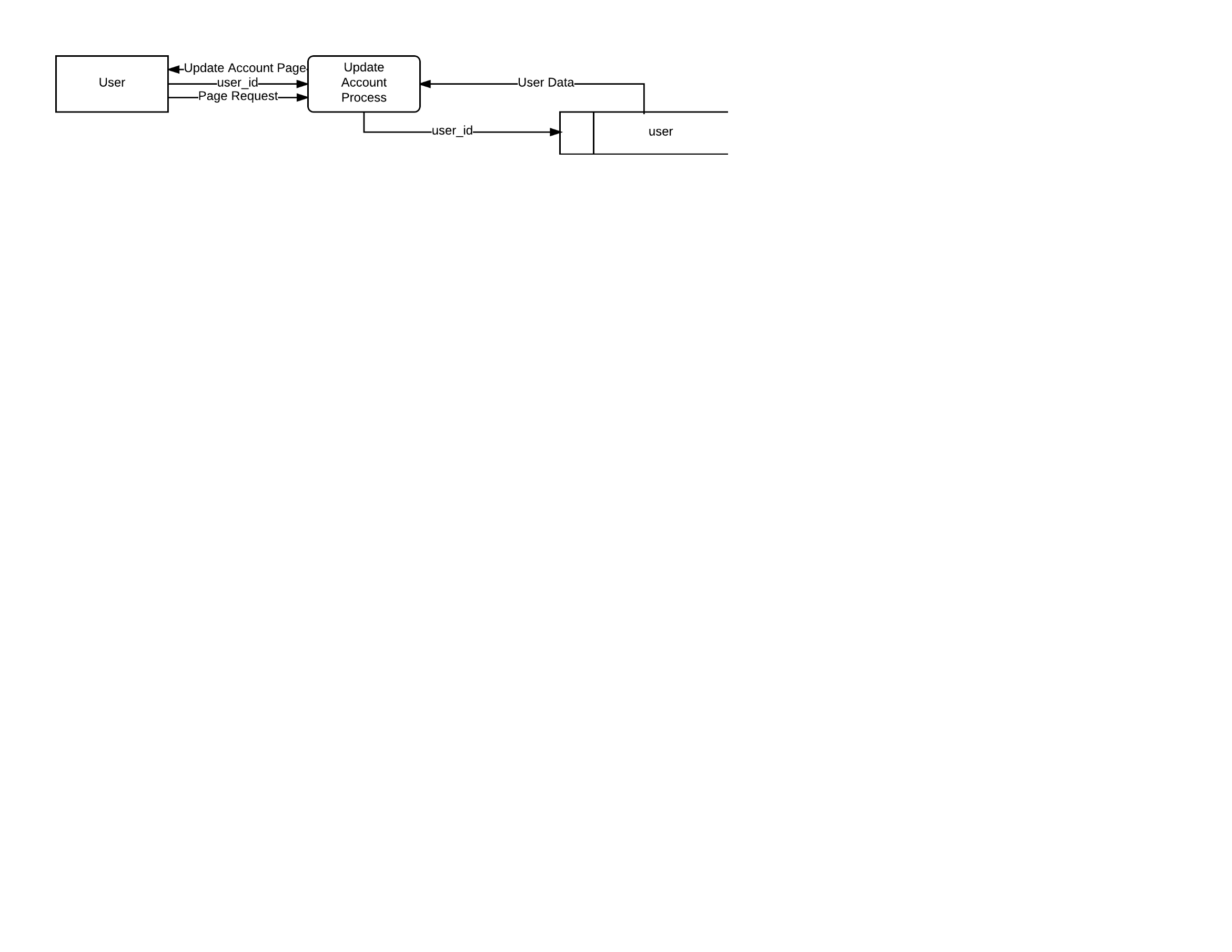
#Changing Distance

Get new distance

Generate Home Feed

Return Home Feed

**L9 - Viewing Update Account Page**



User Data = first\_name, last\_name, email, password\_hash, zip

#Account Update Page

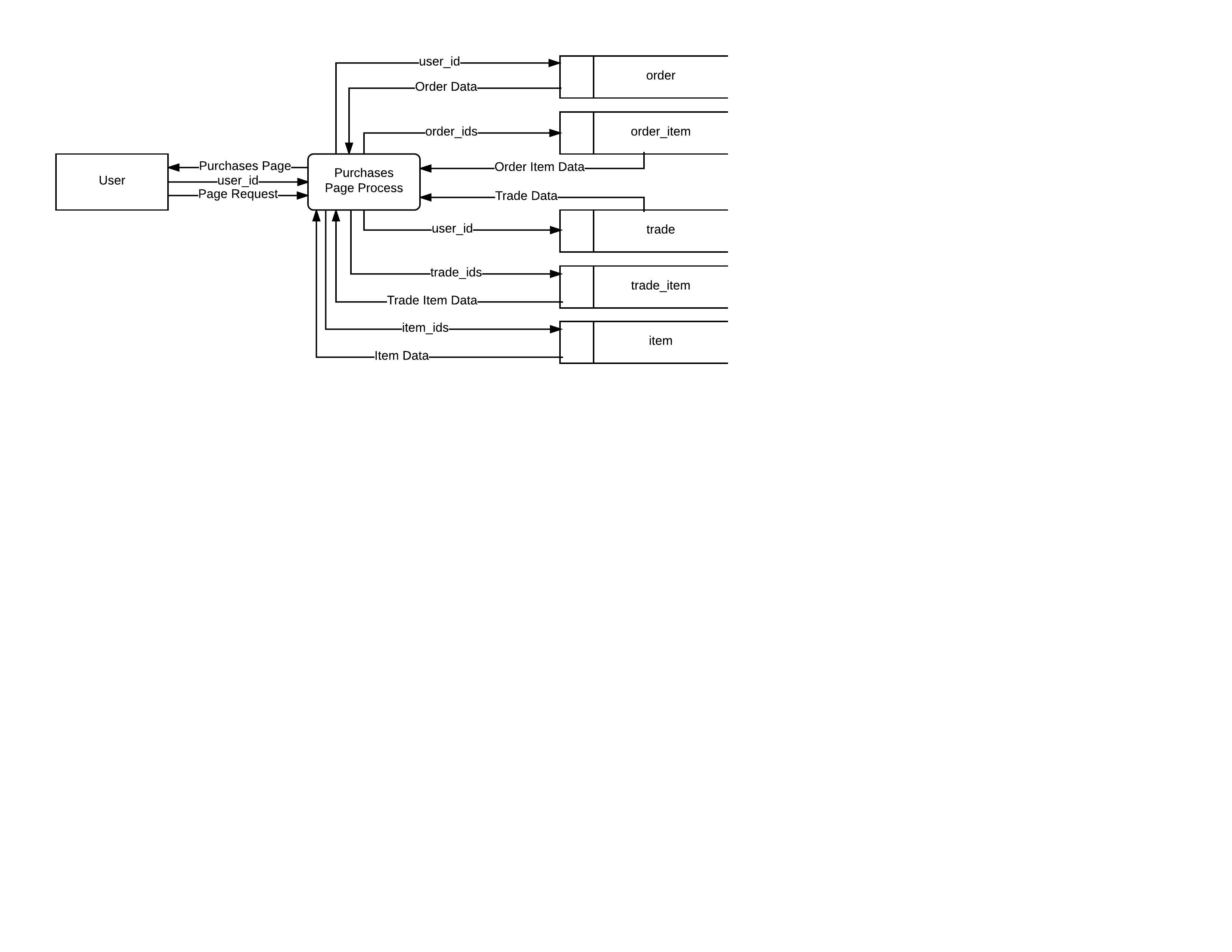
Get user\_id

Get user data from user table

Generate Update Account Page

Return Update Account Page

**L10 - Viewing Purchase Page**



Trade Data = trade\_id, status

Trade Item Data = item\_id, quantity

Order Data = order\_id, status

Order Item Data = item\_id, quantity

Item Data = seller\_id, title, price, quantity

#Viewing Purchases

Get user\_id

Get orders from order table

Get items associated with orders from order\_item table

Get trades from trade table

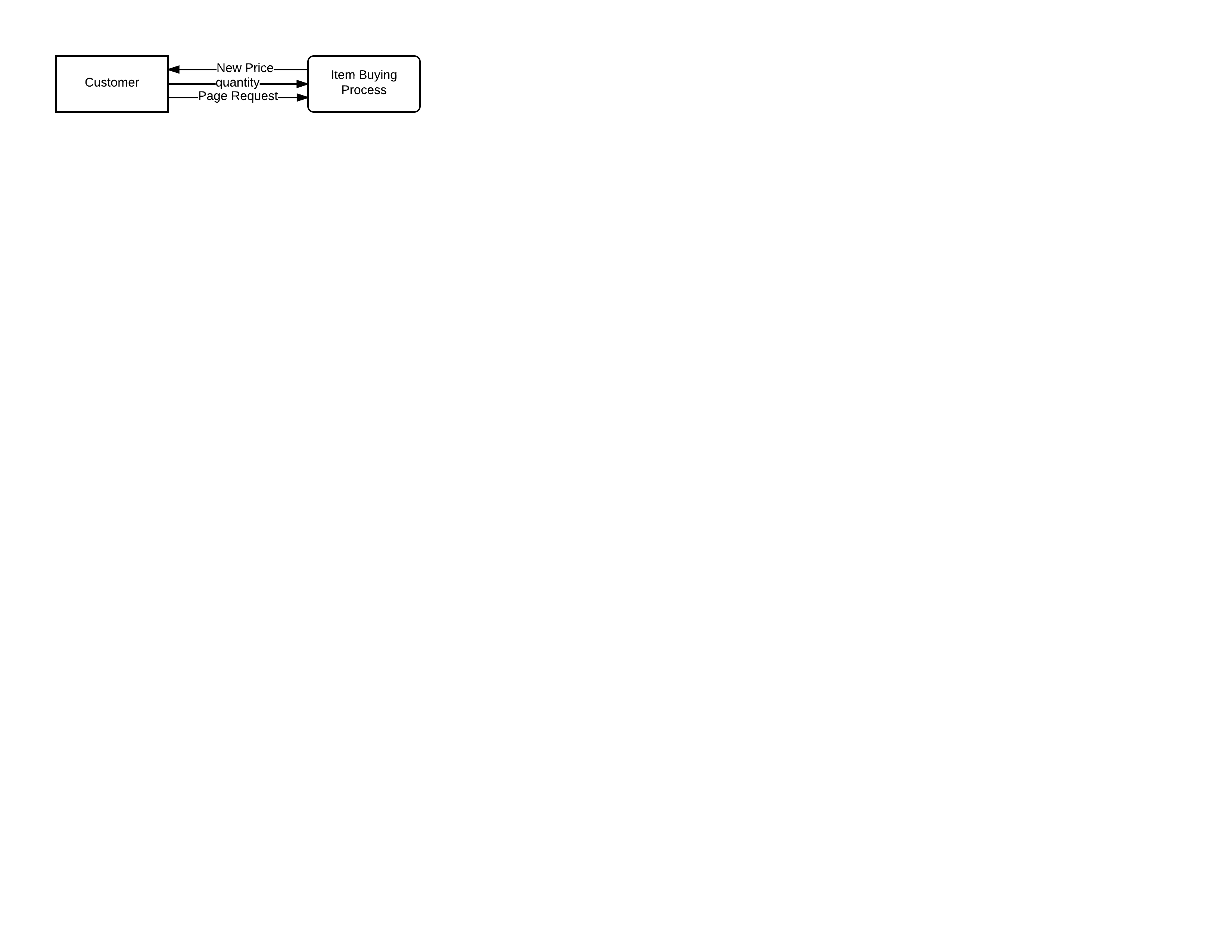
Get items associated with trades from trade\_item table

Get item information from item table

Generate Purchases Page

Return Purchases Page

**C1 - Changing Buy Amount**



#Changing Quantity

Get new quantity

Return new price

**C2 - Buying an Item**



Order Data 1 = user\_id, status, SQL Select for order\_id

Order Data 2 = order\_id, item\_id, quantity

SQL Select = user\_id, status

Item Data = item\_id, title, price, quantity

#Buying

Get order info

Create new order in order table

Get new order\_id

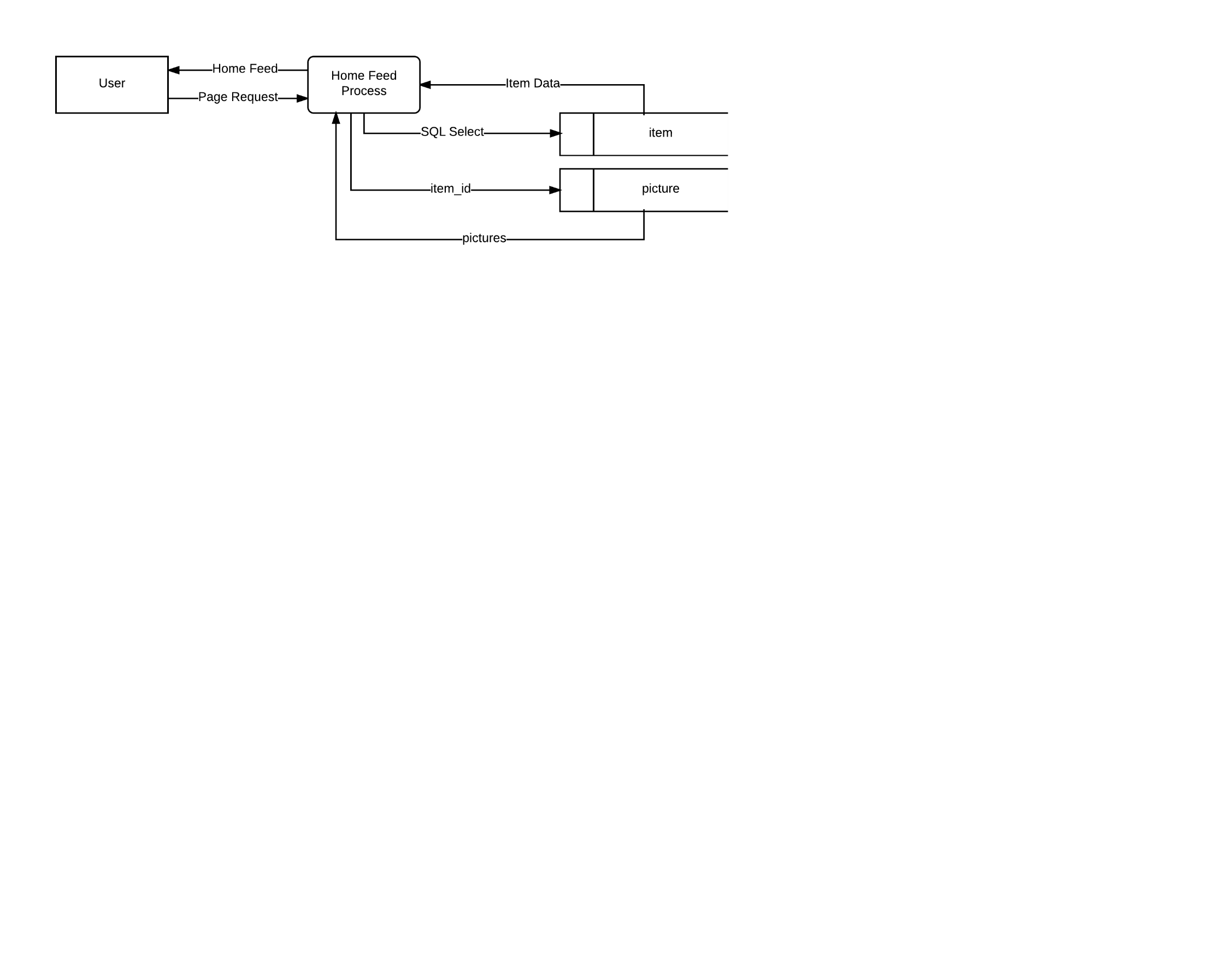
Create associations of items to new order in order\_item table

Email the seller of the item

Generate Home Feed

Return Home Feed

**C3 - Returning to the Home Feed**



SQL Select = user\_id, status

Item Data = item\_id, title, price, quantity

#Hitting the Listing Tab

Request Home Feed

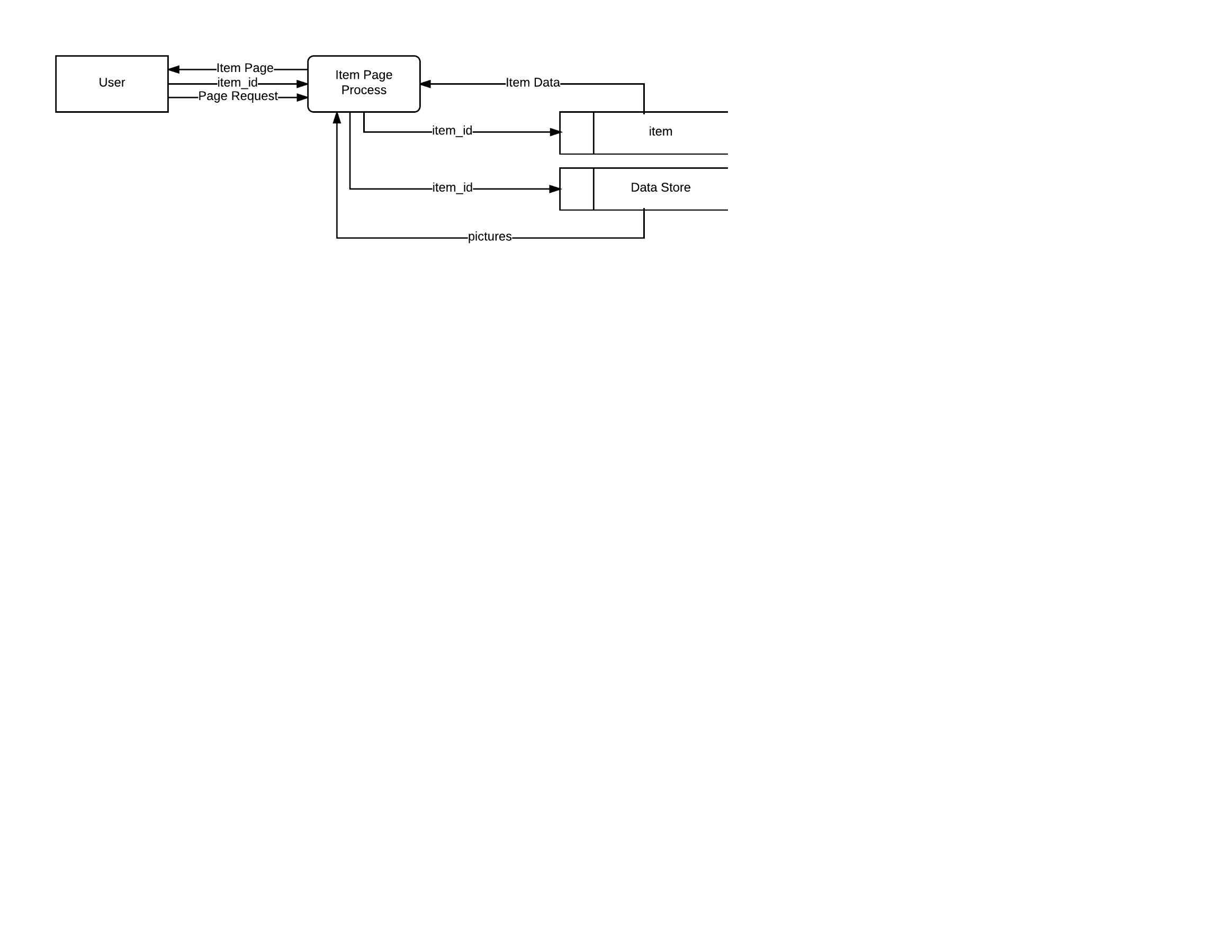
Get items within distance

Get item pictures

Generate page

Return page

**C4 - Viewing Item Information**



Item Data = seller\_id, title, description, type, price, quantity, condition, posted\_data, exp\_date,

pickup\_location, email, phone\_num

#Item Page

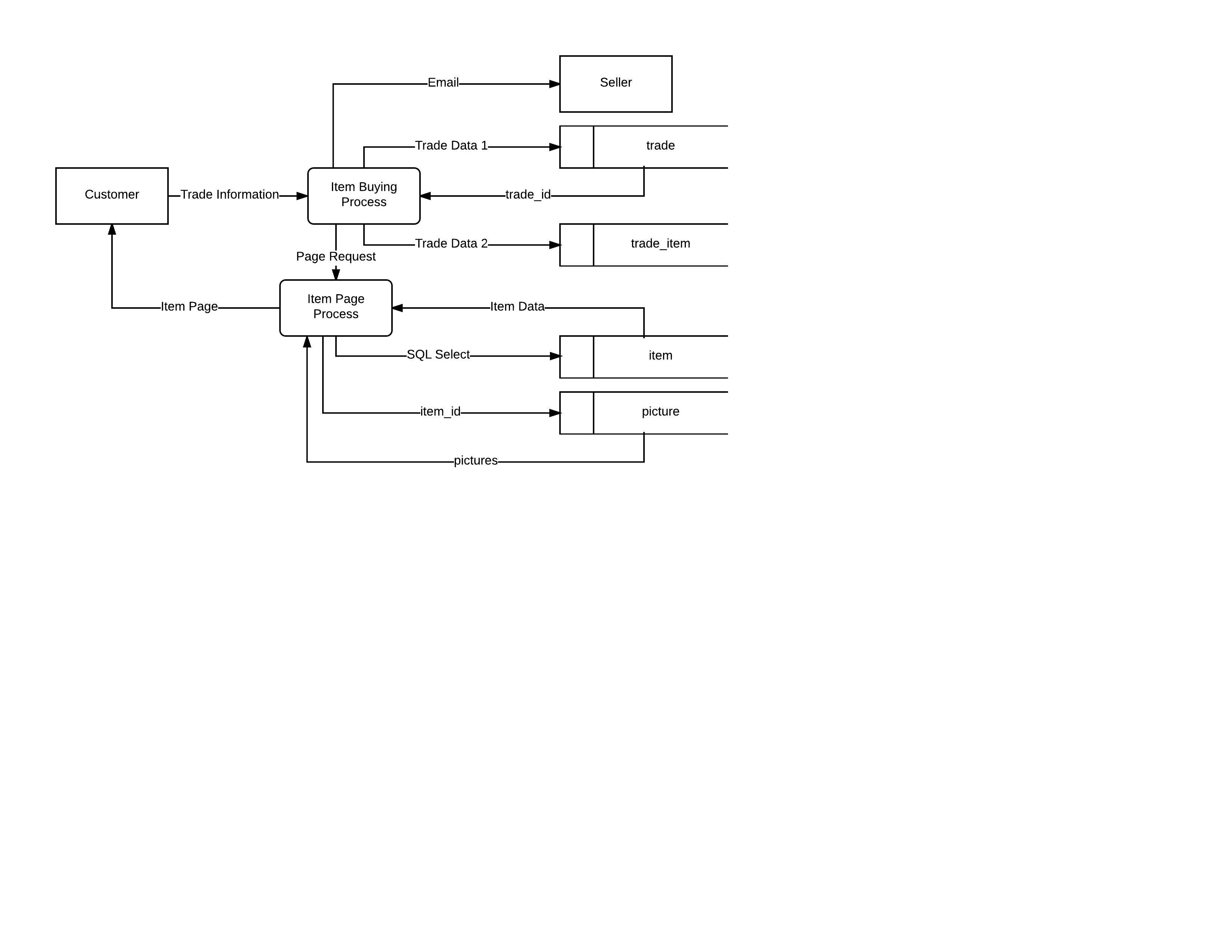
Get item\_id

Get item data from item table using item\_id

Generate Item Page

Return Item Page

**T1 - Offering a Trade**



Trade Data 1 = user\_id, status, SQL Select for trade\_id

Trade Data 2 = trade\_id, item\_id, quantity

SQL Select = item\_id, status

Item Data = item\_id, title, price, quantity

#Setting Up a Trade

Get trade info

Create new trade in trade table

Get new trade\_id

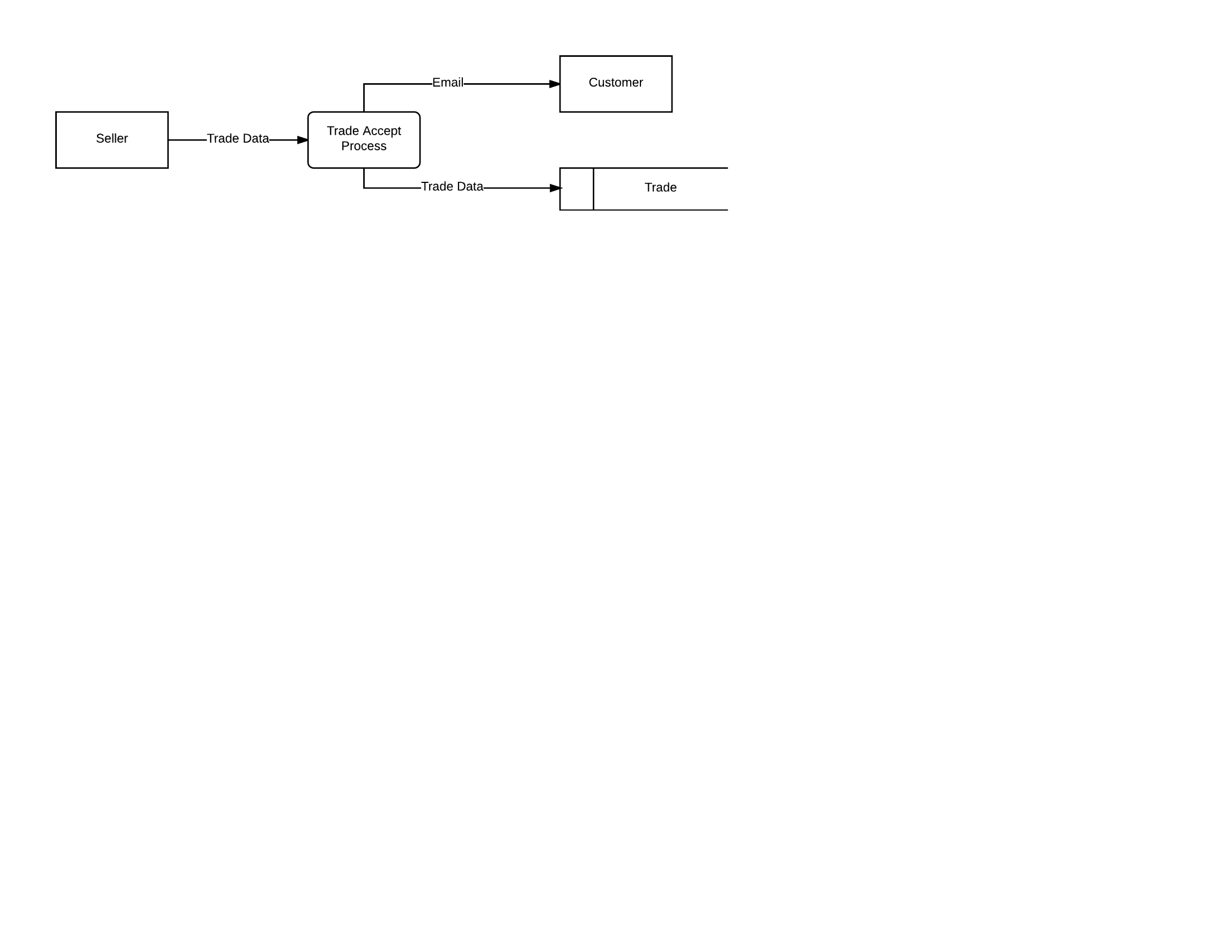
Create associations of items to new trade in trade\_item table

Email the seller of the item

Generate Item Page

Return Item Page

**T2 - Accepting a Trade**



Trade Data = trade\_id, status

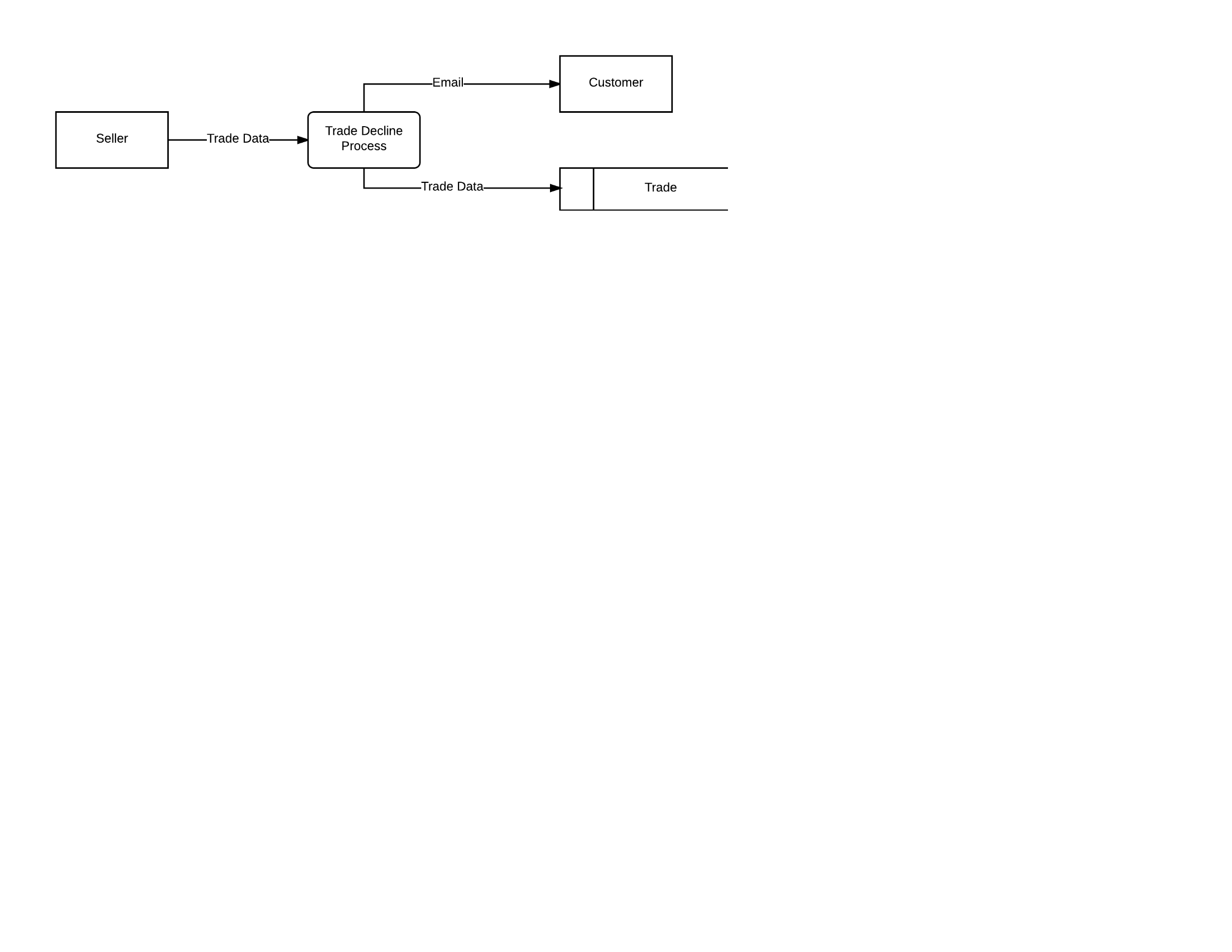
#Accepting Trades

Get trade data

Change status of trade in trade table

Email trading customer

**T3 - Declining a Trade**



Trade Data = trade\_id, status

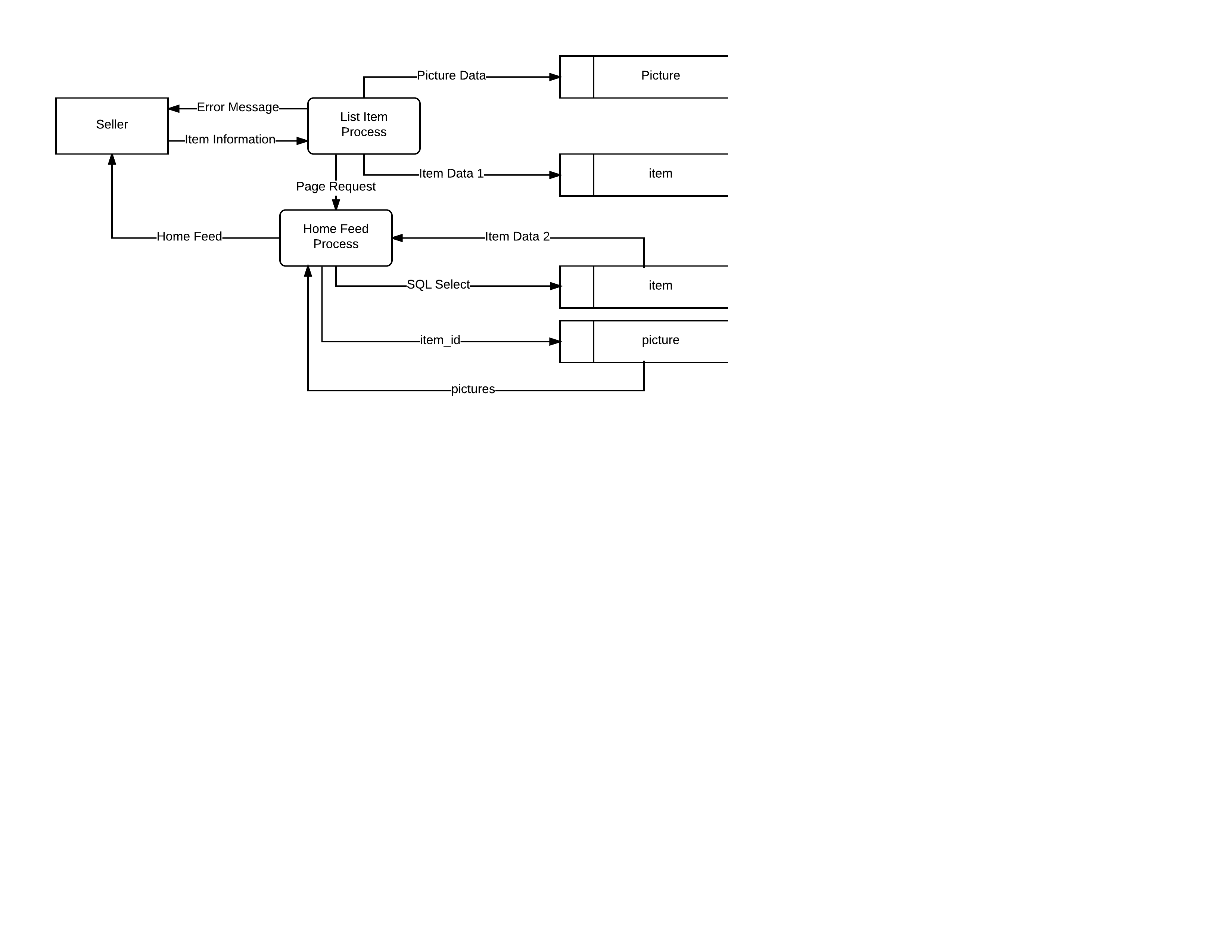
#Declining Trades

Get trade data

Change status of trade in trade table

Email trading customer

**S1 - Listing an Item**



Item Information = seller\_id, title, description, type, price, quantity, tags, condition, posted\_data,

exp\_date, pickup\_location, email, phone\_num, pictures

Picture Data = item\_id, picture

Item Data 1 = seller\_id, title, description, type, price, quantity, condition, posted\_data, exp\_date,

pickup\_location, email, phone\_num

SQL Select = item\_id, status

Item Data 2 = seller\_id, title, description, type, price, quantity, tags, condition, posted\_data,

exp\_date, pickup\_location, email, phone\_num

#New Listing

Get item info

if(info is valid):

Add new item to item table

Add item pictures to picture table

Generate Home Feed

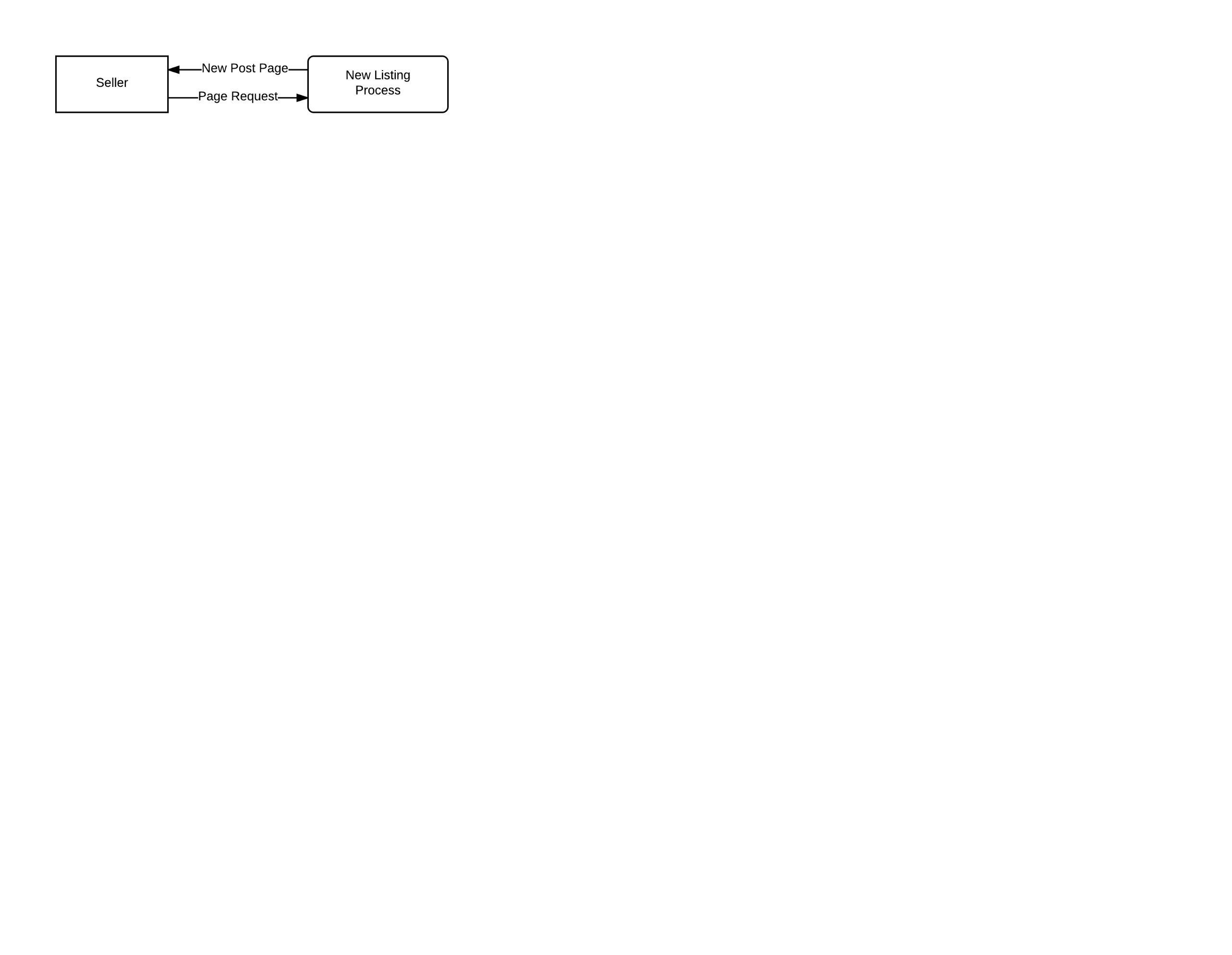
Return Home Feed

else:

Return error message

Request valid information

**S2 - Viewing New Listing Page**

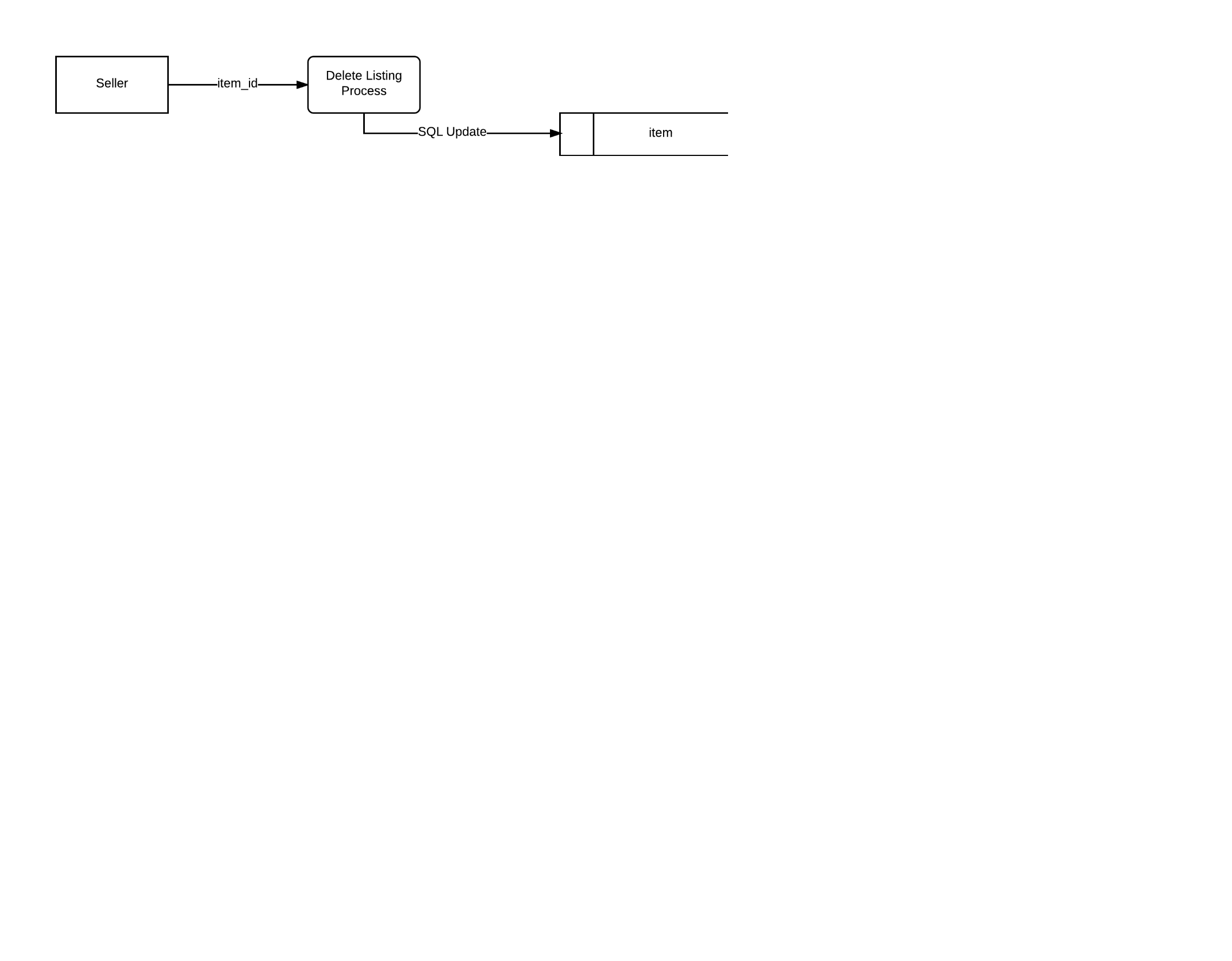


#New Post Page

Request New Post Page

Return New Post Page

**S3 - Removing a Listing**



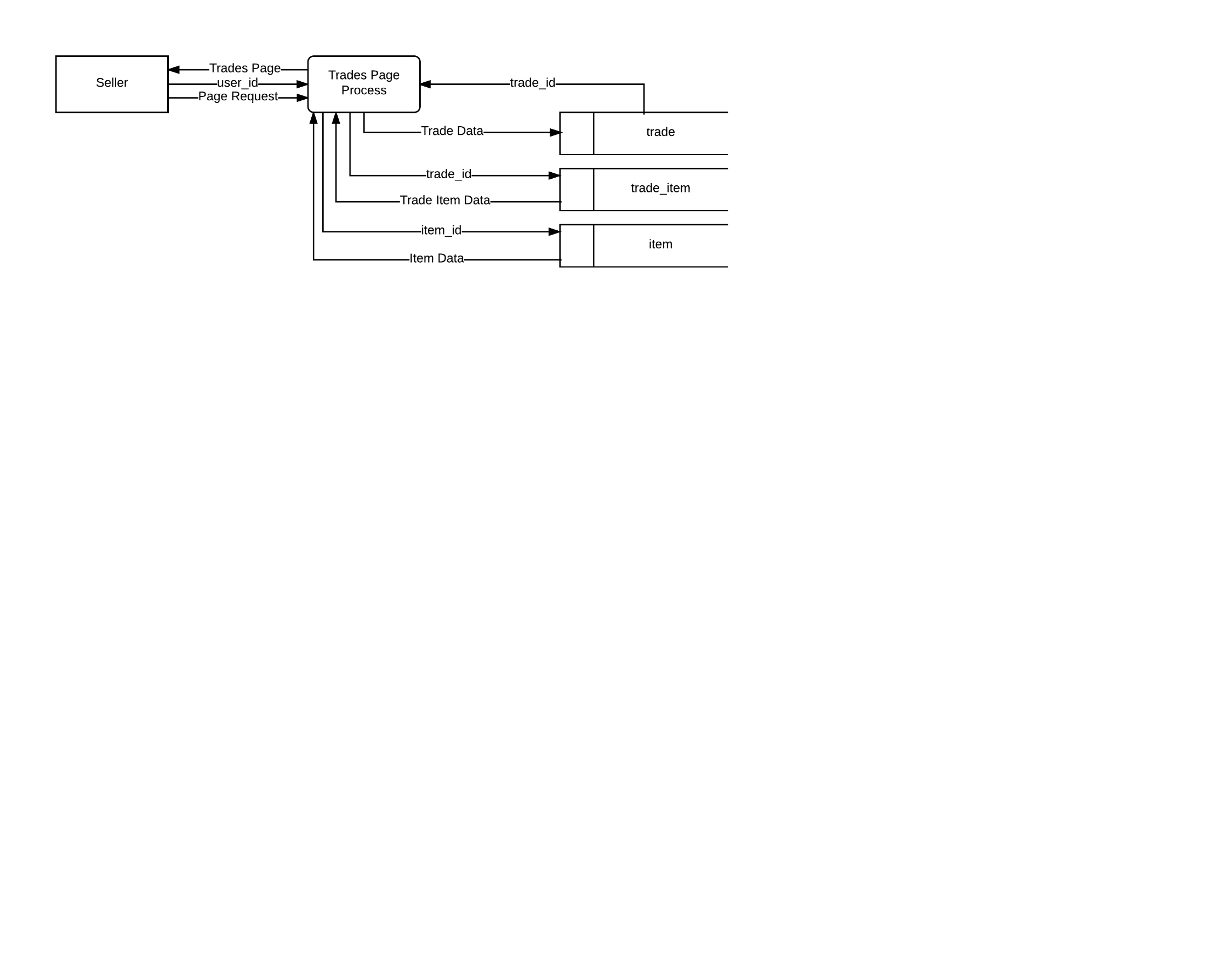
SQL Update = item\_id, status

#Removing Listing

Get item\_id

Update status of item in item table

**S4 - Viewing Trade Offers**



Trade Data = user\_id, status

Trade Item Data = item\_id, quantity

Item Data = title, prices

#Trade Offers Page

Get user\_id

Get trade\_ids for user\_id from trade table

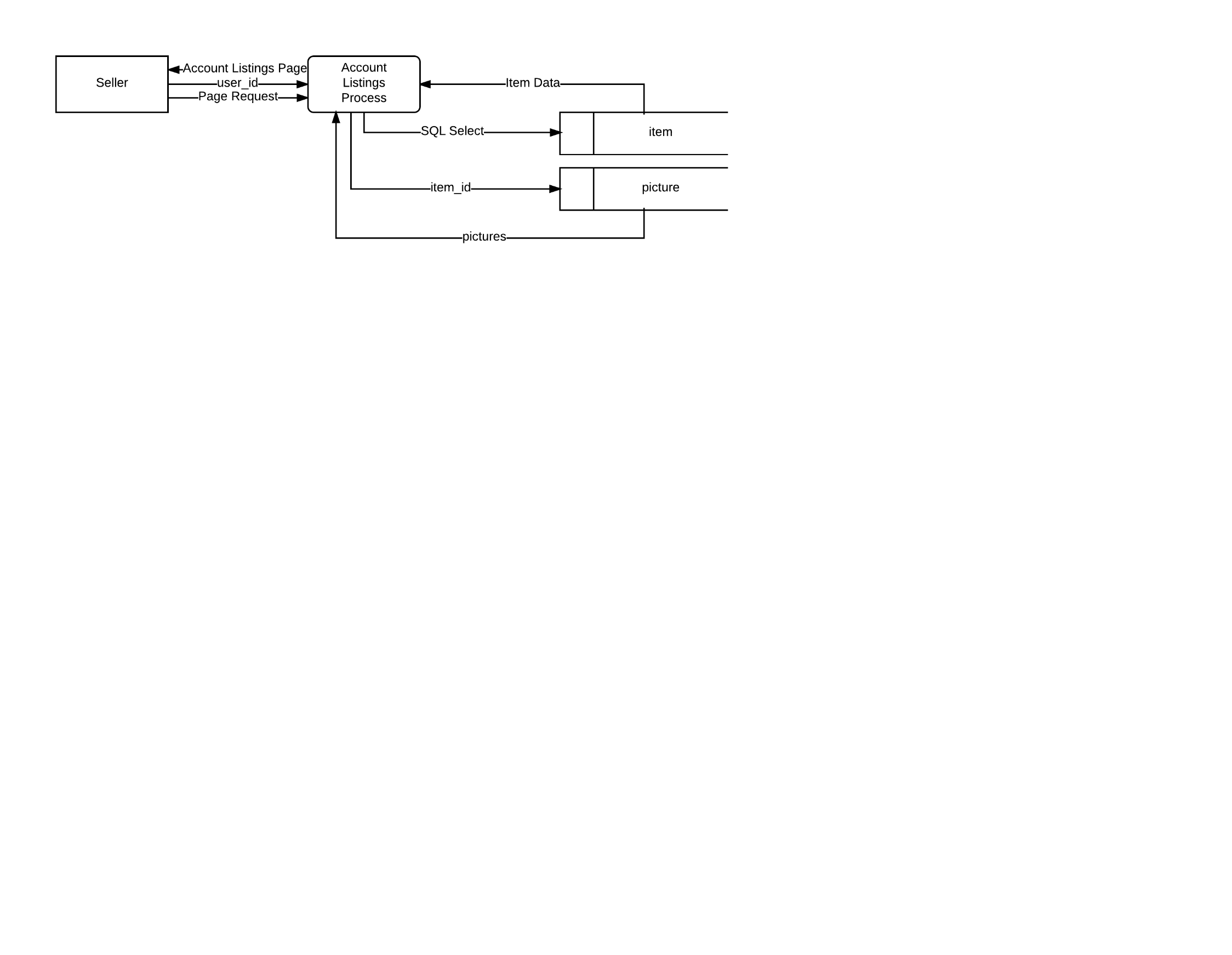
Get item\_ids associated with trades from trade\_item

Get item data of item\_ids

Generate Trade Offers Page

Return Trade Offers Page

**S5 - Viewing Account Listing Page**



SQL Select = user\_id, status

Item Data = item\_id, title, price, quantity

#Account Listings Page

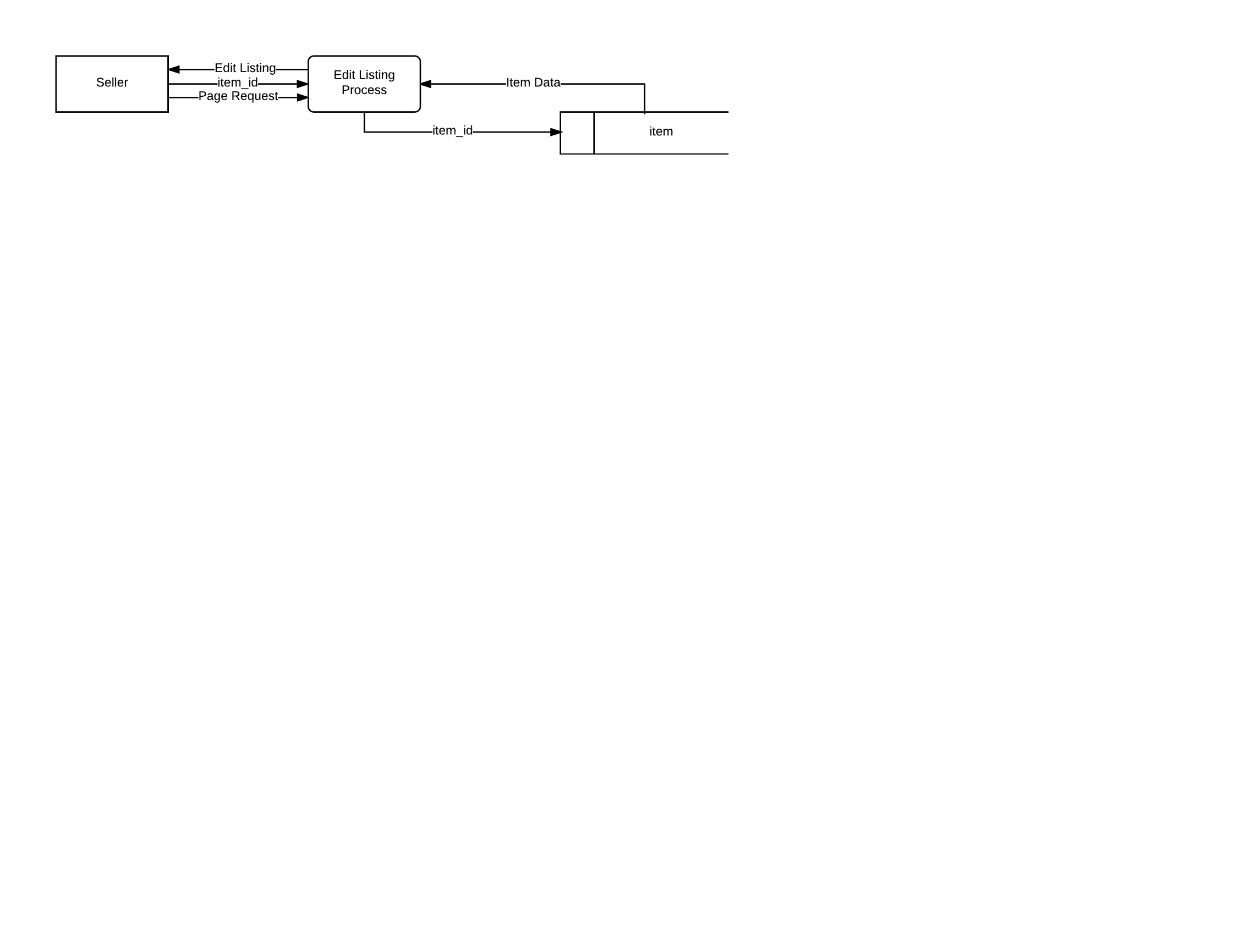
Get user\_id

Get data for users’ active listings from item table

Generate Account Listings Page

Return Account Listings Page

**S6 - Viewing Edit Listing Page**



Item Data = seller\_id, title, description, type, price, quantity, tags, condition, posted\_data,

exp\_date, pickup\_location, email, phone\_num

#Edit Listing Page

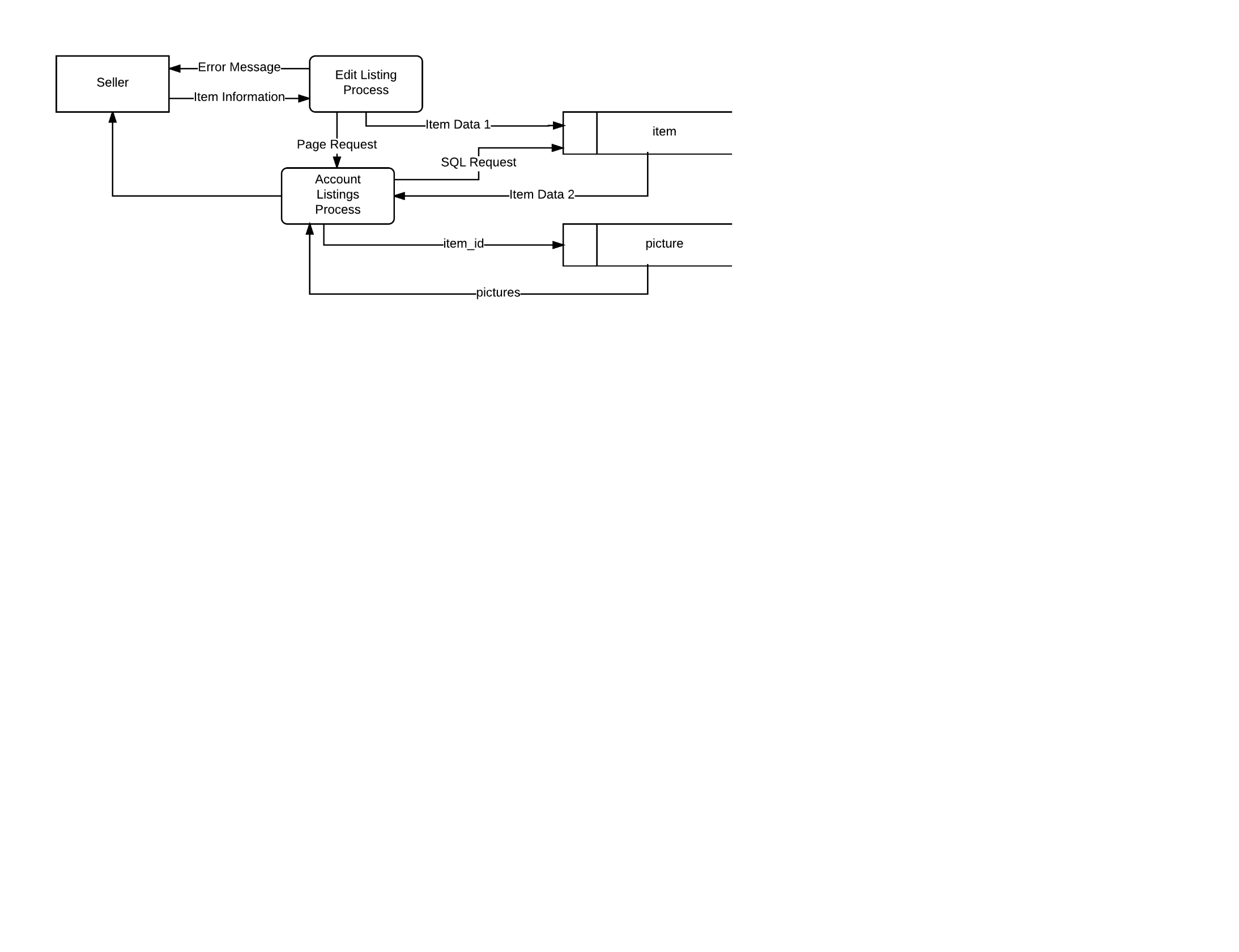
Get item\_id

Get item data from item table

Generate Edit Listing Page

Return Edit Listing Page

**S7 - Editing Listing**



Item Data 1 = seller\_id, title, description, type, price, quantity, tags, condition, posted\_data,

exp\_date, pickup\_location, email, phone\_num

SQL Select = user\_id, status

Item Data 2 = item\_id, title, price, quantity

#Editing Listing

Get item information

if(info is valid):

Update item info in item table

Generate Account Listing Page

Return Account Listing Page

else:

Return Error Message

Request valid information

## Data Dictionary

User - Entity for storing information associated with users

Relationships

* One-to-many with item
* One-to-many with order
* One-to-many with trade
* Many-to-one with zip\_location

Attributes

* user\_id, PK, unique identifier for each user in the system, integer
* first\_name, varchar
* last\_name, varchar
* email, varchar
* password\_hash, varchar
* zip, FK, integer
* active, describes the status of the account, bit

Item - Entity for storing information associated with listings

Relationships

* Many-to-one with user
* One-to-many with order\_item, associative
* One-to-many with trade\_item, associative
* One-to-many with picture

Attributes

* item\_id, PK, unique identifier for each item in the system, integer
* seller\_id, FK, user\_id of the seller, integer
* title, name of the item, varchar
* description, text
* type, category of the item, varchar
* price, decimal
* quantity, integer
* tags, searchable keywords for the item, text
* condition, varchar
* posted\_date, datetime
* exp\_date, expiration date of the listing, datetime
* pickup\_location, varchar
* email, varchar
* phone\_num, integer
* status, status of the listing, varchar

Order - Entity for storing information associated with buy orders

Relationships

* One-to-many with order\_item, associative
* Many-to-one with user

Attributes

* order\_id, PK, unique identifier for each order in the system, integer
* user\_id, FK, user\_id of the customer, integer
* status, status of the order, varchar

Trade - Entity for storing information associated with trade orders

Relationships

* One-to-many with trade\_item, associative
* Many-to-one with user

Attributes

* trade\_id, PK, unique identifier for each trade in the system, integer
* user\_id, FK, user\_id of the customer, integer
* status, status of the trade, varchar

Order Item (Associative) - Entity for associating Orders and Items

Relationships

* Many-to-one with item
* Many-to-one with order

Attributes

* order\_id, PK, compound key with item\_id to identify each item in an order, integer
* item\_id, PK, compound key with order\_id to identify each item in an order, integer
* quantity

Trade Item (Associative) - Entity for associating Trades and Items

Relationships

* Many-to-one with item
* Many-to-one with trade

Attributes

* trade\_id, PK, compound key with item\_id to identify each item in an order, integer
* item\_id, PK, compound key with trade\_id to identify each item in an order, integer
* quantity

Picture - Entity for storing information associated with item pictures

Relationships

* Many-to-one with item

Attributes

* picture\_id, PK, unique identifier for each picture on a posted item, integer
* item\_id, FK, item\_id for the item the picture is associated with, integer
* picture, a picture link or stored picture, blob

Zip Location - Entity for storing information associated with user locations

Relationships

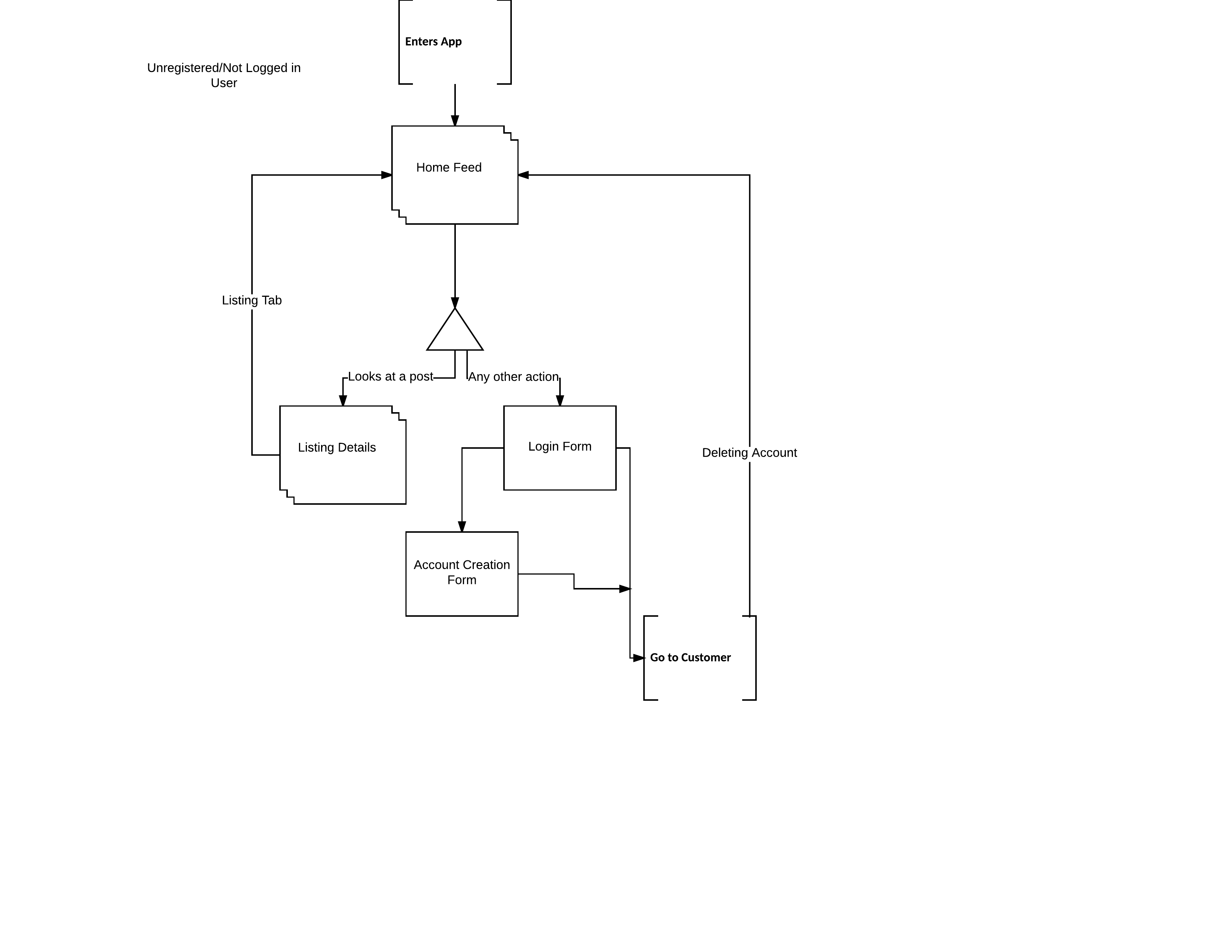
* One-to-many with user

Attributes

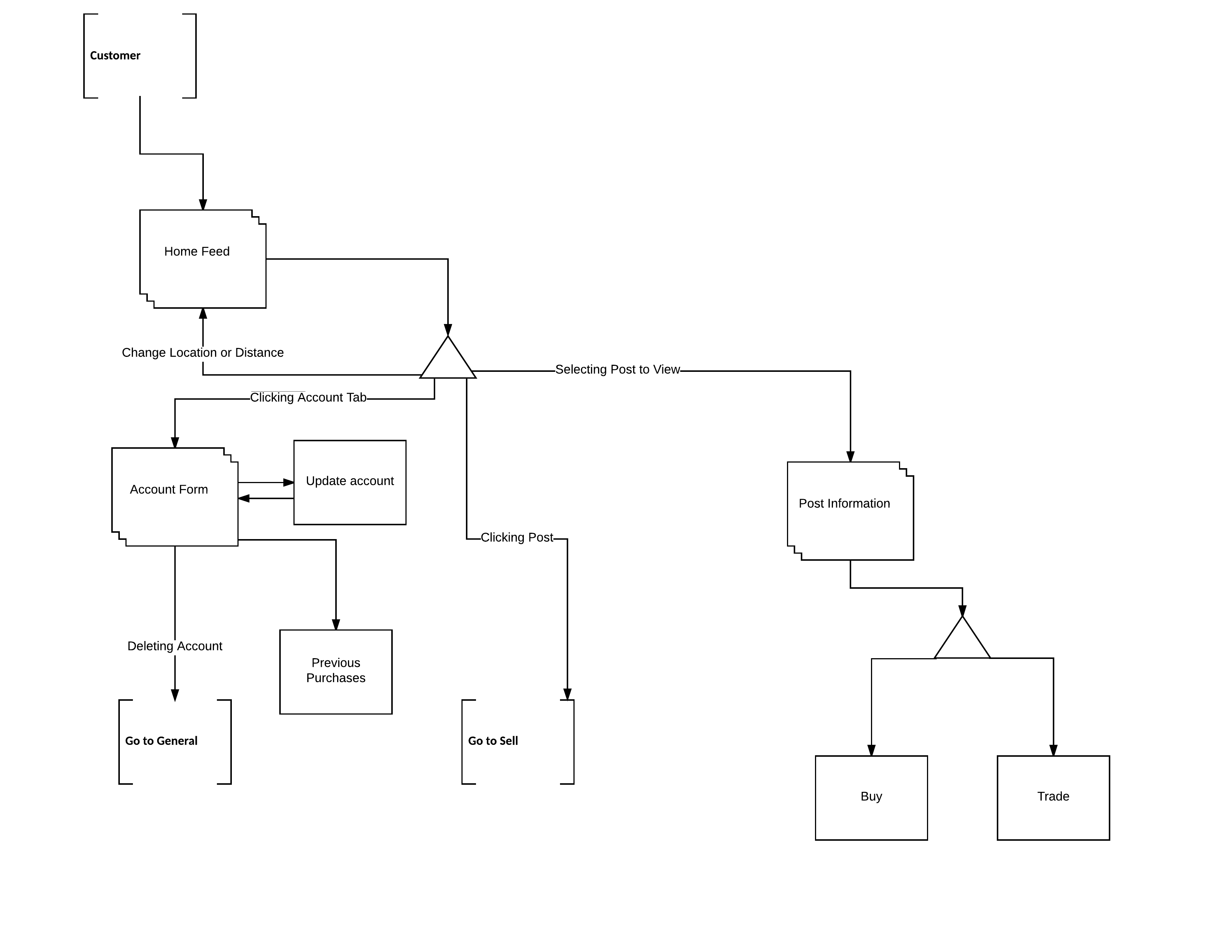
* zip, PK, integer
* longitude, decimal
* latitude, decimal

## User Experience Diagrams

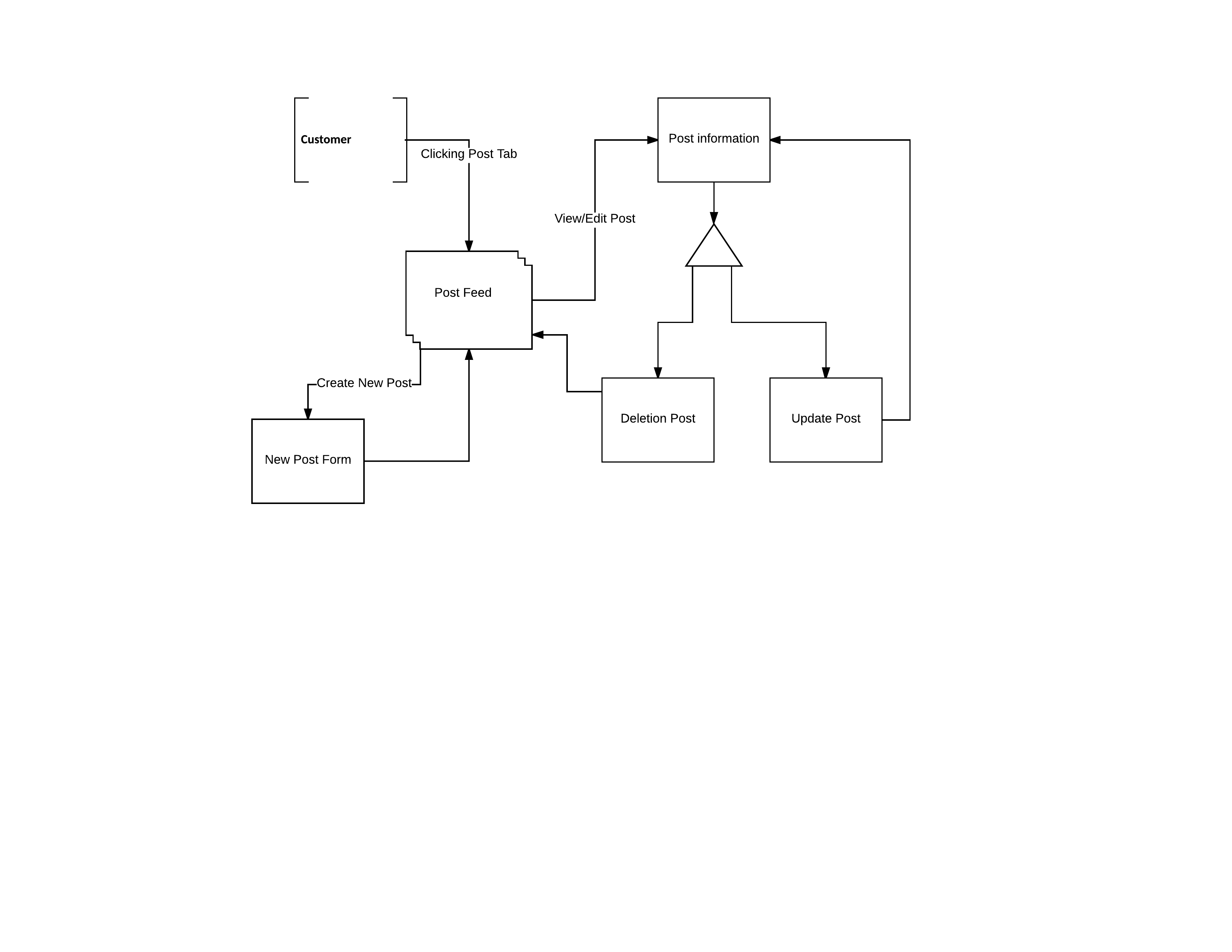
### General



### Customer

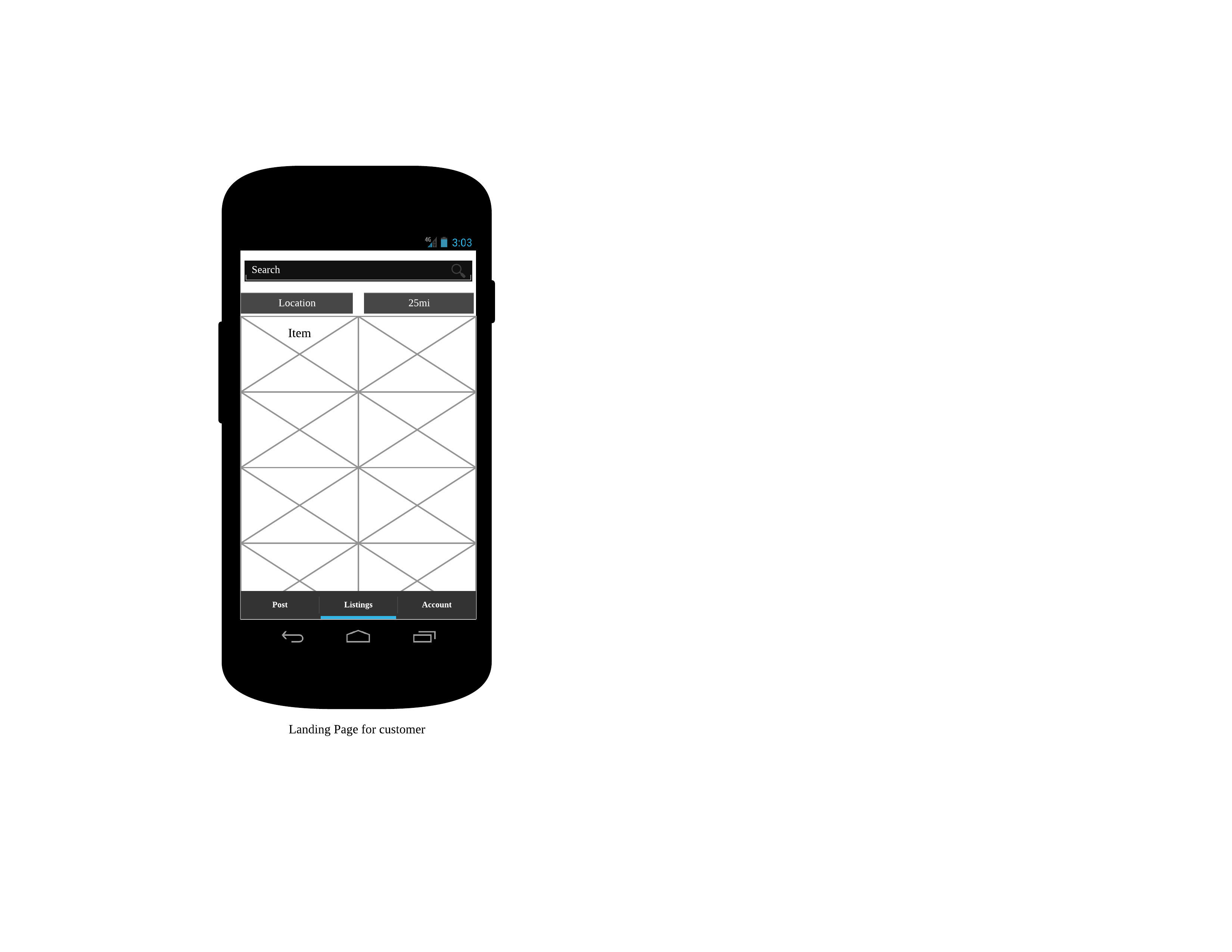


### Sell

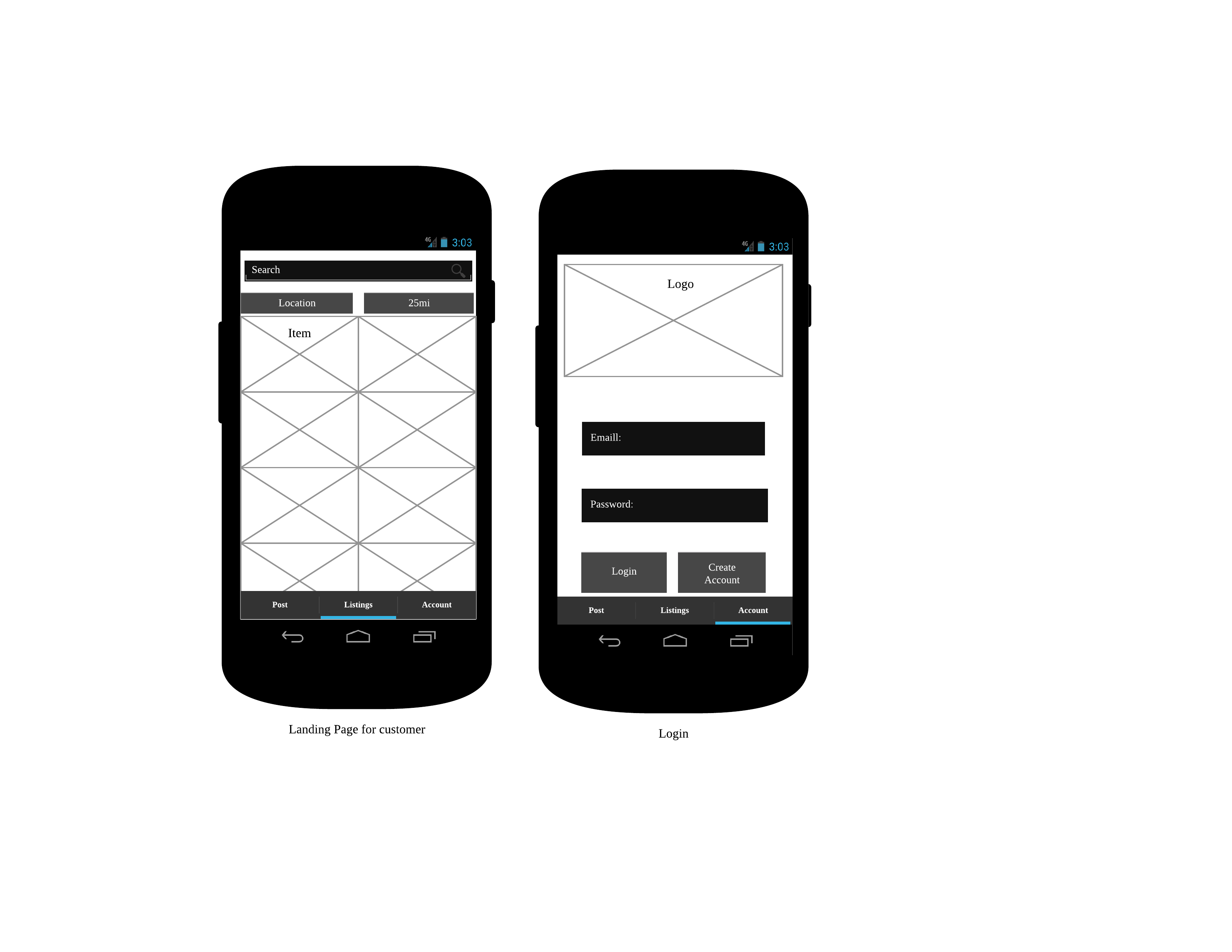


## Wireframes

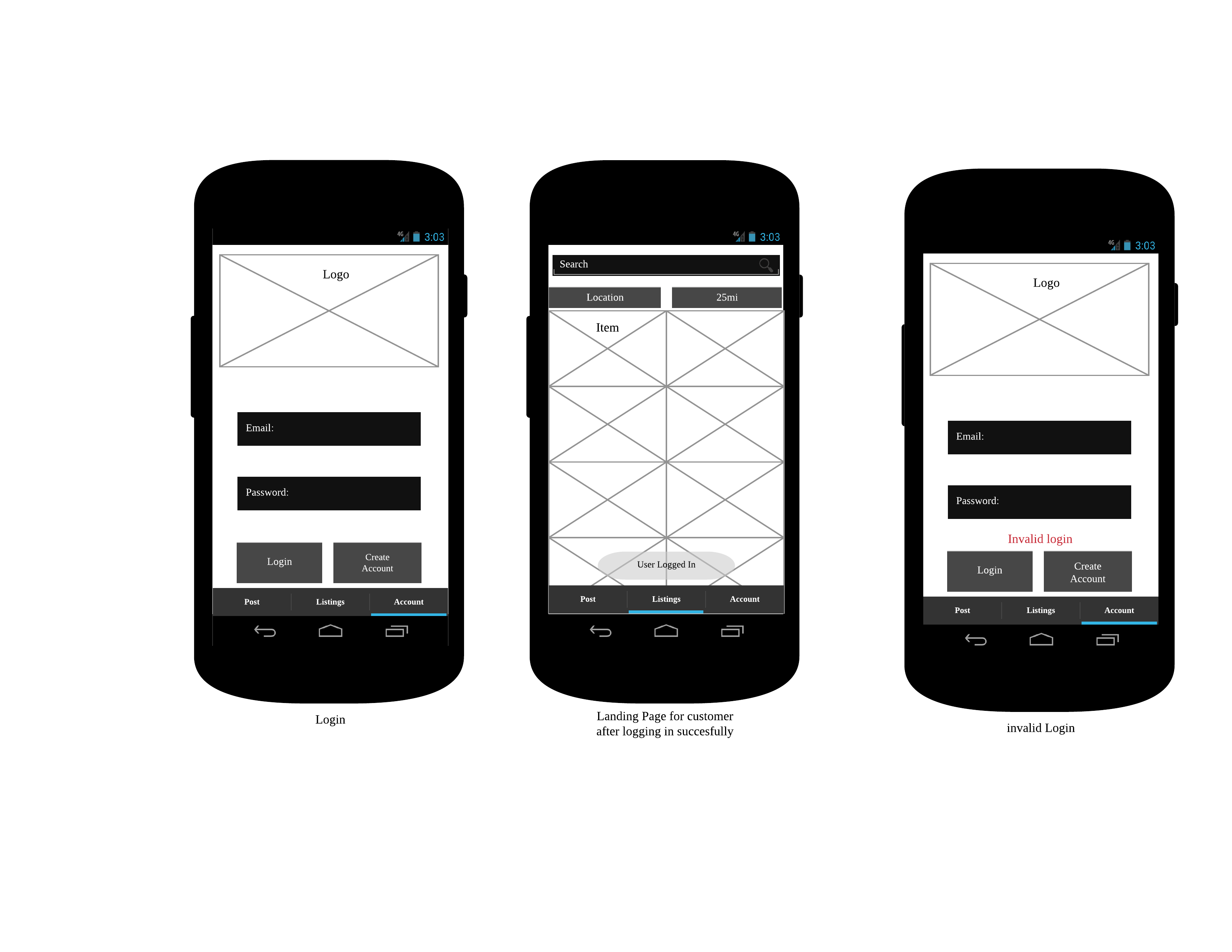
**L1**



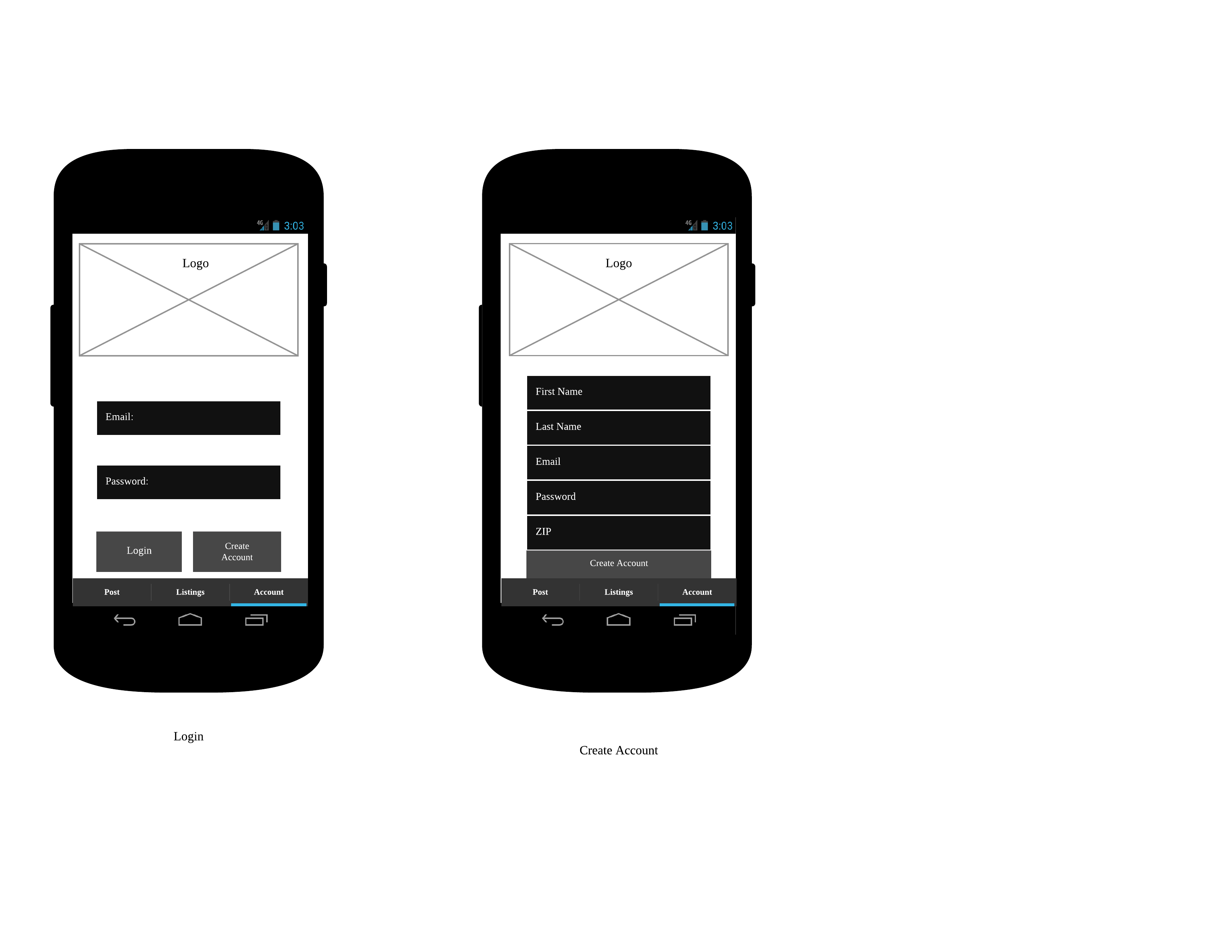
**L2**



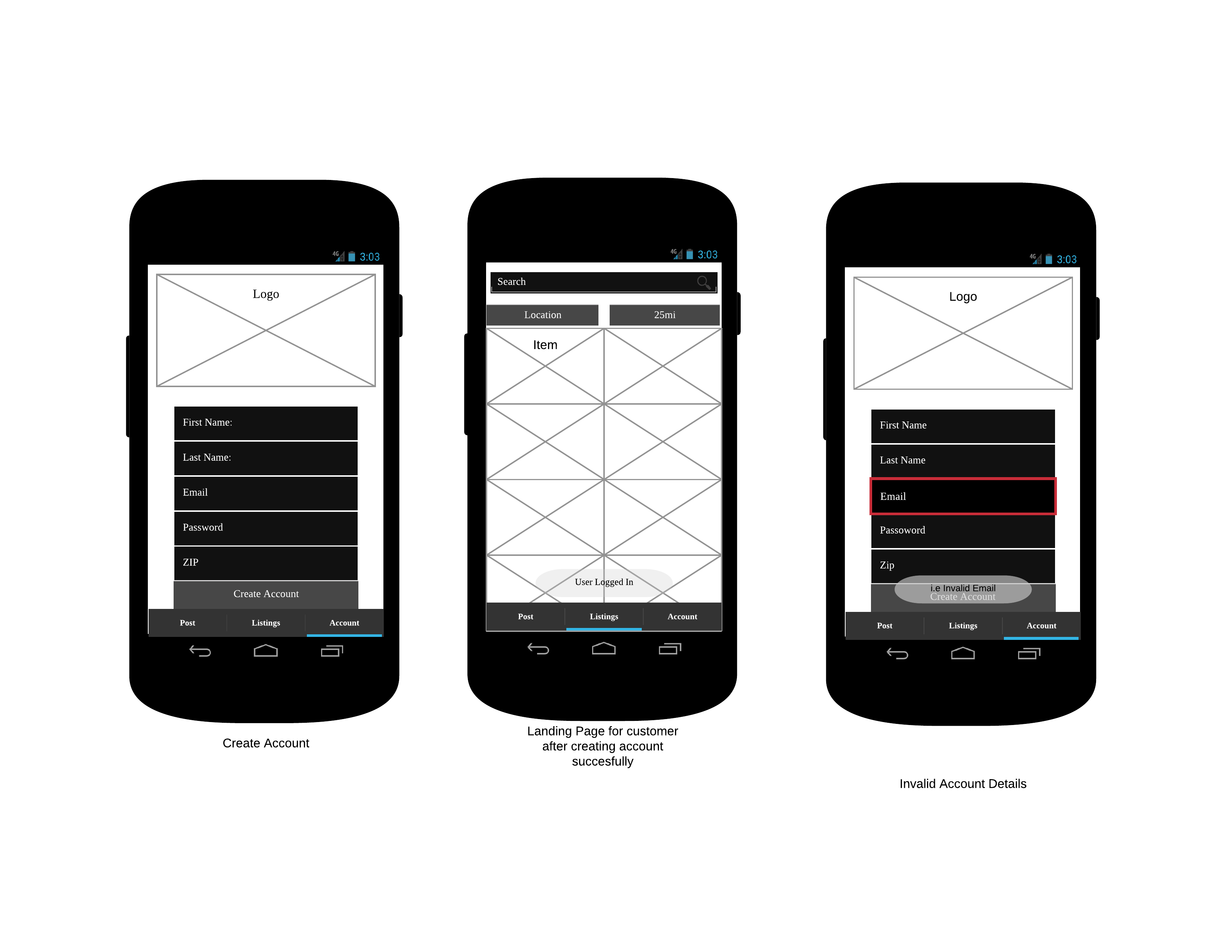
**L3**



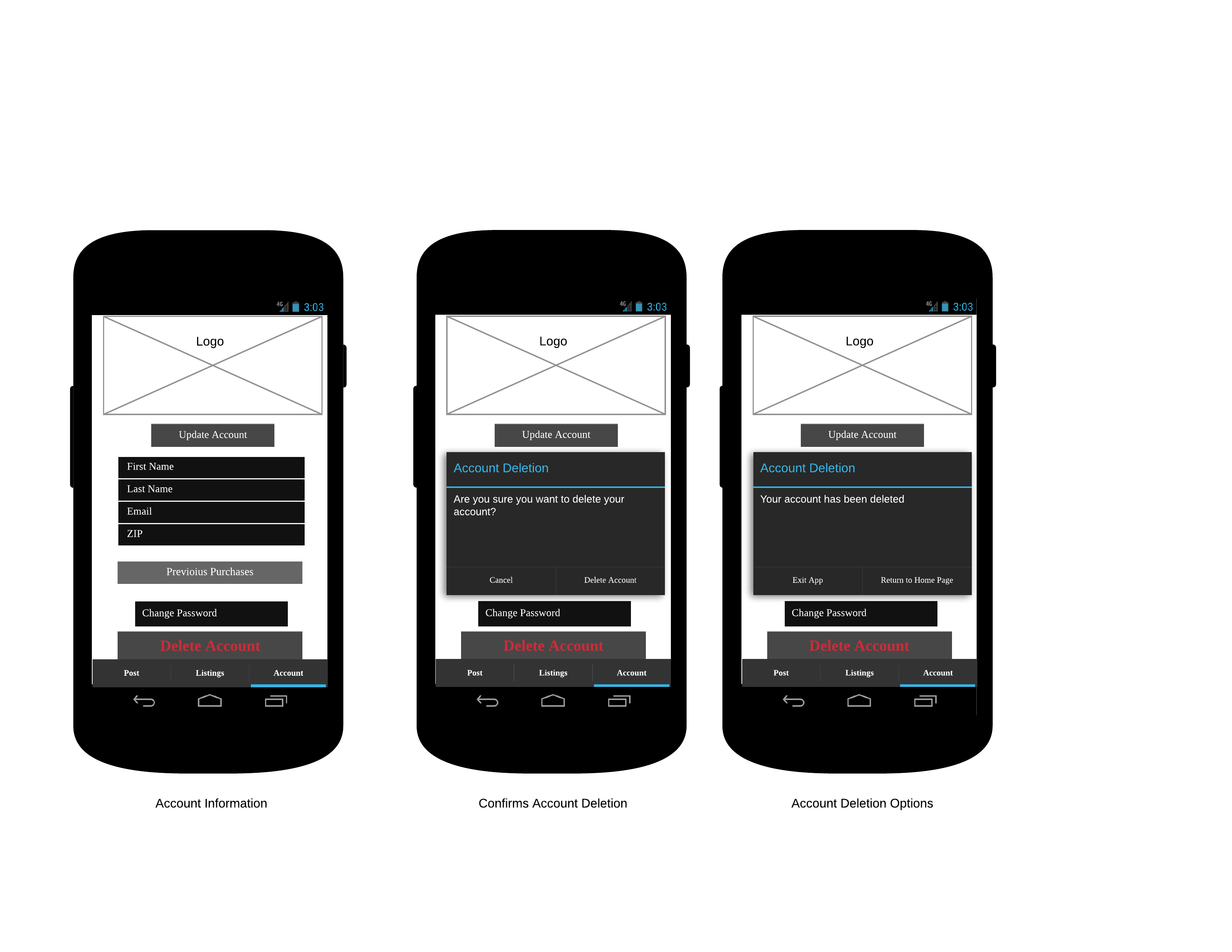
**L4**



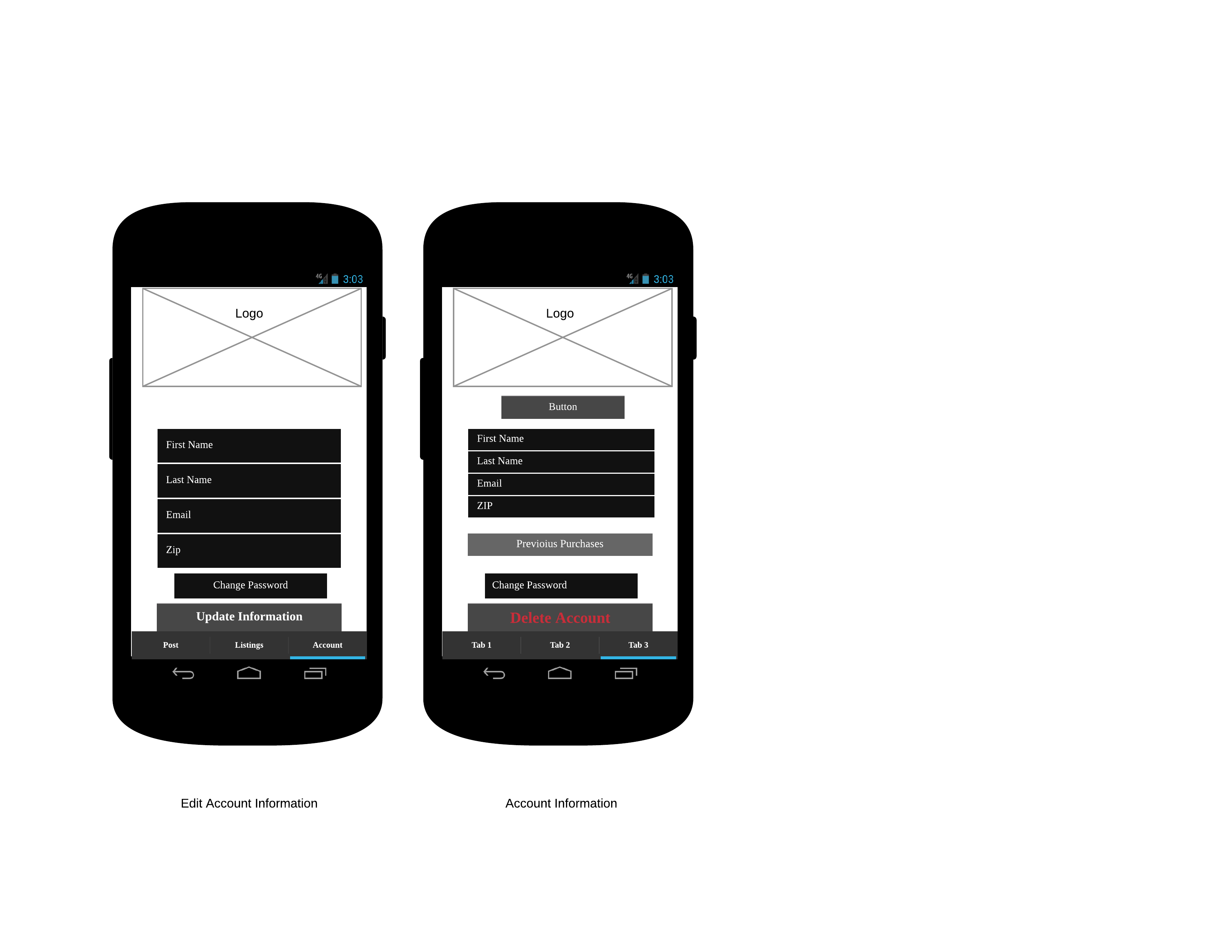
**L5**



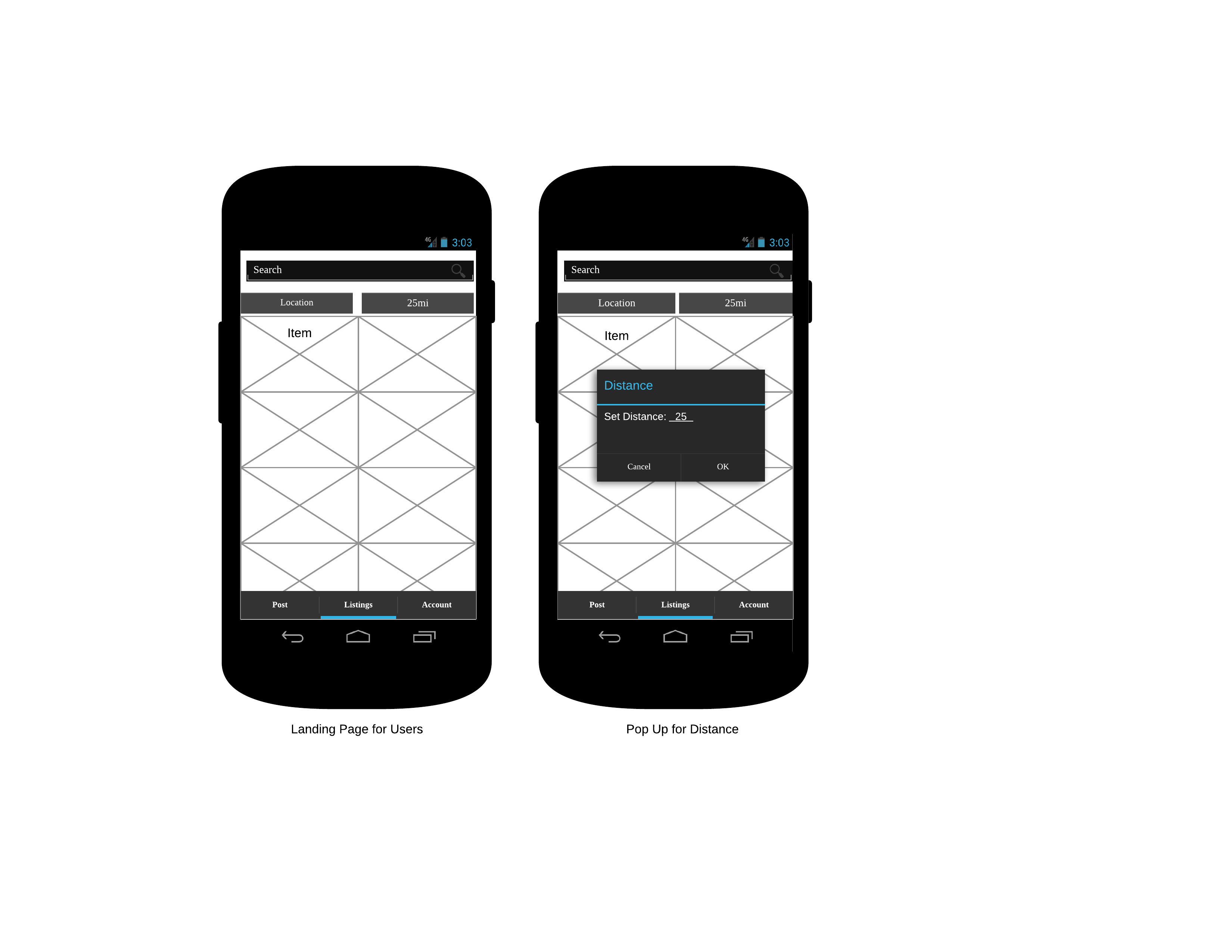
**L6**



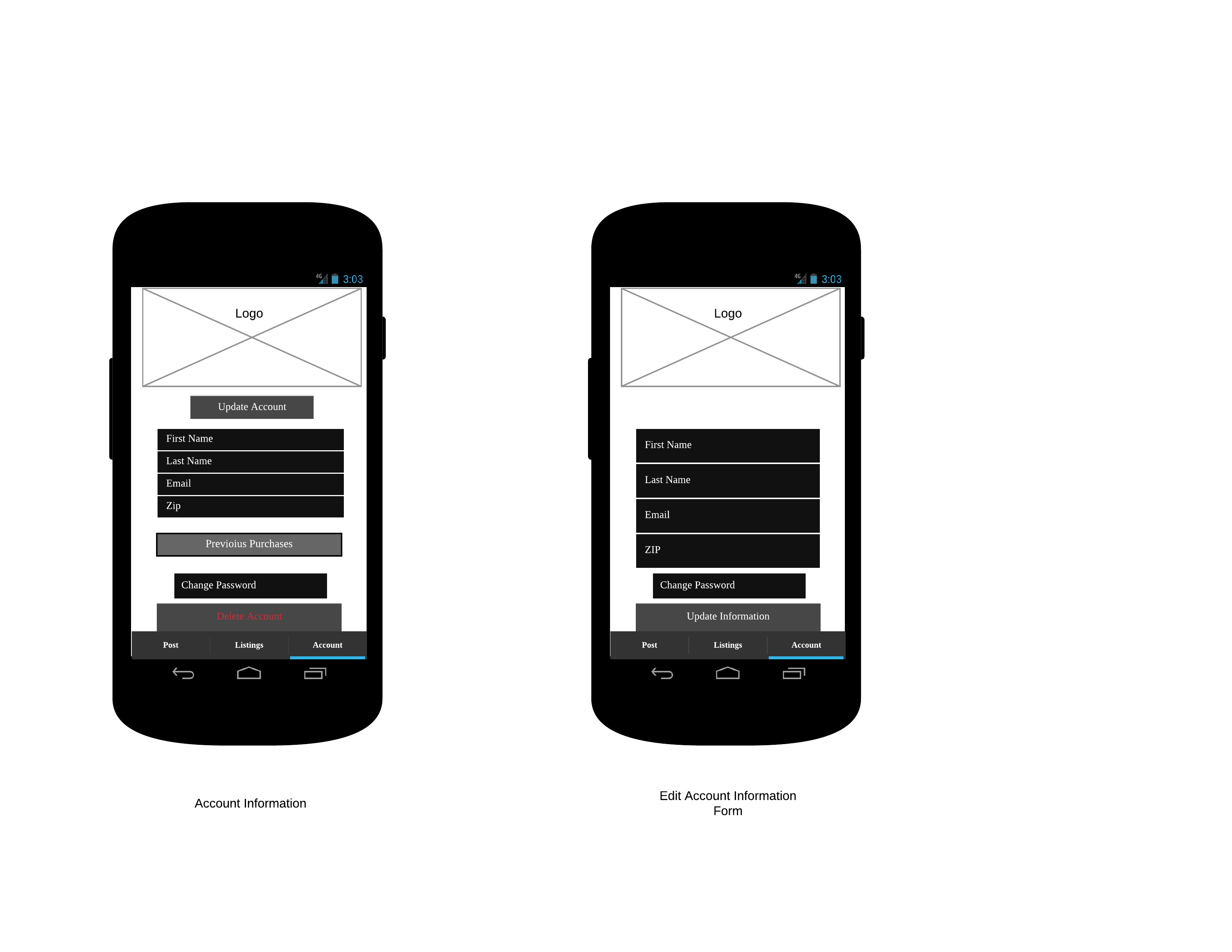
**L7**



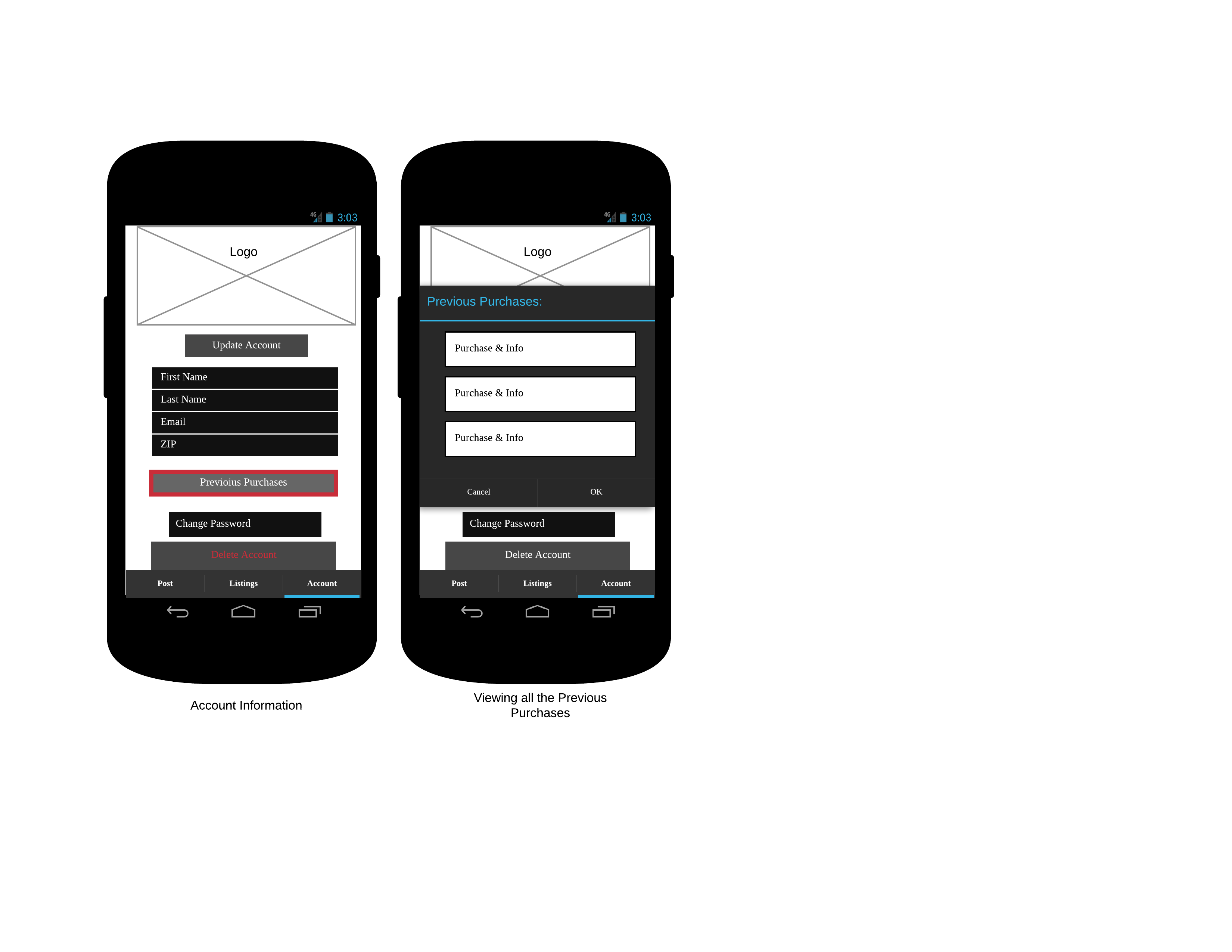
**L8**



**L9**



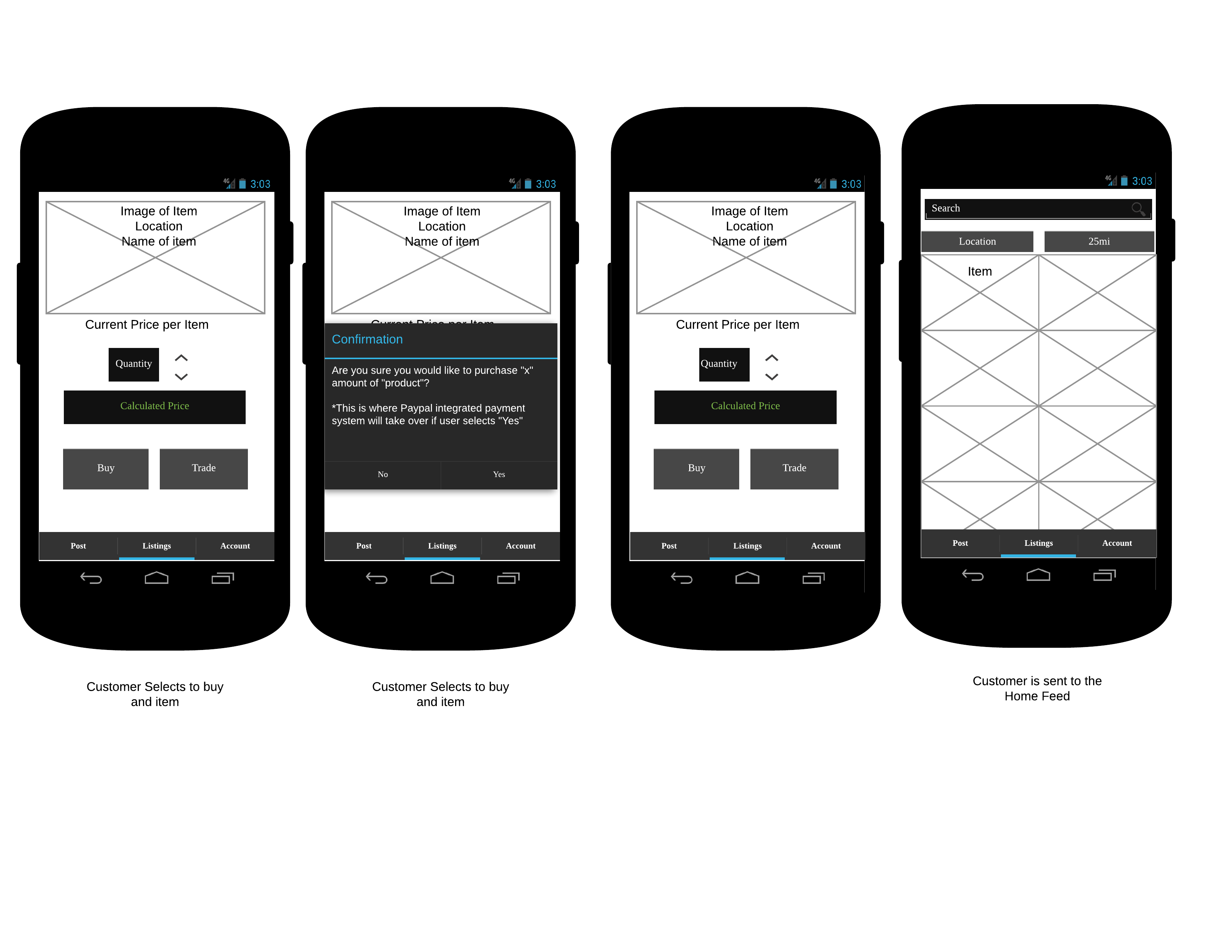
**L10**



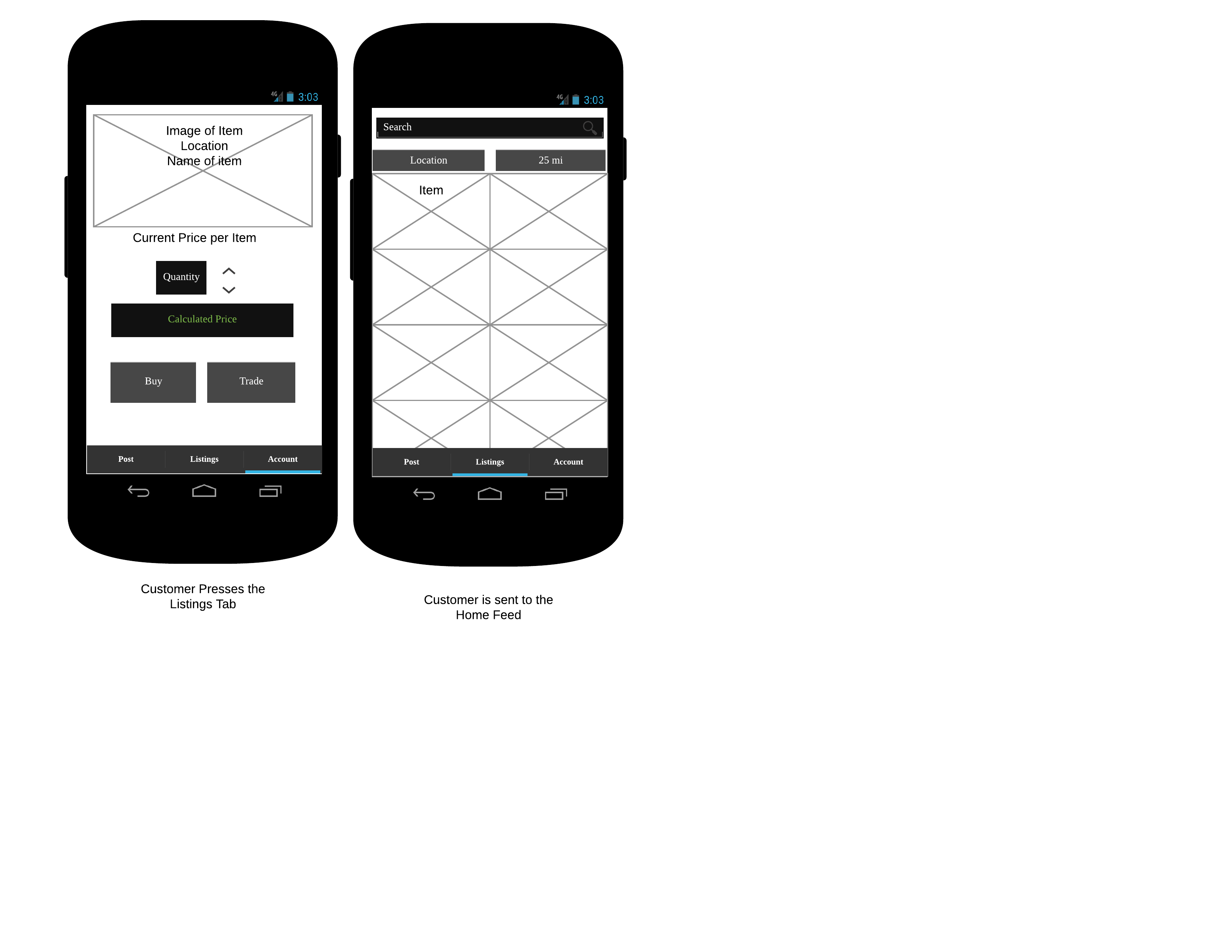
**C1**



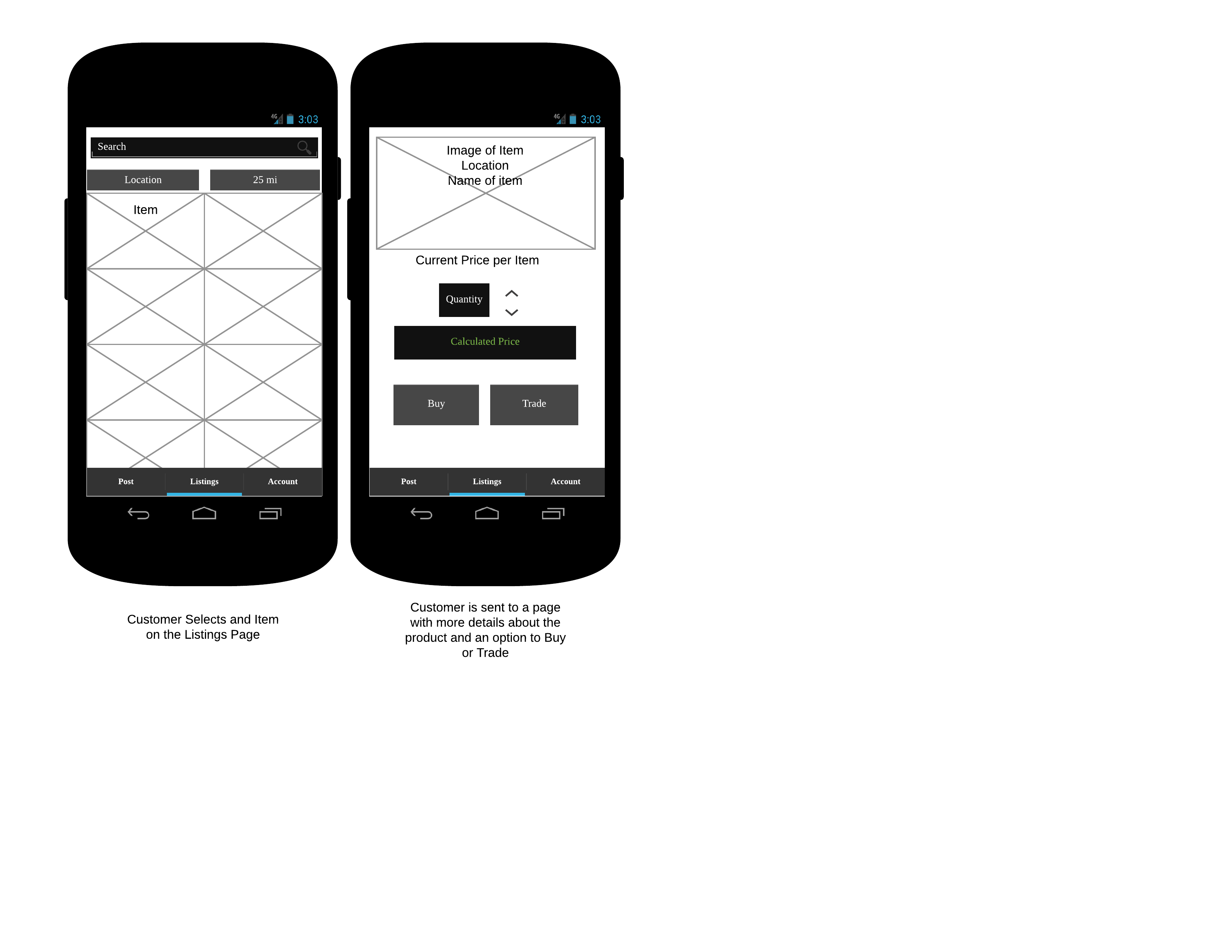
**C2**



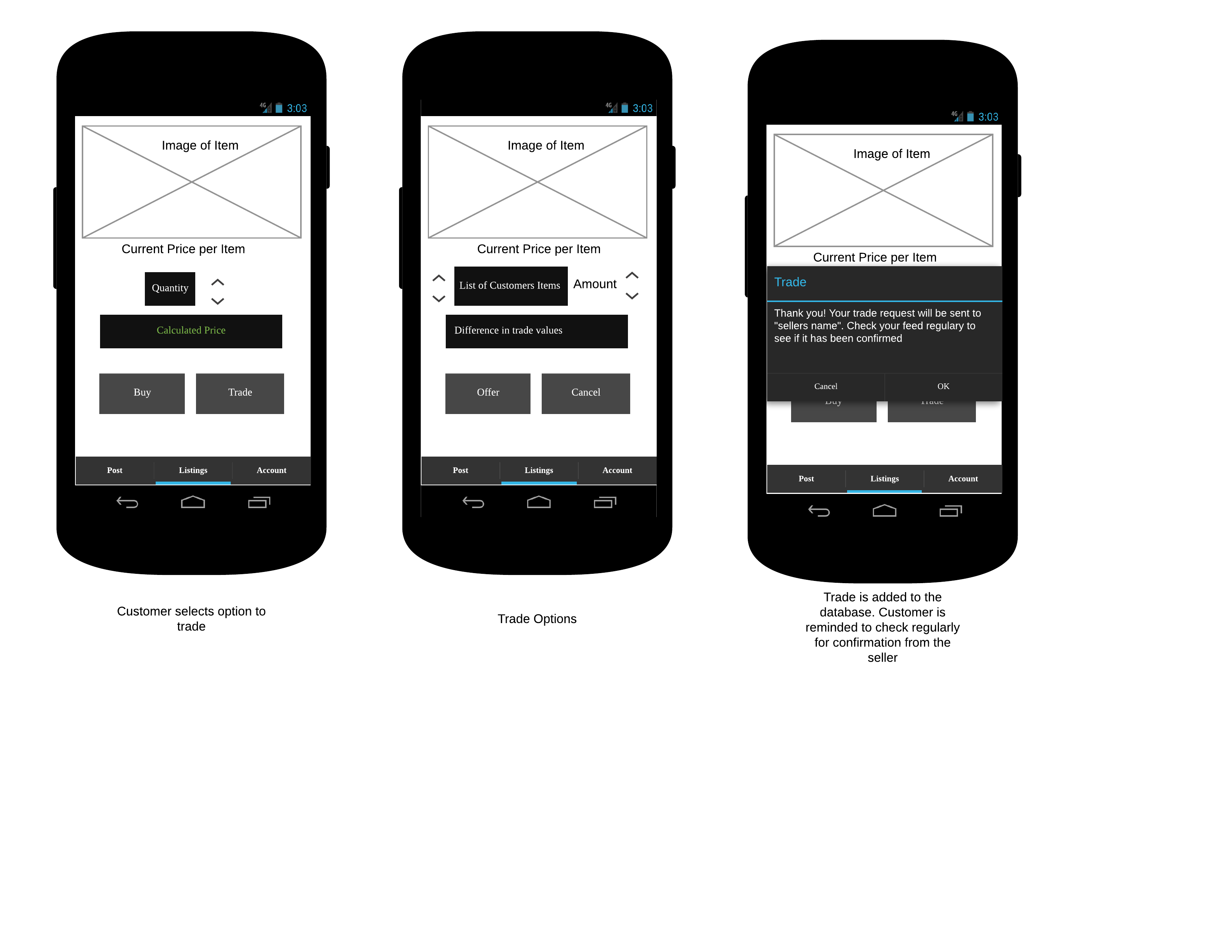
**C3**



**C4**



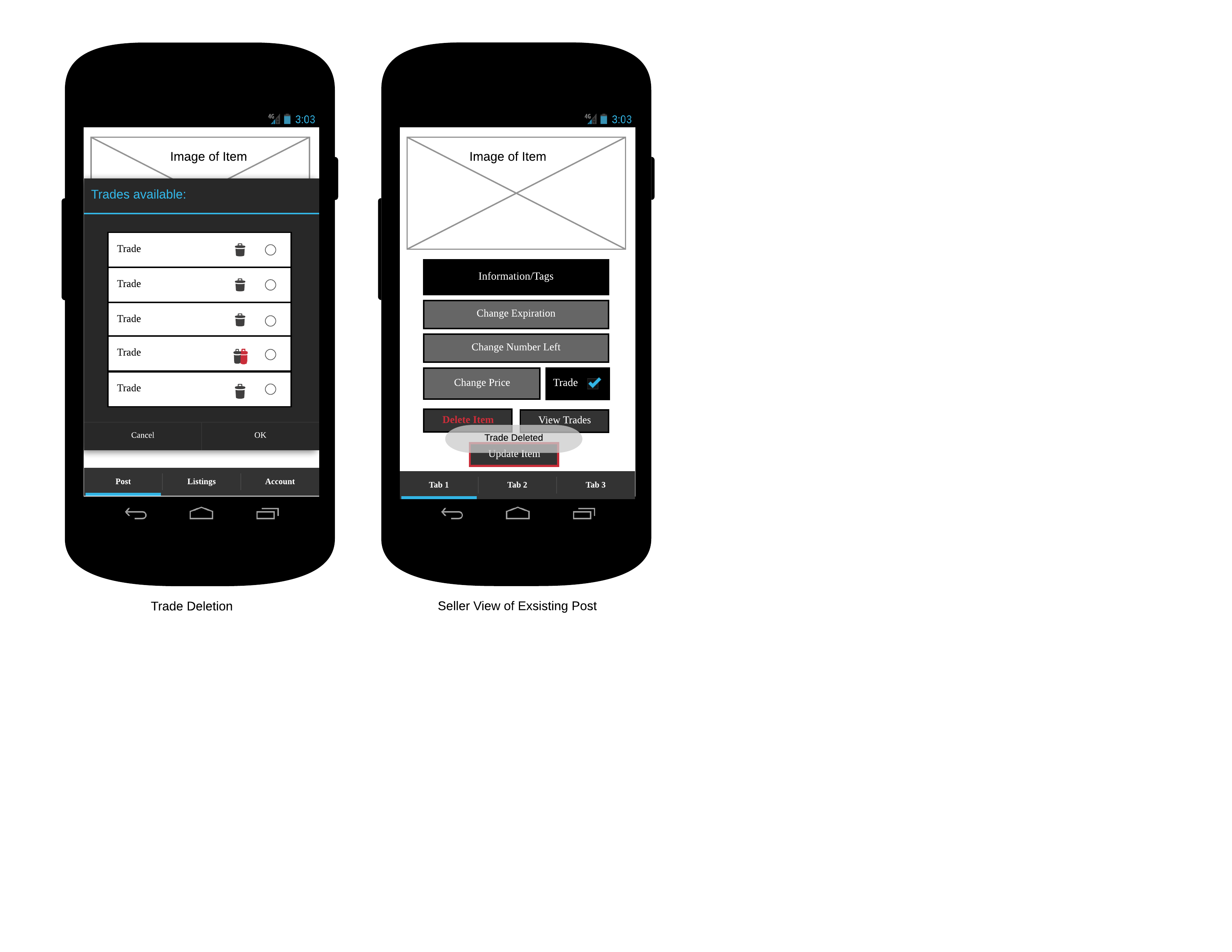
**T1**



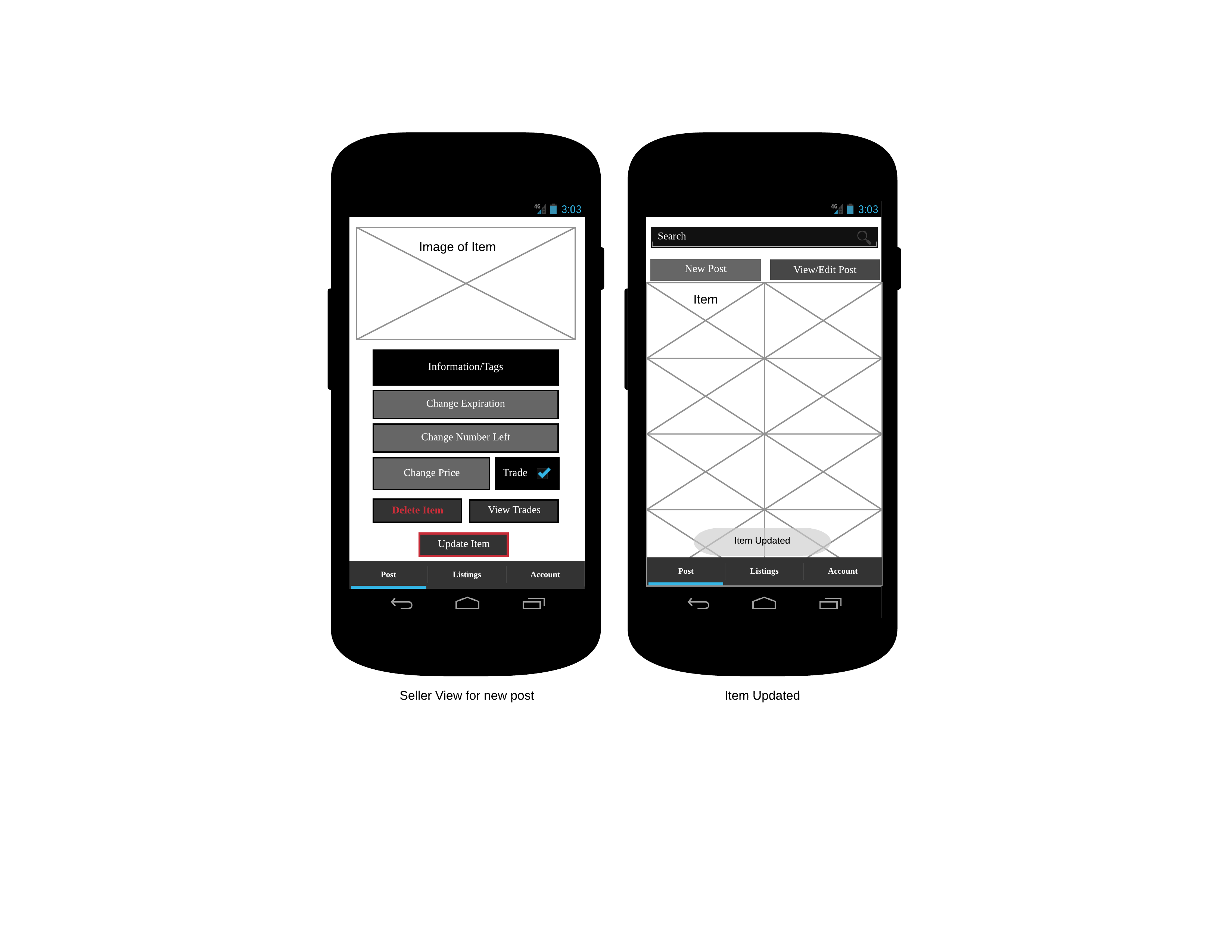
**T2**



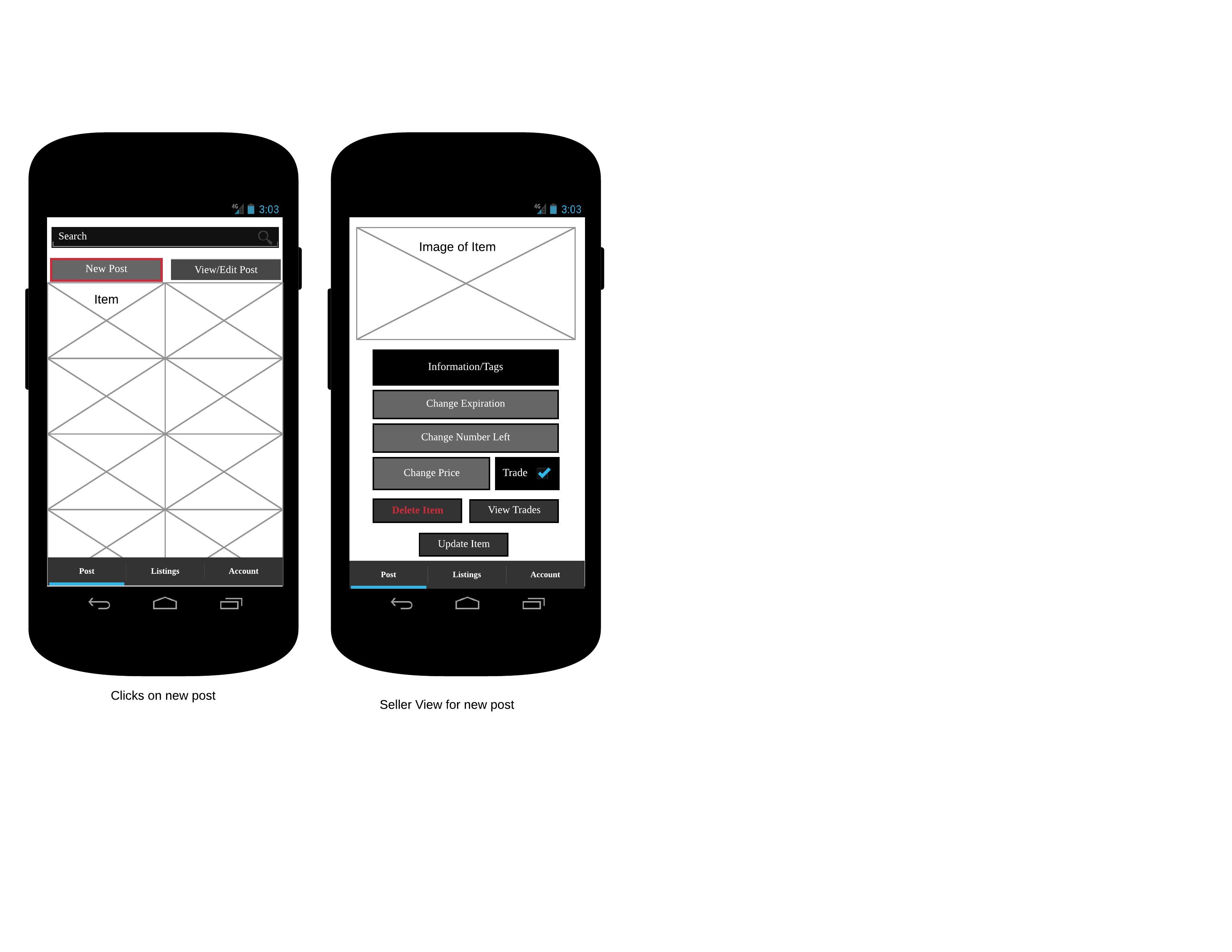
**T3**



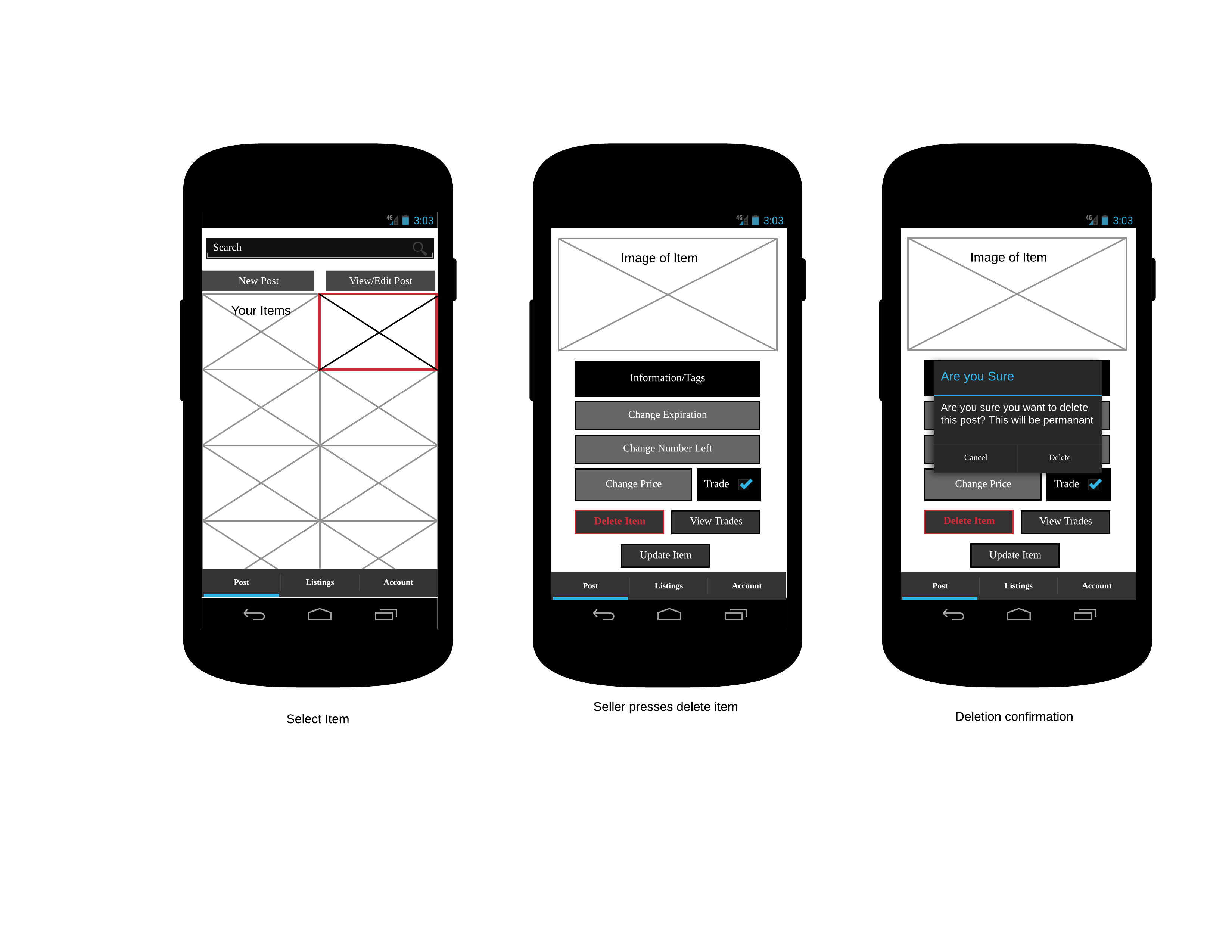
**S1**



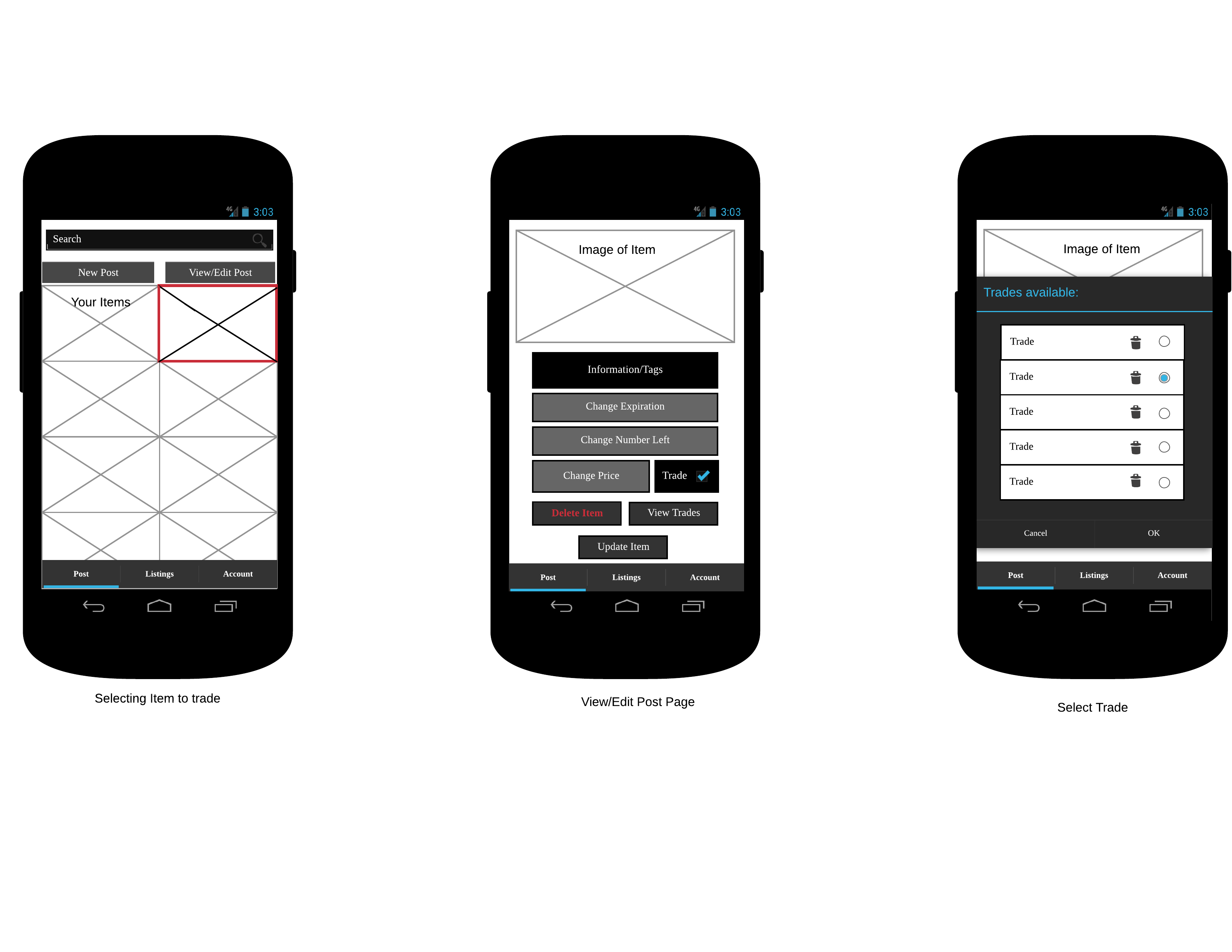
**S2**



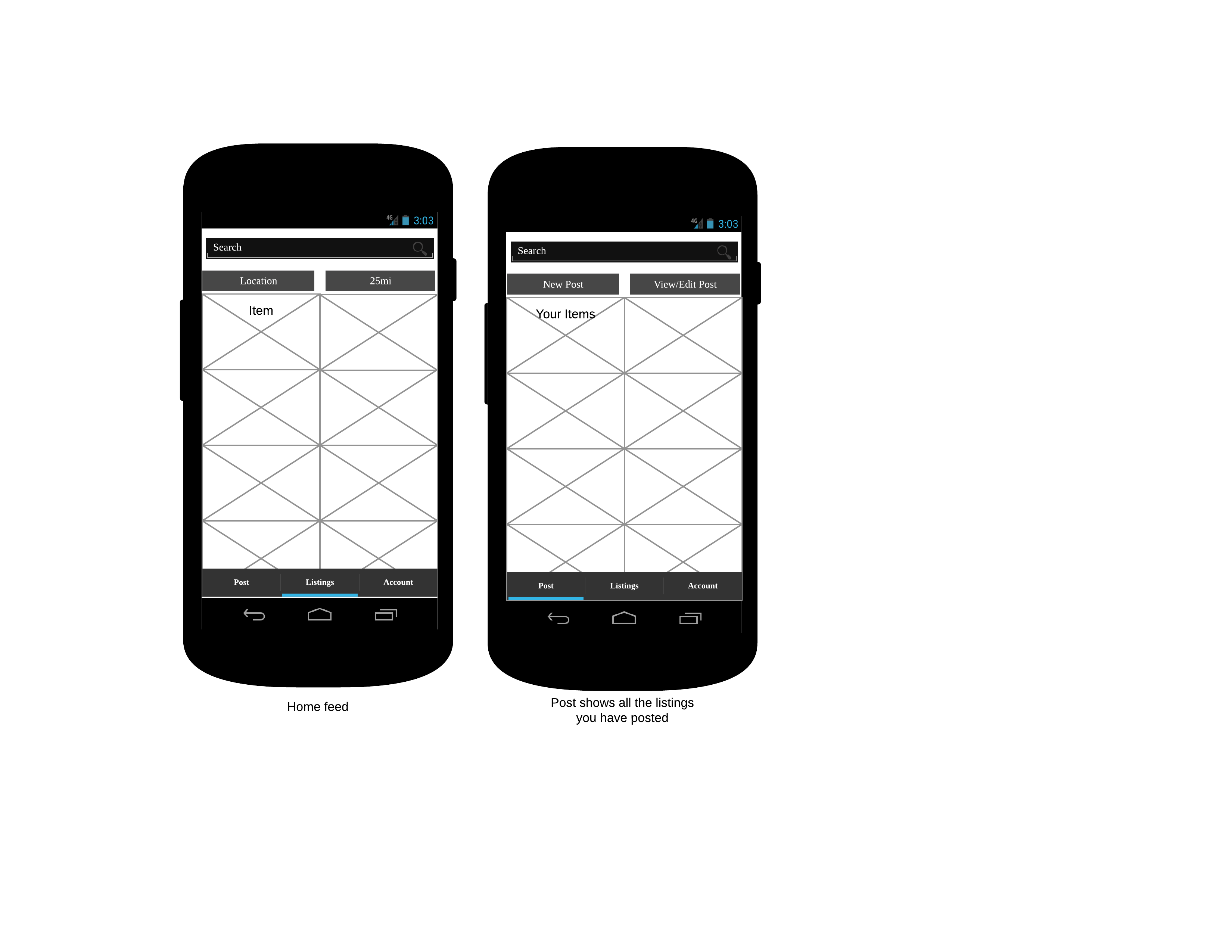
**S3**



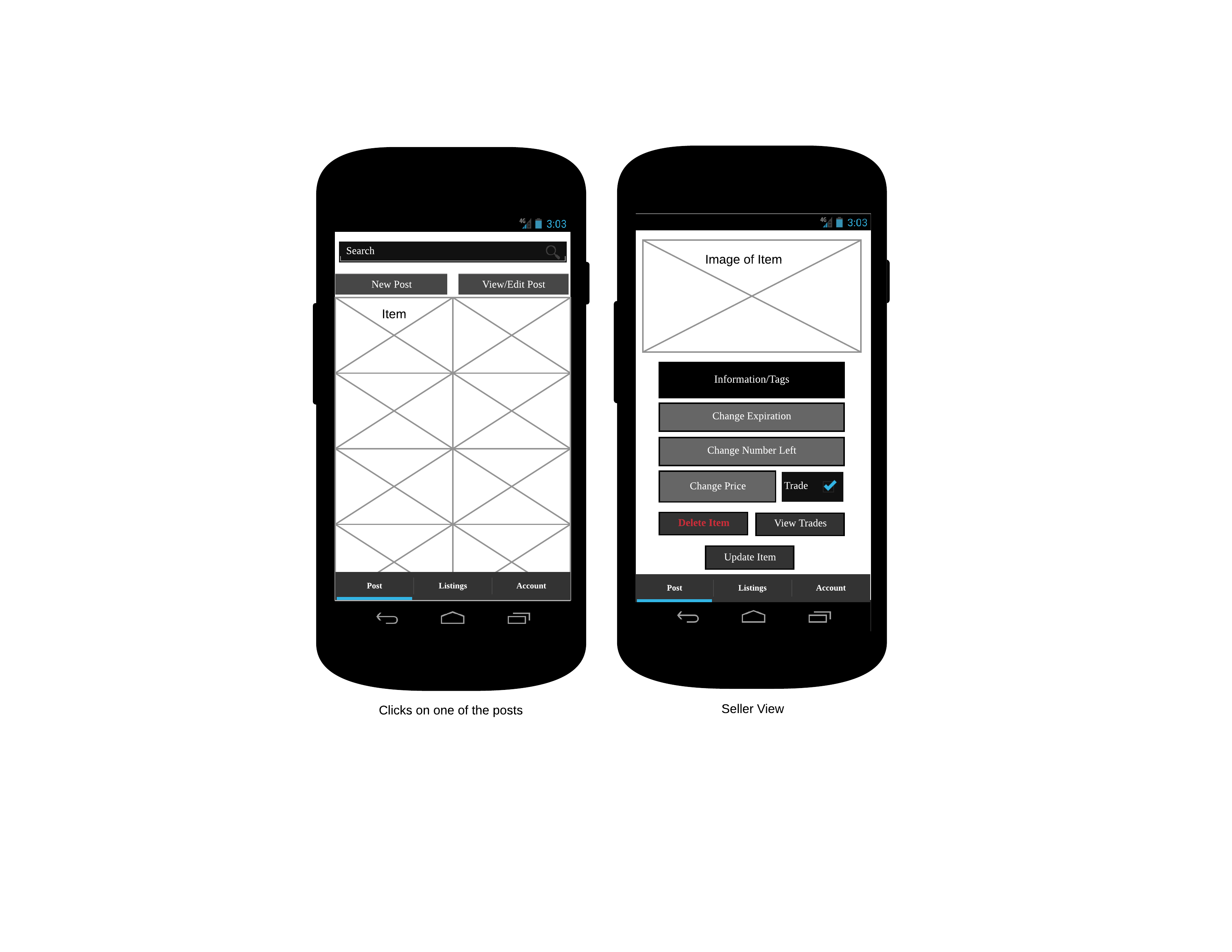
**S4**



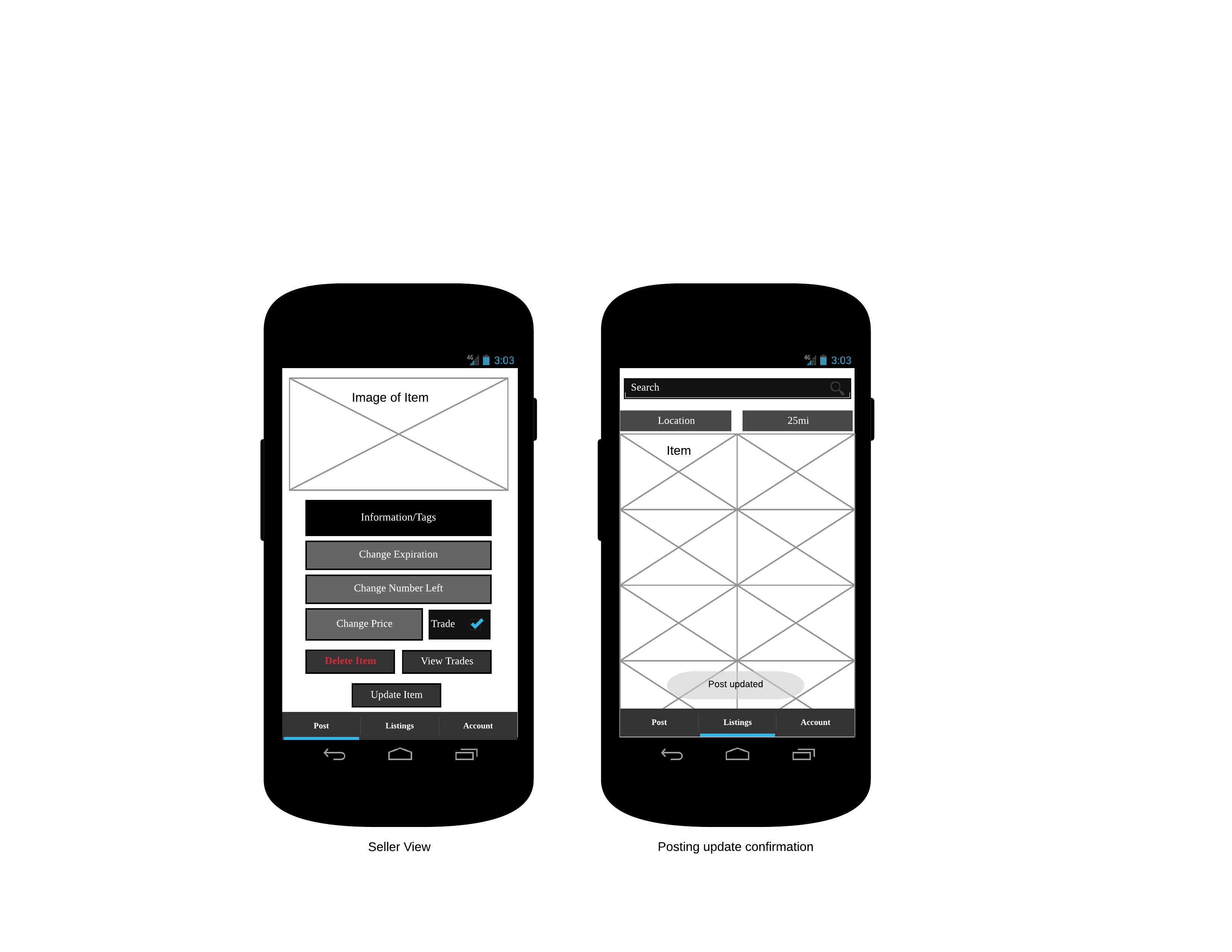
**S5**



**S6**



**S7**



# Non-Functional User Requirements

## Usability

The important function of this application is simplicity. It is important that this application remains intuitive for Users:

* Buy items that other users post
* Sell items that the user will provide to the app
* Trade items that in a way that is simple for both parties
* Browse listings from an unregistered user enticing the users to register

## Accessibility

Since the vision of this application is local, there is currently not a demand to develop the application in many languages and cultures.

* The system will support the English language
* The system will be usable for “non-technical” users
  + Ex. people that have degrees but are not exemplary in computer use
* The system will be user friendly to colorblind users

## Availability

The application should be available at all times. The time in which an item is bought should not have any interference with the system because the system is not in charge of a delivery system. Users instead will meet at a location that is determined between the parties involved. If there are any errors that occur within the system, a notification email will be sent out to all users about maintenance repairs and an estimated time as to when the app will be up and running again. If non-users experience an error with the system, the page will alert them that the system is currently down and undergoing maintenance.

## Documentation and Training

Documentation for this application will include a basic user manual with guides on how to do basic functions within the application, such as creating a new listing or updating a user’s account details. A copy of this SRS document will be provided to the client for future use as well.

There should be minimal training for the average person to use this application. It should follow intuitive smartphone application design choices. The only training necessary for this application will be the database administrator, who will need to have knowledge of SQL and a good understanding of the database itself.

# Non-Functional System Requirements

## Performance

This application is not intended to require a large amount of resources. In its core it designed to be a simple app that is very user friendly, therefore making it very resource intense would not make much sense. It should run smooth and be very responsive, not feeling sluggish or slow. It also should not use a lot of the battery since it is designed to be lightweight and fast.

## Capacity

Due to the limited budget, we will not be able to start out with a large server, therefore our capacity will not be as large as we would want it to be. We will be using the free tier in Amazon Web Services (AWS). This means that our database is limited to 5GB of Standard storage. Standard storage for our application would include things like pictures of items, item data, user data, and transaction data. There would be a limit to the number of GET and PUT requests as well. You would be allowed 20,000 GET requests, and 2,000 PUT requests. Although this is not optimal, this is all we can afford until the application creates enough revenue to upgrade our servers.

## Security

**Access**

A user must be logged in using an email and password to be able to view account information, edit listings, accept or decline trades, and buy and sell items. If they are not logged in, a user will only be able to view listings. A logged in session will begin upon being logged in and expire after a period of inactivity. The session will be validated upon every concerned action.

**Passwords**

Passwords will consist of any string of visible characters and must be at least 8 characters in length. The passwords will be hashed before being stored and each time a login is initiated, the input for the password will be hashed in the same way and compared to the stored hash in the database.

**User Input**

Any inputs into the system by a user will be sanitized before it is used in any query. This will prevent SQL Injection and any other unforeseen issues.

**Other**

Any data problems or issues with user behavior will be dealt with by a database administrator. This can include bugs in data entry on the part of the system, inappropriate content, or dishonest content. These can be removed or fixed directly with SQL.

## Longevity

This application is intended to continue growing in the local area, and then from there hopefully it will grow much larger. For now, it does not have longevity because of our small budget and the price of large servers, but if the community around the application grows, then it will also develop longevity. However, the system can be subject to further development.