University of Nevada, Reno

Department of Computer Science and Engineering

CS 425 Software Engineering

Project Part 2: Specification & Design

Drift - Team 12

Jordan Rood, Fiorina Chau, John Christian Jackson

Instructors: Dave Feil-Seifer, Devrin Lee, Sara Davis, Vinh Le, Zach Estreito

External Advisor(s) w/ Affiliation:

Brittany N Avila - Psychology Department, University of Nevada - Reno

Araam Zaremehrjardi - Grad Student, University of Nevada - Reno

February 8, 2024

1 - Abstract

Drift is a thrifting mobile application as well as a software service specifically for public users to buy and sell thrifting goods (I.e., an e-commerce app for thrifting specifically). Our project is important because it will pave the way for a more sustainable thrifting experience while making it easier to buy/sell, connect with other thrifters, and discover all types of second hand items. The major features that we intend to design and implement are user login/signup authentication, item posting, item discovery and querying, cart use, checkout, and chat functionality. Our thrifting and e-commerce centralized application will provide a virtual thrift shopping experience for browsing for items, posting items to sell on one's profile, and perusing saved items.

2 - Recent Project Changes

The only possible changes since P1 include plans of rescoping to ensure success of meeting functional requirements. These ideas of re-scoping just move a few of our stretch goals to tasks slated for work beyond this semester. These tasks for further development beyond CS 426 include artificial intelligence integration for outfit styling and user customization as well as taking on brick and mortar thrift shops for further roles and advertising features. These changes were needed due to the size of our team and the circumstances we all have with outside factors such as internships, other classes, etc. We noticed that our original aspirations to integrate artificial intelligence and bring in brick and mortar thrift shops may have been too much of a reach; however, we still are going to work towards the goal of the possibility of getting there. These features are nice to have and will definitely be in our minds for the future development and maintenance of Drift.

Additionally, we have decided on using Stripe in creating the payment functionality from check out. This is not a change per say, but rather a decision recently solidified as far as e-commerce integration goes. This decision was needed to move forward with starting the design and implementation around our checkout and payment modules. Stripe was found to be a well documented and a developer friendly API to use and therefore got us to decide to utilize it as a tool for the buildout of our application. Overall, these are the main changes that have recently been made from the last document pertaining to Drift's design and implementation.

3 - Updated Specification

3.1 - Summary of Changes in Project Specification

Drift has undertaken some changes in project requirements, organization, scope, and features since our previous specification document in Fall 2023. One of the major changes was in the rescoping and plans of feature implementation. This includes the making of some previous level two requirements to be pushed back to level three requirements due to the determination of time and commitment restrictions by our team of three. If we were to have an additional developer on the team this may not have happened, but under the circumstances with balancing out commitments to internships, academics, and personal life we had to make this change. We plan to push on with several of our features and functionalities as mentioned last fall; however, the artificial intelligence stylist integration and getting local thrift shops on board with customization application capabilities is likely to be future development outside of CS 426. These changes in the requirements were needed to ensure a realistic set goal for our baseline of functionality that Drift will have.

Some other major changes are mainly technology decisions such as using Stripe's API to help build out orders and payment functionality and some updates to the use cases. This was needed to be finalized in order to move forward with the development of this subsystem. We plan to add a couple more use cases to our diagram as our functional requirements have extended. This will have an effect, due to these modifications, on the traceability matrix that is presented below. Drift ultimately will still encompass its main capabilities of posting an item, searching the Discover page, among several others.

3.2 - Updated Technical Requirements Specification

Drift's technical requirements are separated out into functional and non-functional requirements which allows us to derive an appropriate path for approaching design and implementation of our application. These requirements include the main features and functionalities that Drift will have and behaviors and use cases that the app may encompass. Level one requirements are those that we will have implemented for the finishing of this class. Level two requirements are those that may or may not be implemented due to time constraints. Level three requirements are those that are ideas for future development beyond CS 426 senior projects.

Functional Requirements		
#	Level	Requirement Description
FR01	1	Drift shall allow the user to create an account.
FR02	1	Drift shall allow the user to login to their respective account with a username and password.
FR03	1	Upon signup, a user's password will be encrypted for secure transit to and storage in Drift's database.
FR04	1	Users will be able to view all publicly posted items on the Discover page.
FR05	1	Users shall be able to post items with pictures, title, description, etc.
FR06	1	Drift shall allow users the ability to logout.
FR07	1	Users shall be able to proceed to the cart with the ability to add and remove items from the cart itself.
FR08	1	Users shall be able to checkout from the cart with help from the Stripe API.
FR09	1	Drift shall allow for searching items up for sale based on keywords.
FR10	1	Users shall see all the items they have posted up on their profile page.
FR11	2	Users shall be able to reset their password (via email) from the login screen.
FR12	2	Users shall verify their account via an email.
FR13	2	Users shall be able to chat with other users.
FR14	2	Drift will have an orders screen to display current and/or historical orders placed by the user.
FR15	2	Drift shall have a dark and light mode setting in which the user can toggle to switch app color themes.
FR16	2	Users can save items that they would like to refer to later on, and these saved items will be displayed in the saved items page.
FR17	2	Users will be able to view items in more depth and add to their cart if desired.
FR18	2	Drift shall have a gamification aspect that shall display to the user the seller's rating.
FR19	3	Drift shall encompass a reporting system for the users to report

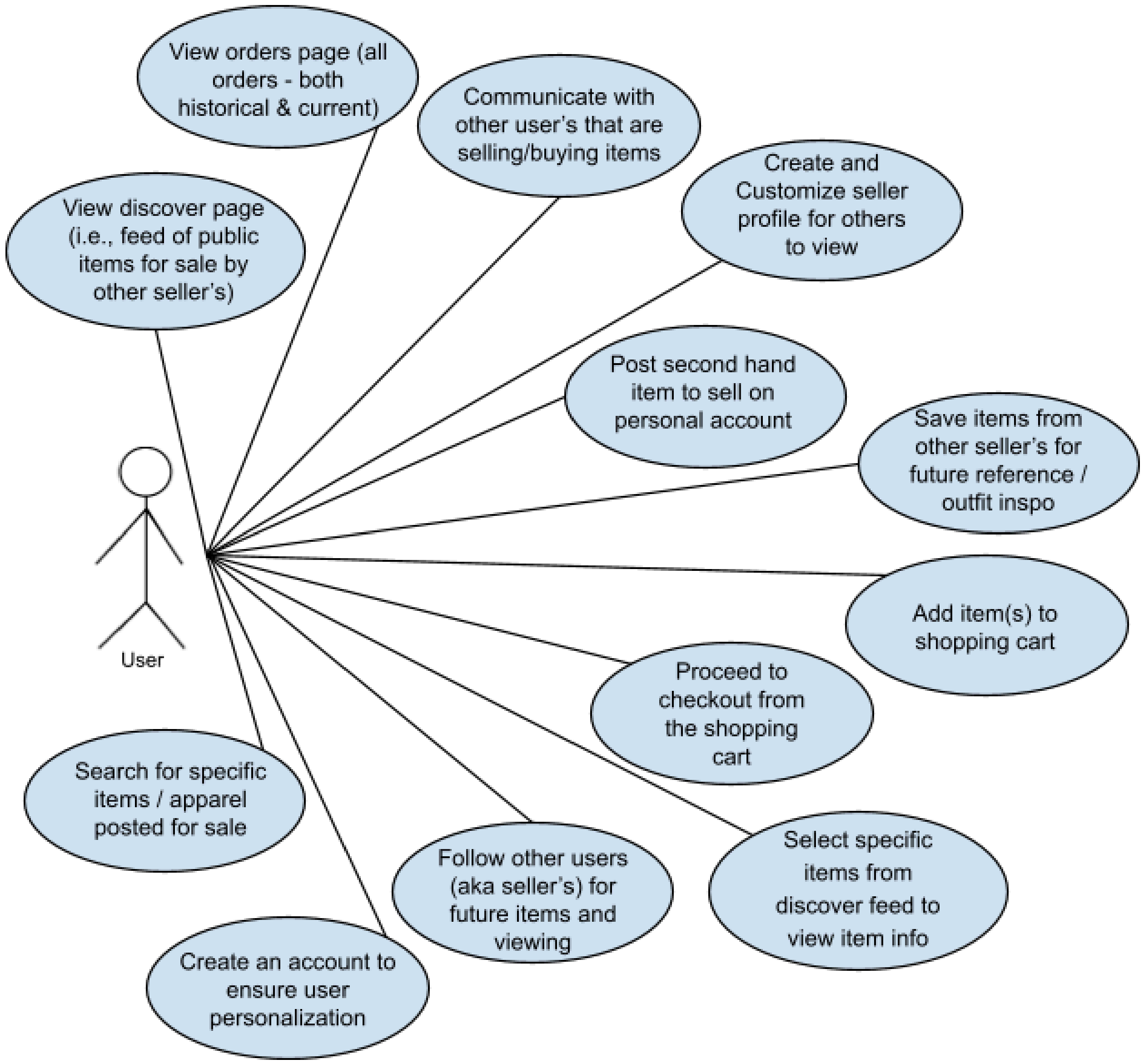
		malicious or inappropriate activity.
FR20	3	Users shall be able to follow other users so that their product for sale will be on their Discover page more often.
FR21	3	Drift shall allow users to sign on via Single Sign-On (SSO).
FR22	3	Drift shall have a notification system that pushes updates on orders, recommended items, chats, and more
FR23	3	Drift shall have two factor authentication for user authentication and security measures.
FR24	3	Drift shall have an artificial intelligence feature integration for outfit styling and user customization.
FR25	3	Drift will have an elevated role for brick and mortar thrifting shops to be on the app and have more freedom in advertising their shop.

Non-Functional Requirements		
#	Level	Requirement Description
NFR01	1	The user interface of Drift shall be implemented with a React Native frontend, TypeScript middleware/API, and a MySQL backend.
NFR02	1	Drift shall be compatible for both IOS and android mobile devices.
NFR03	1	Drift shall utilize a MySQL database to store all user, item, order, and other information pertaining to the usage of the application.
NFR04	1	Drift shall follow accessibility rules and regulations as they apply to mobile applications.
NFR05	1	Drift shall provide a clean, elegant user interface that follows design best practices.
NFR06	2	Drift shall be responsive across several devices.
NFR07	2	Drift shall be able to handle varying thresholds of user activity and interactions while still providing fast, efficient performance.
NFR08	3	Drift shall be able to adapt with space requirements as the amount of users interacting with the application fluctuates.
NFR09	3	Drift shall incorporate localization features to adapt based on users location (can make it easier to find thrift stores or thrifting communities in their area).

NFR10	3	Drift shall meticulously reduce the amount of storage being used per user on their mobile device due to the small amount allotted on such portable devices.

3.3 - Updated Use Case Modeling

Use Case Diagram:


View orders page (all orders - both historical & current)
Communicate with other user's that are selling/buying items
Create and Customize seller profile for others to view
View discover page (i.e., feed of public items for sale by other seller's)
Post second hand item to sell on personal account
Save items from other seller's for future reference / outfit inspo
User
Add item(s) to shopping cart
Proceed to checkout from the shopping cart
Search for specific items / apparel posted for sale
Follow other users (aka seller's) for future items and viewing
Select specific items from discover feed to view item info
Create an account to ensure user personalization

Detailed use case descriptions are below mapped to the corresponding ones with a more in depth description on the case. These describe the intended and planned usage and behavior that the user will experience when using the Drift application.

Use Case Descriptions		
UC01	ViewDiscoverPage	The user should be able to view the discover page which will illustrate a feed of items for sale based on recommendations to specific users. This feed should be refreshable with new posts populating upon action by the user. Finally, the items on this feed will have a preview picture, price, and size to show the user.
UC02	UserCommunication	Users will have the ability to communicate with other users (buyers/sellers) via direct messaging. This will likely be a page that is populated with conversations between the user and others. We see this as a use case because an avenue of interaction is needed if any individuals have questions about products or anything else to specific sellers.
UC03	ProfileCustomization	A user will be able to create their "closet" from the ground up along with differing functionalities for customization of their seller profile. These customization capabilities are envisioned to be profile picture setting, bio creation, etc. This will allow users to express themselves how they want to buyers which will create a more interactive experience.
UC04	PostItems	Actors will be able to post items on their profile, also known as their "closet", for others to see and be able to purchase. This will be similar to other experiences for posting things with other applications such as through a sort of form like input collection. This will give the user fields to add to with differing types of media such as pictures, title, description, size, condition, etc.

UC05	SaveItems	An individual using our application will have the functionality of saving items posted by other users for future reference. This is a great capability because there are a multitude of reasons for a user to need to save an item. For example, the user may not want to buy it right away or they just want to utilize it for future reference to find something similar.
UC06	ShoppingCart	A user will be able to add discovered item(s) to their shopping cart (this will be another page within our web application) which holds items the user is intending to buy soon. That is the key difference between saving an item in that when an item is in the shopping cart, that user is staging it for purchase, whereas a saved item is just for future reference.
UC07	ItemCheckout	Furthermore, the next step after a user adds items to their shopping cart is to proceed to checkout from the shopping cart page. If there are no items in the users cart, then proceeding to checkout should be disabled. Otherwise, the user can continue when desired to checkout for billing, shipping, and order confirmation for the various items.
UC08	ItemSelection	The user can browse through the discover feed page and select items to view more information pertaining to that specific item. This selection from the search of the discover screen will redirect the user to the page for that seller's item.
UC09	FollowingUsers	Similar to the interaction with social media applications, we expect the user will benefit from the use case of following other user's for future reference and viewing of items. This is for users who liked or bought previously from the seller because they like the

		items that they post. This adds to the ease of use and interaction built within the application.
UC10	SearchItems	If a specific item or apparel is desired, the user will be able to search for that within a search bar. This bar will take user input and pull up a feed of items most relevant to the search query for the user to then peruse through.
UC11	ViewOrders	A Orders page will encompass both historical and current orders for the user to view a listing of for reference. This use case is for the main orders view where a link to this page will lie in the drawer for the user to navigate to a peruse.
UC12	CreateAnAccount	Users shall have the capability to create their own personal account through a sign up page. This will allow for much more personalization that is custom fit to the user of the account created. This use case must also imply login and logout.

Requirement Traceability Matrix

Provides the mapping between use cases and functional requirements. Note - the intersections and relations are depicted in black.

Requirement Traceability Matrix												
	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11	UC12
FR01			■									■
FR02			■									■
FR03												■
FR04	■			■				■		■		
FR05			■	■						■		

FR06												■
FR07						■					■	
FR08						■	■					
FR09	■							■		■		
FR10			■	■				■				
FR11												■
FR12												■
FR13		■										
FR14			■	■		■	■				■	
FR15			■									
FR16			■		■			■		■		
FR17	■				■	■	■	■			■	
FR18			■						■			
FR19		■										
FR20	■	■						■	■			
FR21												■
FR22		■	■		■						■	
FR23												■
FR24			■									
FR25			■									

4 - Updated Design

4.1 - Summary of Changes in Project Design

Drift's project design has moved forward with most if not all of the high-level and medium-level designs mentioned in our Design Document written last Fall. The biggest changes were done to our database design. The changes are highlighted in yellow under Database Tables. These changes were added as we saw a need to add more attributes that further helped users to find the items they need or that were mandatory for the task at hand.

4.2 - Updated High-Level and Medium-Level Design

Context Model

The high level design components are illustrated in our context diagram below (Fig. 1). As shown, there are going to be six subsystems that handle differing functionalities meant to be modularized for efficient and clean code which will work together seamlessly. These systems are the main components that will make up the Drift application with the underlying program units that fall below as the working components of each respective system.

The query system has the main purpose of fetching from the database a multitude of records from tables; these include the users table, items table, etc. which is illustrated more in the following sections. The orders system is for functions surrounding all things to do with order processing. The accounts system is the subsystem that will have capabilities surrounding the adding, deleting, and managing user accounts data. The view management system will be for controlling the fetching and rendering of data for the frontend UI and views that are user facing. The security system is for modules that will keep the application secure and only allow authenticated users into their own accounts. The items management system has a purpose of encompassing the middleware capabilities for managing product records and information.

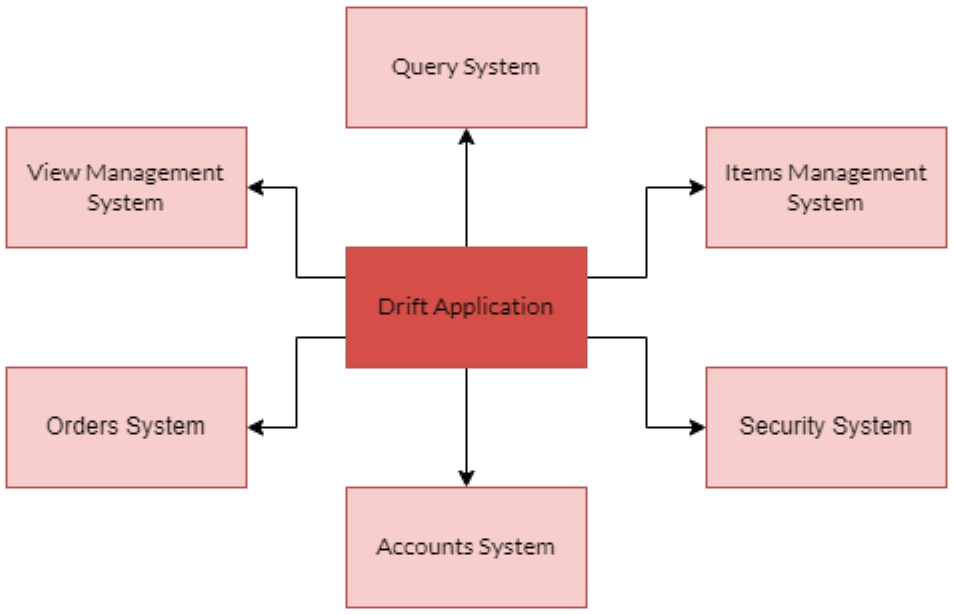


Figure 1: The context model of the Drift mobile/web application. The application is planned to use six interacting systems/functionalities to communicate with each other to provide all of the capabilities needed for our software.

Program Units

Drift, our thrifting mobile and web application, is a non-object oriented project solution which means the software will be designed with modules, functions, and procedures as such program units. The tables that follow illustrate the modules and functions, in a high-level view, that will be working together to interface and act as a cohesive unit that makes up Drift, our thrifting web application. Furthermore, these functions and modules will be implemented through a TypeScript middleware connecting a MySql database backend to a React native frontend.

Routing and Organization

Further, our application will have routes using the express node.js (NPM) library. Most will be GET and POST requests that utilize the functions within all the subsystems described above. The high level design for the middleware will be a directory of folders containing routing which will be separated by system as per the responsibility of the route. For example, a route for adding a new product, after an event trigger due to a form submission which occurs on the frontend, will call functions likely from the Items Management System. These modules all have responsibilities in the same realm and thus will be encapsulated in the same directory for organization and readability. Ultimately, all the routes implemented are the main controllers and end points for our application. The subsystems are modules that will encapsulate the functional program units illustrated on the coming pages, and make our application operate as needed.

Navigation

The navigation menu design will consist of the Discover, Post, Saved, Messages, and Profile pages. These will be the main scaffolds and containers for the program units to be integrated into with them being the overarching controllers and views of the application. The navigation bar will be found on the bottom of the application (illustrated in the User Interface Design section below). The query system and view management system will be of high use in navigation functionalities. The fetching of information and rendering of specific views as triggered by the users is a significant function of our application, and therefore is one of the first steps in our application development. The navigation will likely be one of the sectors that use most, if not all, subsystems. Now, take a look below to find each subsystems table of program units with their respective details and specifications.

Table 6: This table shows the orders system along with its corresponding program units.

Orders System	The main purpose of the orders system will be to keep order records up to date through manipulating the products/items and orders tables within the database. This will allow for efficient capabilities encompassed around user ordering and purchasing.
InsertNewOrderInfo(Int OrderID, Int UserID, String Name, String BillingAddress, String ShippingAddress, Int ItemCount, Int OrderStatus [ItemID] Items Float TotalShippingPrice Float SalesTax Float TotalPrice Int TrackingNumber)	This method's responsibilities include insertion of a new order record which is set into action from a checkout event. The checkout form submission by a user will result in a call to this function in which a call to the database is constructed to add and modify the correct tables (i.e., items and order history tables). The input will be the foreign key item ID and userID that ordered, time, date, and order status. The returned output will be the inserted row which was added to the orders history table. This will help with keeping the most up-to-date and accurate data regarding previous order history that the query system will use per user. An exception for this method could be triggered upon invalid order input from the user. This will be mitigated by user input validation checks throughout our application.
UpdateItemStatus(Int newStatus)	This method is for updates to the database after an item has been processed and purchased. It will be the main point of communication between the frontend checkout functionality and the database respective updates. The input for this will be the corresponding item ID which will be found to then update the sold status field. The output will be the updated item record row. An exception for this could be if there is an API error in which a status input is invalid. If this occurs an error message will be thrown with the application continuing gracefully after.

ReturnOrder(Int OrderID)	This method has the functionality of changing the orderStatus field along with making function calls to the UpdateItem function in the Items Management system. The input to the method is the OrderID so that a get call will occur for obtaining the record along with item records based on the foreign key Order ID in the items table. The items need to be found so that the sold status and order ID fields can be modified correctly. The output of this function will likely be the order record that depicts the modified order status to “canceled”. The exception or interrupt for this will occur if the check of order status depicts that the order holds a “canceled” status already.

Database Tables

The Drift application will have a MySQL database backend for storing all types of data for user profiles, items posted, orders made, as well as a multitude of other data points. The database is planned to have tables for users, items/products, and orders. Below are tables for each to depict the schema and field description for such tables.

Table 7: This table shows the fields and descriptions for the users table that will be in our database.

Users	
Schema	Description
UserID <Int>	This will be the primary key that is a generated ID incremented by one for each user.
FirstName <String>	The first name of the user signing up on Drift.
LastName <String>	The last name of the user signing up on Drift.
Username <String>	This is the username string that is specified by the user signing up.
EmailAddress <String>	The email address of the user registering a new account. This can be used as a way to log in and a point of contact for the user.
PhoneNumber <String>	An optional field for the user's phone number as a way of logging in for the user. This can also be used as a point of contact.
Password <String>	The password specified by the user upon registration of a new account.
Listings <ItemID>]	This field will keep track of all the items the user is selling
Orders <[OrderID]	This field will keep track of all of the user's orders
SavedItems <[ItemID]	This field will keep track of all the items the user has saved
SavedFolders <[SavedFolderID]>	This field will keep track of all the folders the user created to save their items

Table 8: This table illustrates the schema that falls under the items table. This table has fields that are all about an item.

Items	
Schema	Description
ItemID <Int>	This will be the primary key that is a generated ID incremented by one for each item.
UserID <Int>	This is a foreign key which stores the user id of the account that posted the item for sale.
Title <String>	The item title of the product posted by a user for sale.
Description <String>	The description of the product that a user is posting for sale.
Photo <File>	This is the attachment media file of the item being posted.
Size <String>	This is the size of a posted item. This field is optional depending on the nature of the item being posted.
PostDateTime <Date>	The recorded time and date as generated at the time of posting.
Condition <String>	The condition of the item being posted as determined by the user. This can have a range of descriptions such as fair, good, other, etc.
SoldStatus <Boolean>	This is a boolean value that is true if an item is sold and false if an item is still up for sale. The default value upon creation of a new item is false.
OrderID <Int>	This field is to store the corresponding order ID for when the item is sold. This field will be null until the soldStatus updates to true in which case the InsertNewOrder() method will be responsible for updating this field.
Category <String>	This field will store the broadest category that the item fits within.
Subcategory 2 <String>	This field will store the 2nd broadest category that the item fits within.
Subcategory 3 <String>	This field will store the 3rd broadest category that the item fits within.

Color <String>	This field will store the main color of the item which will help with filtering.
Hashtags <[Strings]>	This field will store a list of strings (hashtags) that will help the item be more accurate in search results.
Brand <String>	This field will represent the brand of the item being sold. If not indicated, then the brand will be "Other".
Shipping Price <Float>	This field will represent the shipping price of the item being sold.

Table 9: This table depicts the orders table from our database that will encompass the following fields.

Orders	
Schema	Description
OrderID <Int>	This will be the primary key that is a generated ID incremented by one for each order placed.
UserID <Int>	This is a foreign key which stores the user id of the account that has placed the order.
CustomerName <String>	This is the name of the customer that will be receiving the item(s) in the order.
BillingAddress <String>	This field is for storing the billing address of the user paying for the order.
ShippingAddress <String>	This is for the address to which the order is going to be shipped.
ItemCount <Int>	This field will populate with the amount of items in an order.
OrderStatus <Int>	This field will represent an enumeration that shows the status of the customers' order
Items <[ItemID]>	This field will represent all the items within the order through a list of the items' ItemID.
TotalShippingPrice <Float>	This field will represent the total price of shipping for the order which will be totaled from each item's individual shipping price set by the seller.
SalesTax <Float>	This field will represent the sales tax based on the items ordered.
TotalPrice <Float>	This field will represent the total price of the order which includes tax, item prices, and other fees.
Tracking Number <Int>	This field will represent the order's tracking number through the carrier.

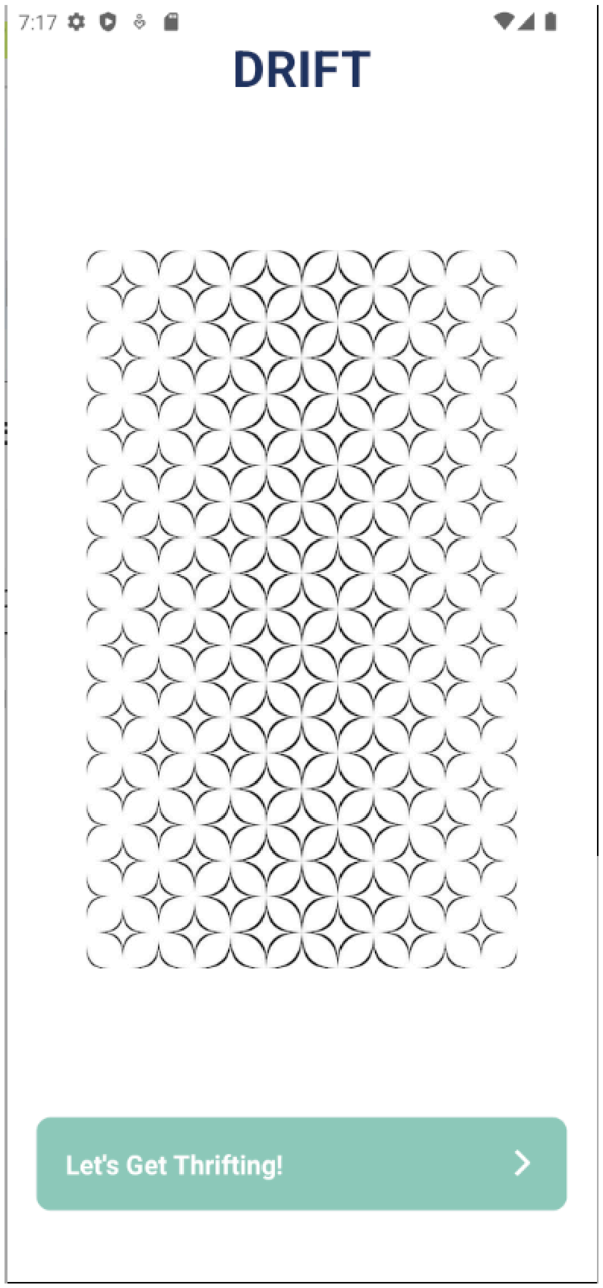
Table 9: This table depicts the saved folders table from our database that will encompass the following fields.

Saved Folders	
Schema	Description
savedFolderID	This will be the primary key that is a generated ID incremented when a new folder has been made.
UserID <Int>	This is a foreign key which stores the user id that created the folder.
FolderName <String>	This is the name of the folder.
Photo <File>	This is the attachment media file for the cover of the folder.
Items <[ItemID]>	This field will represent all the items that the user saved to the folder.

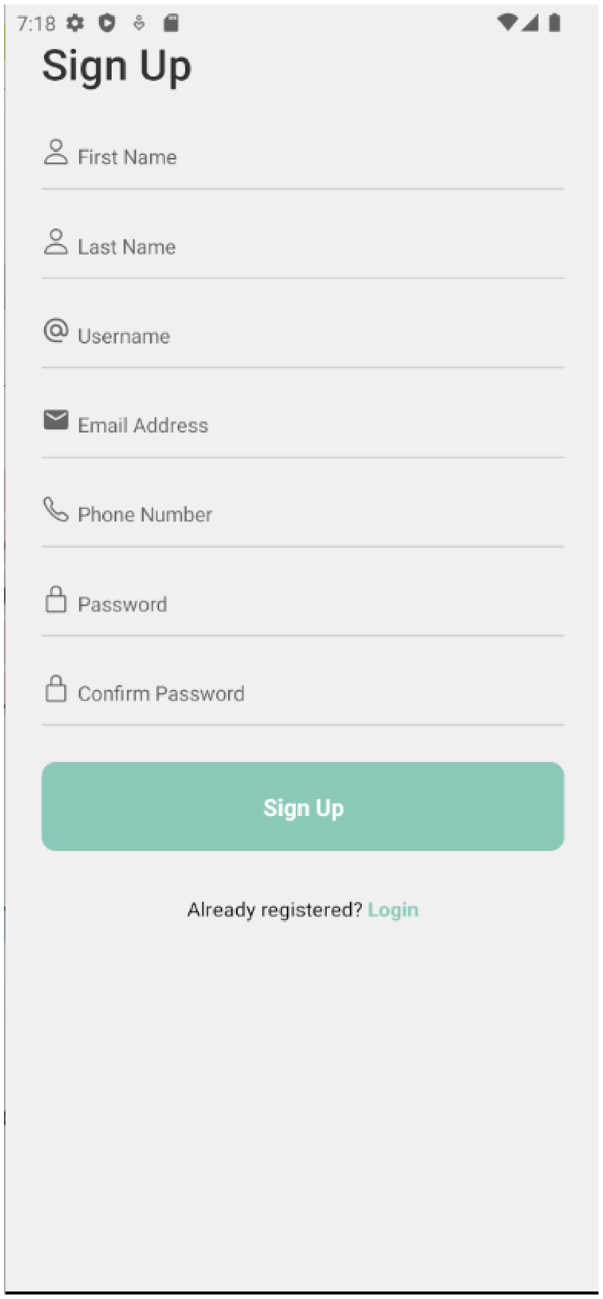
4.3 - Updated Hardware Design

Drift does not currently encompass any hardware components.

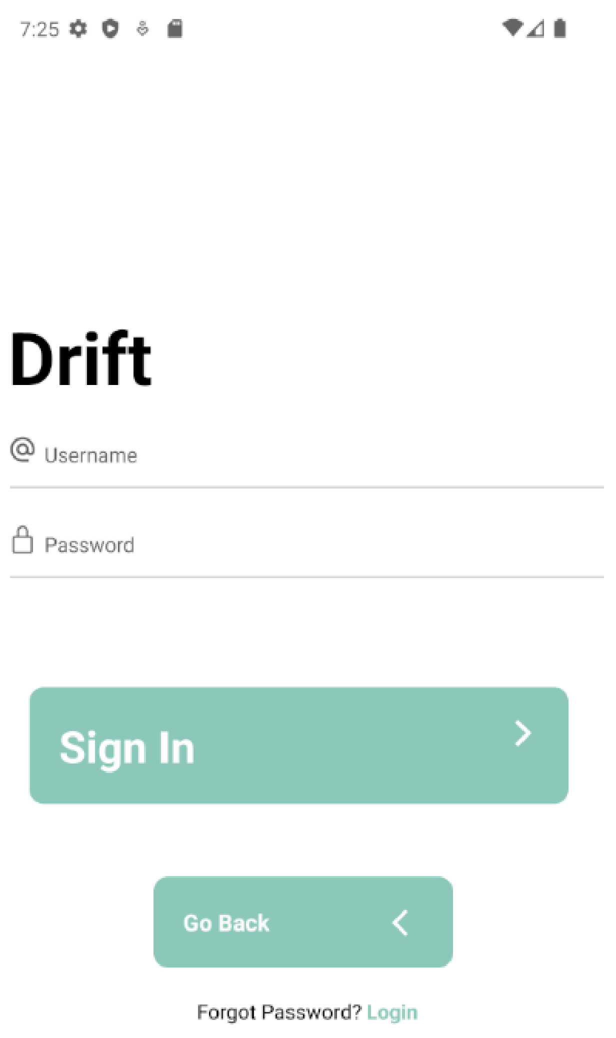
4.4 - Updated User Interface Design



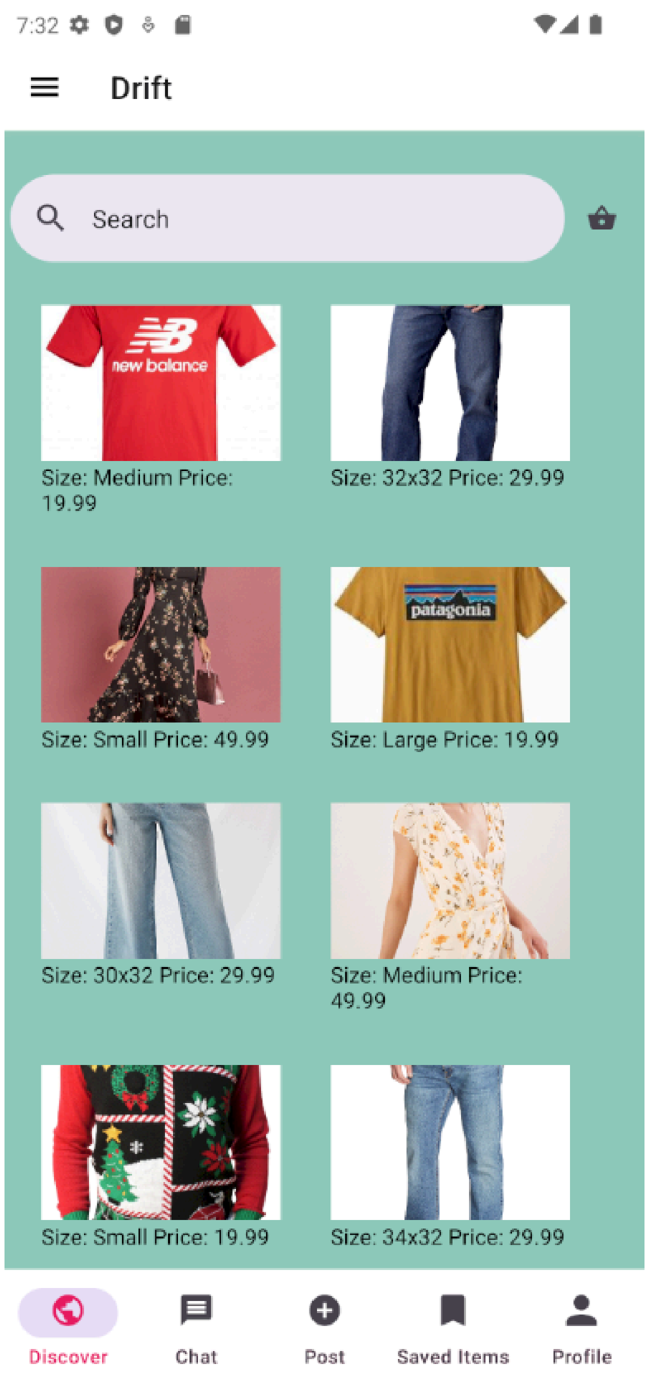
This screenshot depicts our applications splash screen where users will first land upon opening Drift for the first time. The “Let’s Get Thrifting!” button navigates to the Sign Up page for users to create an account. Also, this screen will be where users are navigated to upon sign out.



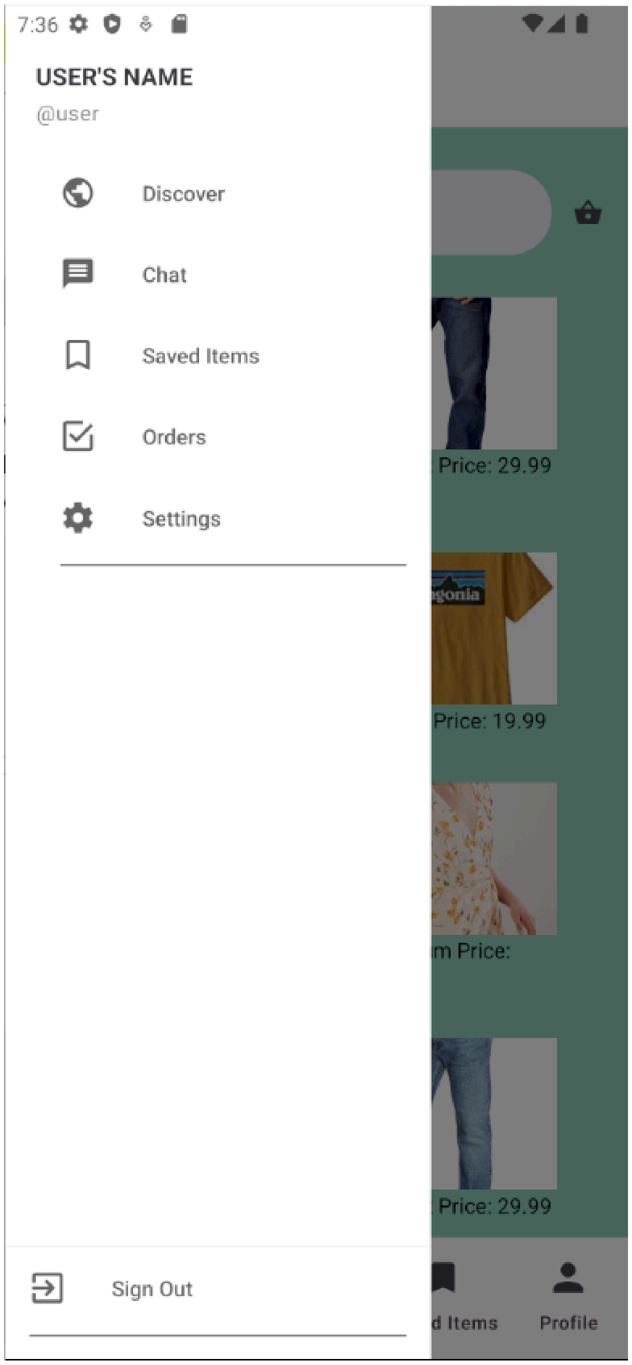
This screen shows the Sign Up page with seven text input fields which take input from the user to register a new profile with Drift. The "Sign Up" button navigates to the main Discover page and executes an API post call to insert the user into our database. The already registered section links to our Login page where users who have an account can re sign in if they have been signed out.



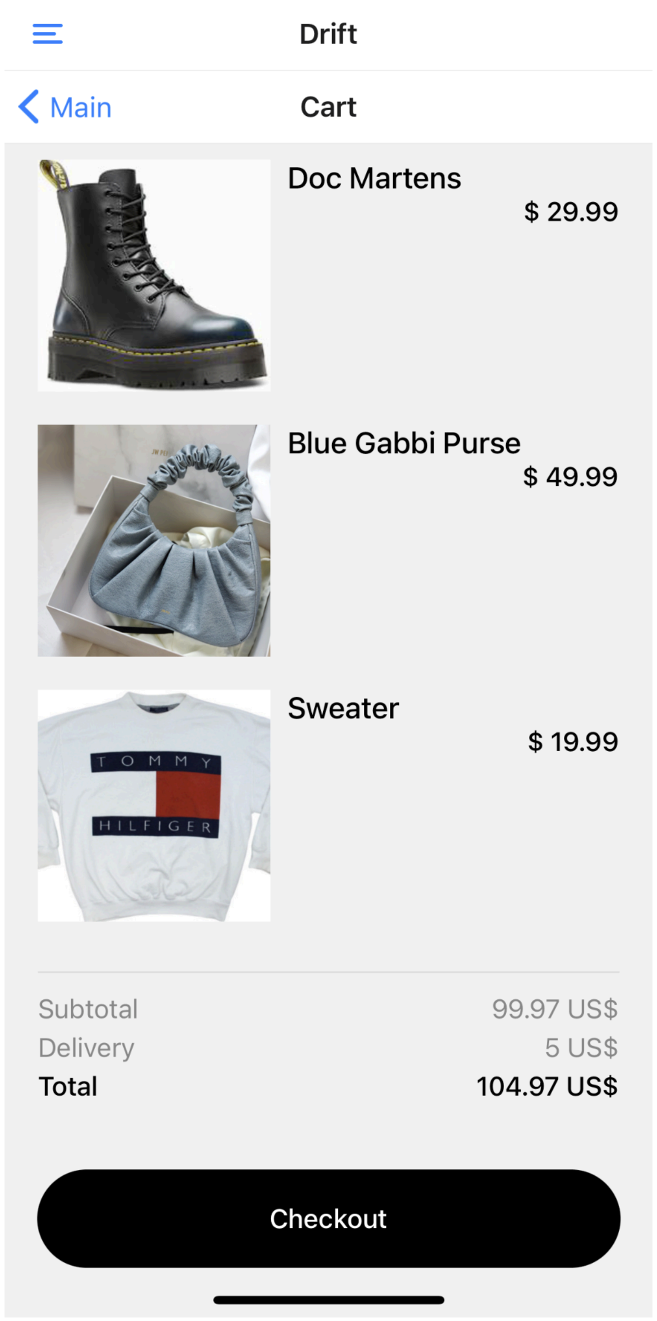
This screen illustrates Drift's Login page which allows previously registered users to sign in to our application using the username and password that they signed up with. The sign in button navigates to the Discover page if the credentials input are correct and match those in our database. If none match and/or the username or password is incorrect an alert message pops up for the user to acknowledge and try again. The forgot password will link to a page prompting the user to enter the email address for which account to reset.



This screen is our Discover page which is the main page for searching for items up for sale by other users, clicking on items for more details, and shopping for all the Drift has to offer. The bottom of the screen has the main navigation to our other four pages. A user can go to our Drawer for a settings and Orders page or click on the cart button to view whatever they may have in their cart.



This screen presents our Drawer navigator that has some of our main page links as well as buttons to navigate to an Orders page, Settings page, or a Sign out option. This is another useful component for users to interact with that helps with navigating the Drift application. The user's name will be depicted at the top to indicate whose account is signed into and being used.



Drift

Main

Cart

Doc Martens

$ 29.99

Blue Gabbi Purse

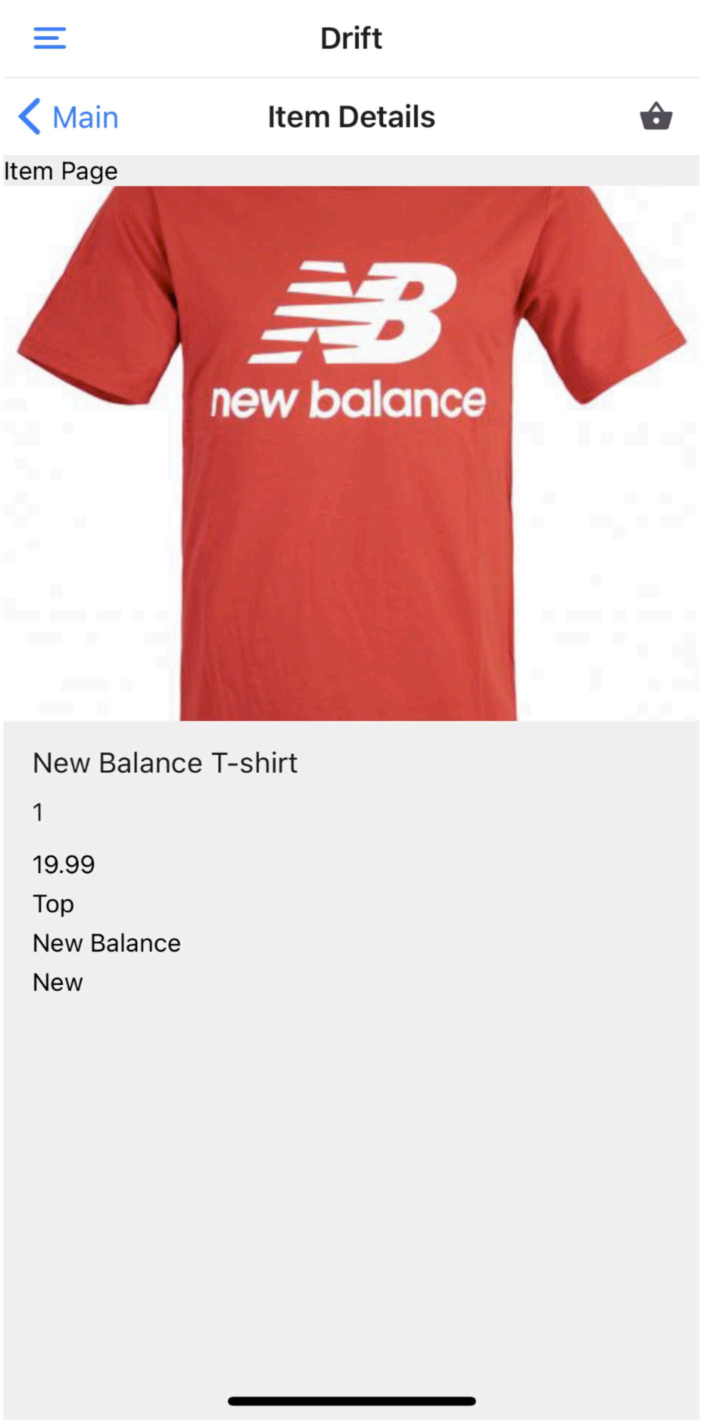
$ 49.99

Sweater

$ 19.99

Subtotal	99.97 US$
Delivery	5 US$
Total	**104.97 US$**

This screen is our Cart page which shows a list of all the items the user added to the cart. Each section has the item's name, photo, and price. At the bottom, the user can see the calculated price along with a button to check out all the items.



Item Page

New Balance T-shirt

1

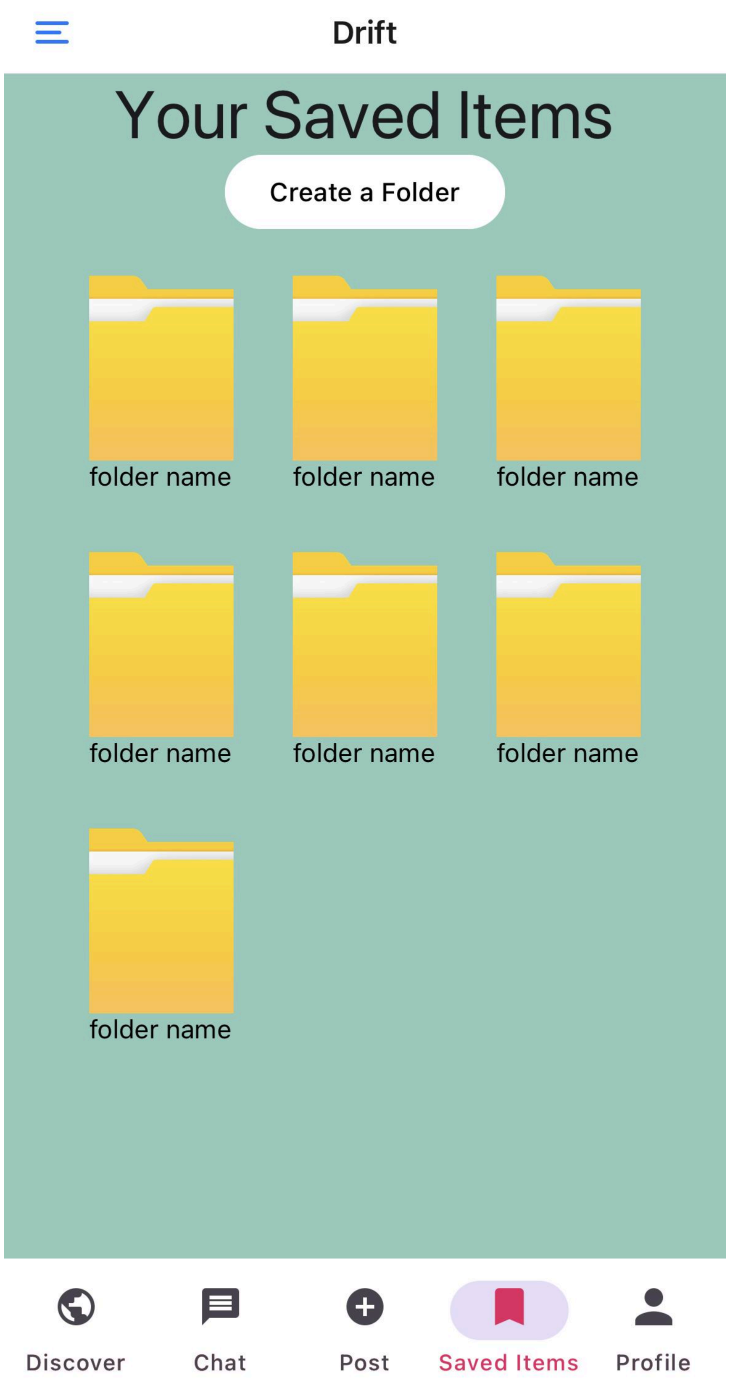
19.99

Top

New Balance

New

This screen is our Item Details page which can be navigated to after clicking on a photo of an item such as in the Discover page. The item details features more information about the item such as it's price, category, condition, and brand.



This screen is our Saved Items page. In this screen, the user can see all the folders they created to organize their saved items. The user can also create new folders through the "Create Folder" button. Clicking on a folder will navigate the user to a screen that has all the items under the corresponding folder.

5 - Updated Glossary of Terms

Thrifting -Shopping for secondhand clothing.

Secondhand clothing - Clothing once owned by another person that is resold.

Listing - Item/product available for sale on our platform along with its description and product image.

Discover Page - A feed of recommended products posted by other users. It may be filtered by keywords.

Shopping Cart - A collection of items selected by a user for purchase. The list of items is displayed on the shopping cart page.

Seller - The user profile that has posted a specific item.

Buyer - The user profile interested in purchasing an item.

Product Images - An image posted by a seller to represent a listed item.

Product Description - A text description written by a seller to include information about the size, condition, brand, price, and other details of a listing.

Chat - Feature to facilitate direct and private communication between two users.

Inventory - The product listings uploaded by a seller.

Search - Feature implemented within the discover page to allow users to browse products and users by keywords, entered into a search bar.

Order - Product purchased by a buyer to be fulfilled by the seller.

User Profile - Personal accounts containing information about individual users, including their listings, bio, and rating.

Condition - State or quality of an item, indicating whether it is new, lightly worn, or used.

Brands - Manufacturers associated with products.

Categories - Groupings of items based on type of clothing (shirts, pants, choes, etc).

Ratings - Feedback provided by users to evaluate the quality and reliability of sellers and their products.

6 - Engineering Standards and/or Technologies

1) React Native - React Native is the mobile version of the React language. We are using this for our frontend along with various libraries such as React Native Paper.
2) SQL - SQL (Structured Query Language) is a computer language used for programming databases, in particular relational database management systems. We are using SQL queries to access things such as our user data which is stored on a MySQL server.
3) Expo Go - Expo Go is a sandbox environment to test React Native applications on android and iOS. Expo itself is a set of tools for working with React Native and its environment.
4) Typescript - Typescript is a programming language that itself is a superset of javascript such that it compiles to Javascript. It is called that as the language itself is strongly typed in comparison to javascript and offers greater tooling for building large applications.
5) ACM Standard 1.6 (Respect Privacy) - We respect the privacy of our end-users by encrypting their passwords which is stored securely on our MySQL server.
6) ACM Standard 1.4 (Be fair and take action not to discriminate) - We protect against discrimination of our users by making sure that the order of the advertised items is in chronological order and no extra preferentially is given. This way user postings are fair and there is no additional sorting which could discriminate against certain users over others.

7 - Project Impact and Context Considerations

Our project explores the technological solutions that exist at the intersection of environmentalism, human rights, and affordability. Bringing options for secondhand shopping to a wider audience, Drift contributes to a reduction of the pollution and labor issues involved with fast fashion production and disposal, while providing more economic shopping choices for buyers. To ensure that the positive impacts of our project reach their full potential, it is necessary to make accommodations for differently abled users. For keyboard access we will manually attempt navigation of pages using an external keyboard, and for screen reader compatibility we will manually check individual input elements and buttons for appropriate audio descriptions. For color contrast we plan to use contrast checker websites available online. Finally we will explore the React Native

Accessibility Engine for accessibility recommendations overlooked by manual testing [Lacerda, 2022].

Keyboard Accessibility -

Sign Up & Login Page - When tested with an external keyboard using the 'Full Keyboard Access' feature on iOS, the fields for sign up and login information were navigable using 'Tab' and '↑Tab'. 'Login' and 'Sign Up' buttons could be focused and activated using the enter key. Following standard practices for implementing React Native button and input elements allowed for full keyboard accessibility on these pages.

Discover Page, Item Details, and Cart Page - When tested with an external keyboard using the 'Full Keyboard Access' feature on iOS, the 'Search' field within the Discover Page was accessible, scrolling to view items further down the FlatList was possible using the arrow keys, and the Cart Button was selectable using Tab. The components of the Cart Page, including the checkout button and the payment sheet input fields, were similarly accessible.

Post Item Page - When tested with an external keyboard using the 'Full Keyboard Access' feature on iOS, the Image Pickers were accessible with 'Tab', and the native iOS camera roll was similarly navigable. The input fields for description, title, price, and category could also be accessed using keyboard controls. Finally the Post Item button could be selected and pressed using the keyboard.

Screen Reader Compatibility:

Sign Up & Login Page - When tested with the 'VoiceOver' accessibility feature on iOS, we were able to confirm that the input fields and buttons on the Sign Up and Login page were read according to their corresponding labels. When tested with the React Native Accessibility Engine it was recommended to add the 'accessibilityLabel' prop to input fields, though we will apply this prop only to fields that could benefit from more detailed descriptions than their current labels, namely the 'confirm password' field.

Discover Page, Item Details, and Cart Page - When tested with the 'VoiceOver' accessibility feature on iOS, we were able to confirm all text fields belonging to item descriptions, buttons, and the Search input were read according to their appropriate labels. The React Native Accessibility Engine recommended we add an 'accessibilityLabel' prop to the Search input field. The payment sheet initialized with the checkout button in the Cart Page was similarly compatible with the screen reader.

Post Item Page - When tested with the ‘VoiceOver’ accessibility feature on iOS, we were able to confirm each input field was appropriately read by the screen reader, as well as the categories within the category modal. The React Native Accessibility Engine recommended for the Image Pickers to be given an accessibilityLabel prop.

Color Contrast:

Sign Up & Login Page - Using the contrast checker from Web Aim [WebAIM, 2024], we discovered that the background color of #8FCBBC for our Login and Sign up buttons failed to meet the WCAG standards for accessibility [WCAG, 2023].

Discover Page, Item Details, and Cart Page - Using the contrast checker from Web Aim, we discovered that the background color of #8FCBBC on our Discover Page had suitable contrast with the description text accompanying product cards. The contrast for the Item Details page and the Cart Page also met the WCAG standards.

Post Item Page - Similar to the issue with the Login and Sign Up buttons, the Post Item button, as well as the placeholder text for the input fields fails to meet the WCAG contrast standards for accessibility.

Text Size and Scaling:

Sign Up & Login Page - The smallest font size throughout our Login and Sign Up pages is 20px, which meets the minimum font size according to the U.S Web Design System standards [USWDS, 2023]. Furthermore, our app supports dynamic text scaling through the Display & Text size accessibility feature on iOS, so for users with visual impairments the font can be made larger.

Discover Page, Item Details, and Cart Page - The font sizes on the Discover Page and Cart Page meet the standards for minimum font sizes, though the Item Details page does not. We will add styling to increase the font size to at least 16px for all text components on this page. Furthermore, the React Native Accessibility Engine considered the cart button within the Discover Page to have too small of a hit area.

Post Item Page - The font sizes on the Post Item page meet the standards for minimum font sizes, and are responsive to dynamic text scaling.

Alt Text:

Discover Page, Item Details, and Cart Page - The images on the Discover Page, Item Details, and Cart Page are currently not associated with alt text.

Improving Accessibility -

Our main issues with accessible design include a lack of accessibilityLabel props on input components, color contrast issues, font and button size issues, and a lack of alt text for images. We will add accessibilityLabel props to certain fields that could benefit from more detailed descriptions, change the background color of the Login, Sign Up, and Post Item buttons, increase the font size on our Item Details Page, and increase the button size of the cart button on the Discover Page.

8 - Updated List of References

“Problem-Domain” Book:

- David Choi. Full-Stack React, TypeScript, and Node. Packt Publishing, 2020.
  - This book covers in-depth the tactics and inner workings of full-stack web development with a React, TypeScript, and Node technology stack. It goes through everything from basics of each technology as well as in depth descriptions of integration of these tools along with other library components. This book will help very much in understanding connecting our React frontend to our SQL backend through use of a TypeScript middleware.

Project Reference Articles::

- Mobile Development Articles -

  IBM Developer, https://developer.ibm.com/technologies/mobile/articles/ . Accessed 29 Oct. 2023.

  - This link provides a culmination of articles regarding mobile app development which will prove beneficial when diving into the responsiveness functionality. We plan to have our application work on a multitude of devices. Therefore, these articles from IBM will help greatly. It

also has articles that discuss mobile security and privacy and how to implement these within applications which are great topics to highlight for our work in the future.

- Dirin, Amir, et al. "Feelings of Being for Mobile User Experience Design." International Journal of Human-Computer Interaction, vol. ahead-of-print, no. ahead-of-print, 2022, https://doi.org/10.1080/10447318.2022.2108964.
    - Dirin's article referenced above covers the multitude of topics surrounding User experience design which will help for our development to be a competitive application in the mobile thrifting market. This will be immensely relevant for our mobile application development and frontend UI design. The article dives into human computer interaction and how that couples with UX implementation to make more appealing applications.

Project Related Websites):

- React Native Tips & Articles -

  "Articles with Tips on React Native Technology." *RubyGarage*,

  https://rubygarage.org/blog/tag/react-native. Accessed 29 Oct. 2023.

    - This website will be great documentation to look through with working in the React native environment. We are all versed in the use of the React framework, but this website discusses tips and information for utilizing the framework in one's own applications. This site includes links to articles ranging from libraries to use to the costs to create applications running with React native technologies.

- Building E-Commerce Mobile Application -

  "Build an eCommerce Mobile App in 2023." *Northell*, 14 Aug. 2023,

  https://northell.design/blog/how-to-build-an-ecommerce-mobile-app .

    - This will help with the e-commerce aspects that we expect our application will encompass. Further, this website goes into great detail about the basics of e-commerce, design challenges in such a realm, as well as labor and time costs for implementing these types of applications. It gives great insights into the steps and tasks needed for creating an application that works for the buying and selling of various products.

9 - Contributions of Team Members

Team Members	**Time Worked on Project Part**	**Specific Activities Worked On**
Jordan Rood	7 hours	- Abstract - Updated Specification (3.1, 3.2, 3.3) - Updated List of References - Helped with section 4.4
Fiorina Chau	6 hours	- Updated Design (4.1,4.2,4.3,4.4) - Engineering Standards And/Or Technologies
John Christian Jackson	4 hours	- Updated Glossary of Terms - Project Impact and Context Considerations

Works Cited

"react-native-accessibility-engine/README.md at main · aryella-lacerda/react-native-accessibility-engine," GitHub. https://github.com/aryella-lacerda/react-native-accessibility-engine/blob/main/README.md (accessed Feb. 20, 2024).

"How to Meet WCAG (Quickref Reference)," www.w3.org.https://www.w3.org/WAI/WCAG22/quickref/?versions=2.1#contrast-minimum

"Typography" *USWDS*, June 2023, https://designsystem.digital.gov/components/typography/.

"Contrast Checker" *WebAIM*, 2024,

https://webaim.org/resources/contrastchecker/.