

SuitUp: An Android application for donating used business attire to those in need

1. UML Diagrams

1) Use case descriptions

Use case	Make Donation
Iteration	1, last modification 10 November
Primary actor	Donor
Goal in context	Enter items they want to donate into the system database
Preconditions	Donor has logged into system, selects the "Donate" button on the dashboard or from the menu shortcut
Trigger	Donor wants to donate clothing item(s)
Scenario	<ol style="list-style-type: none">1) The donor logs into the application2) Donor is directed to the dashboard3) Donor clicks the "donate" button4) Donor is directed to donation page5) Donor fills out the form with dropdown options for each item they are donating6) System requests use of the donor's phone camera7) Donor takes a photo of the garment8) Submit button is enabled after all page items are filled9) Donor clicks the submit button
Exceptions	<ol style="list-style-type: none">1) Donor does not grant system the permission to use the camera2) Donor fills out inaccurate information about the item and submits the donation
Priority	High priority, donations are necessary for the applications functionality
When available	Functional by second iteration

Frequency of use	Frequent
Channel to actor	Via application's donate page
Secondary actor	System Database
Channels to secondary actors	Function to populate donation database
Open issues	<ol style="list-style-type: none"> 1) How do you prevent a catfishing attempt where inaccurate information is intentionally filled out? 2) Can a donor pull out of a donation?

Use Case	Update Donation Status
Iteration	1, last modification 10 November
Primary actor	Donor
Goal in context	The donor has to follow the step-by-step donation process in order to ensure the donation has been shipped to the recipient
Preconditions	Donor is logged into system, they are on the dashboard and click their donation notifications or access the status page from the menu shortcut, they have donated at least one item that has not been claimed
Trigger	Donor's item has been requested by a recipient
Scenario	<ol style="list-style-type: none"> 1) The user logs into the application 2) Directed to the dashboard 3) Donor clicks the notification item in their donation notifications 4) Directed to the donation status page 5) Current step needed is highlighted 6) Donor checks off the step it is completed 7) The next step is activated, and the previous step is turned into a lighter color <ol style="list-style-type: none"> a) Request recipient's address b) User goes to post office to ship package c) They enter the tracking number for the package d) Wait until the user confirms their package has been received after a week 8) If the user has not entered their tracking number after 3 days, then they have the

	option of canceling the donation 9) Text message displayed at bottom if all requirements are fulfilled
Exceptions	1) The user enters the wrong tracking number 2) They are catfished - get the wrong address or a fake address 3) The donor cancels a donation after it has been shipped out 4) The recipient never notes that they received the order
Priority	High Priority, since the donor has to update the status of the donation and whether or not they have actually donated the item
When available	Second iteration alongside the Make Donation function
Frequency of use	Frequent, if the user is a regular donor
Channel to actor	Via the donation notifications page
Secondary actor	Donation database, Recipient
Channels to secondary actors	Donation database is updated through a function Recipient is prompted by a pop up view to enter their address Recipient is prompted by a pop up after donor has entered tracking number to see if they received the order
Open issues	1) How do you encourage honesty about addresses and receiving packages? 2) How do you know if the donor has entered a valid/accurate tracking number 3) How to handle cancelling an order if the order is already shipped?

Use case	Search donations
Iteration	1, last modification 10 November
Primary actor	Someone in need (hopefully)
Goal in context	To select one or more clothing items and add items to donation cart
Preconditions	The user has logged into the system, selected the "search" button or shortcut in their dashboard

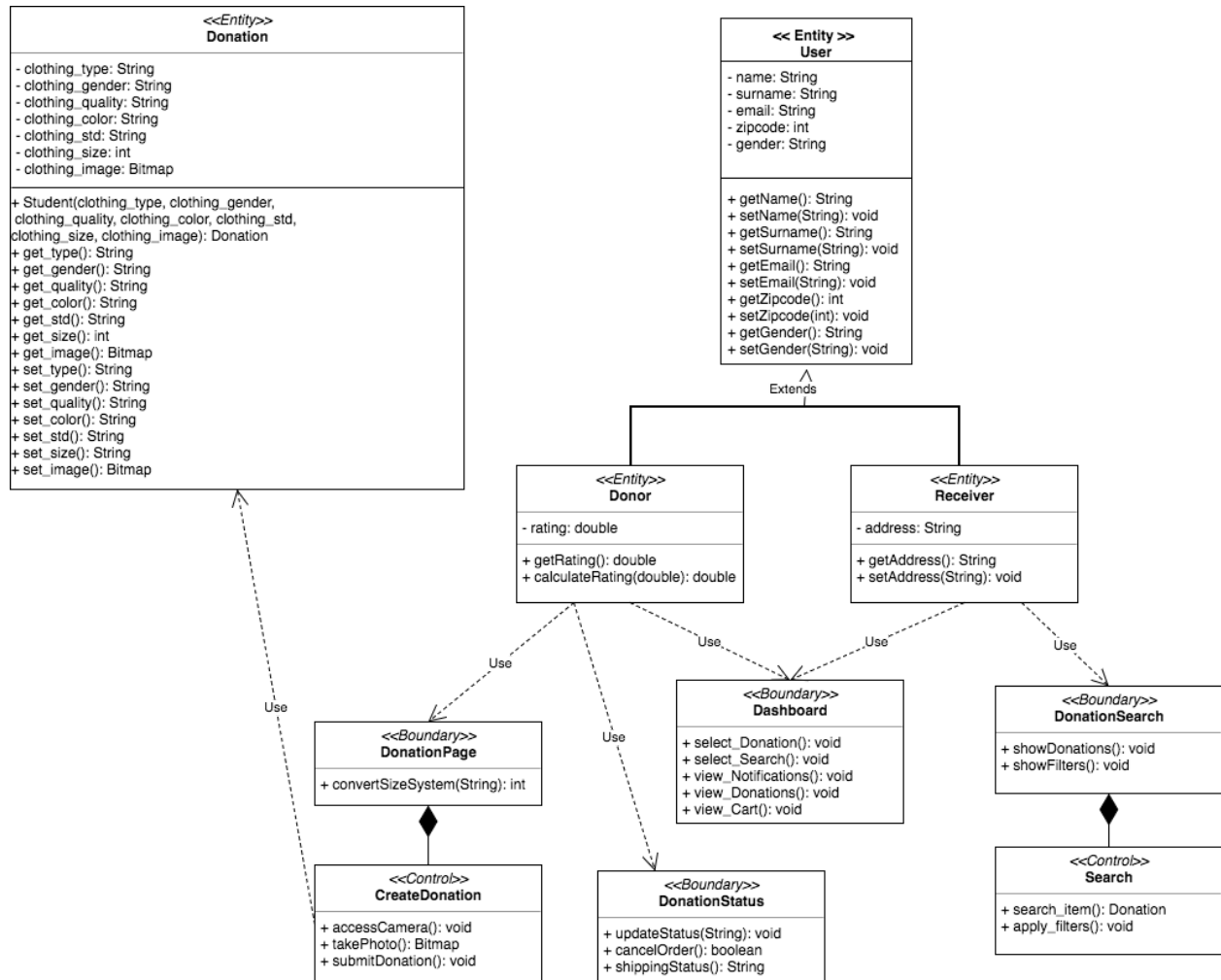
	menu, and selected items
Trigger	A person in need of a business outfit has selected the donations they wanted and needs to have their order shipped
Scenario	<ol style="list-style-type: none"> 1) User logs into application 2) User is directed to the main dashboard 3) User selects "search button" on page or accesses search page from menu shortcut 4) User is directed to the search page 5) User utilizes the search bar and filter menu to find the clothing they want 6) User clicks "Search" button and results are displayed 7) User can scroll through results list view and press the "Add to donations" button on each list item they would like 8) The system adds the selected items to the donations cart 9) User can click the "cart" button to see their selected donations
Exceptions	<ol style="list-style-type: none"> 1) User id and passwords are not correct 2) User does not select any donations 3) User's query in the search bar does not match any available items
Priority	High priority, necessary for the applications functionality since users need to search for donations
When available	Functional by second iteration, with additional filters implemented by final iteration
Frequency of use	Frequent
Channel to actor	Via application's search page
Secondary actor	System Database
Channels to secondary actors	Database queries in search function
Open issues	<ol style="list-style-type: none"> 1) Is a search bar necessary? Can the user just have a dropdown where they can check off options they want? 2) Will there be a message where the listview is if there are no items in the database? 3) User cap of how many items they can select? Should this function be

	implemented in the actual cart?
--	---------------------------------

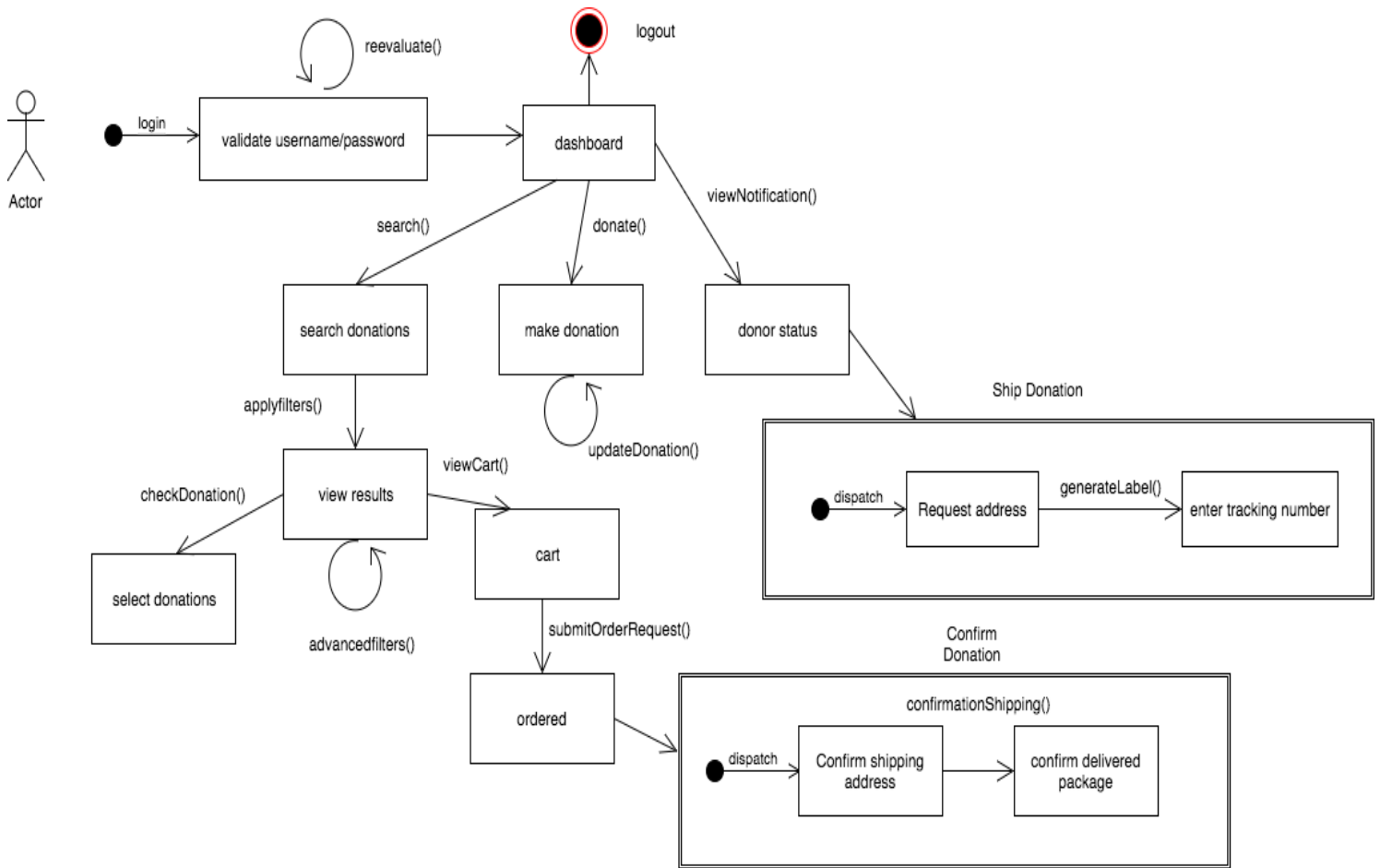
Use case	Manage donation cart
Iteration	1, last modification 10 November
Primary actor	Donation recipient
Goal in context	The user wants to manage the donations they want to receive and submit an order
Preconditions	Recipient is logged into application, they click the item in the cart on their notification page or access the cart page from the menu shortcut
Trigger	Recipient wants to definitely confirm the donations they want to receive or delete items they no longer want
Scenario	<ol style="list-style-type: none"> 1) The user is logged in 2) Directed to dashboard 3) Either clicks the cart shortcut on the dashboard or from the menu shortcut 4) Directed to cart page 5) User checks items they want using the checkboxes in the corresponding item listings 6) User can either click the "Select all" option to select all items or the "Empty cart" option to delete all items 7) If "Empty cart" button is selected, then a popup is generated to ask whether user is ok with this option, otherwise they can cancel the delete 8) The user can check items they want to delete, and click the delete button 9) If the "Delete" button is selected, a message will pop up asking if the user wants to go ahead with the decision or they want to keep the items 10) Once the user selects items, the "Request Donation" button is activated 11) User clicks "Request Donation" button 12) Receive a pop up message stating that their order is being processed and passed to the donor
Exceptions	<ol style="list-style-type: none"> 1) Case where user accidentally clicks the

	<p>“Request Donation” button for items they do not want</p> <p>2) User accidentally selects “yes” in the popup for deleting all items, and deletes all items</p>
Priority	High Priority, user needs to request the donation in order to receive items
When available	Second iteration
Frequency of use	Frequent for regular recipients of donations
Channel to actor	Via the donation cart page
Secondary actor	Donation database, Order database
Channels to secondary actors	<p>The donation database is updated through a function to note that it is no longer available</p> <p>The order database is updated with the item to be ordered</p>
Open issues	<p>1) What would be the cap of items the user can request at once? Should it be below 5, since the donor will pay shipping for donation?</p>

2) Class Diagram

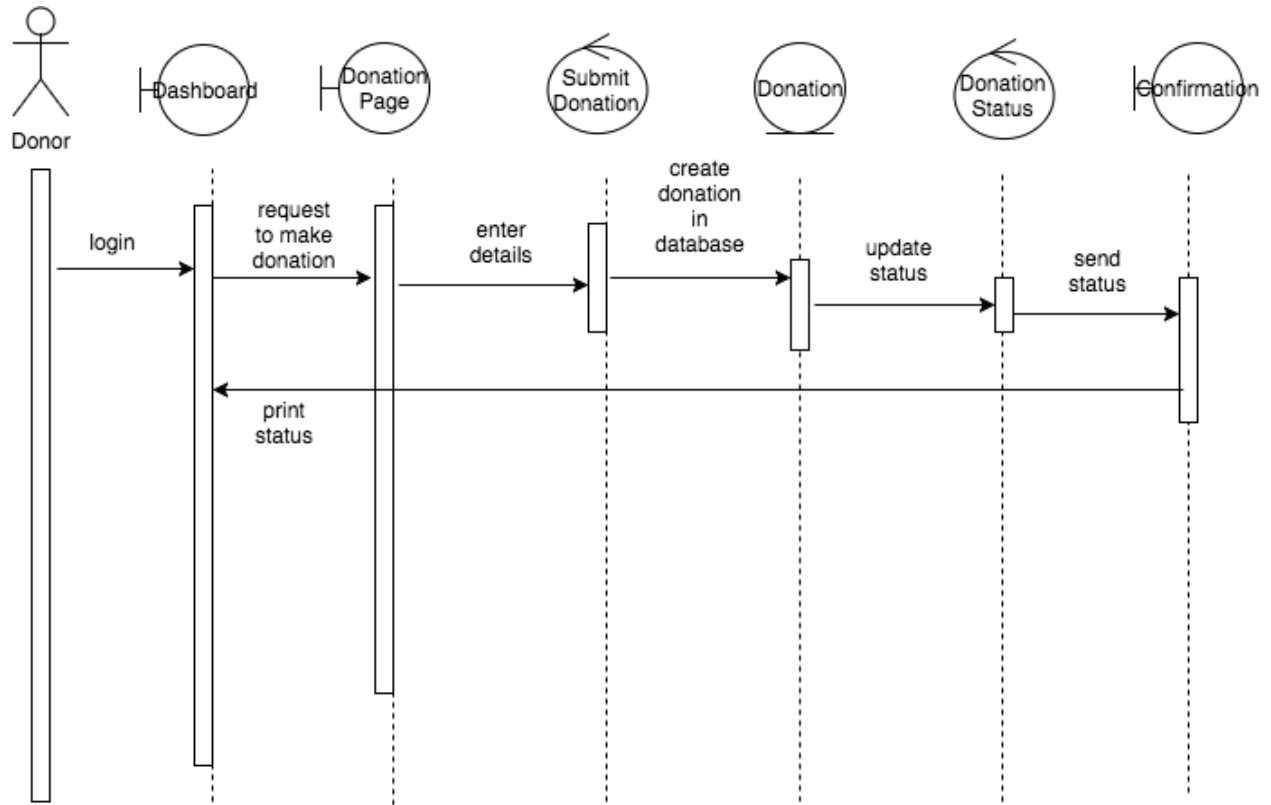


3) State Chart

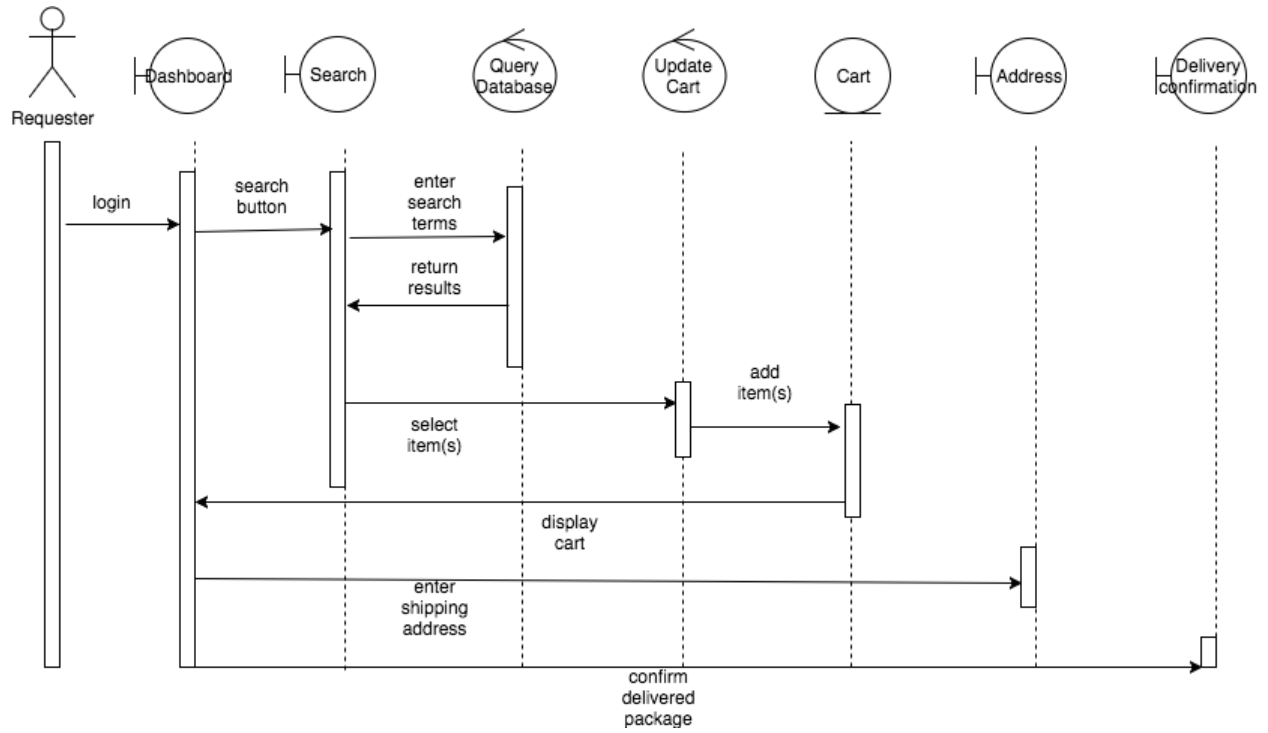


4) System Sequence Diagrams

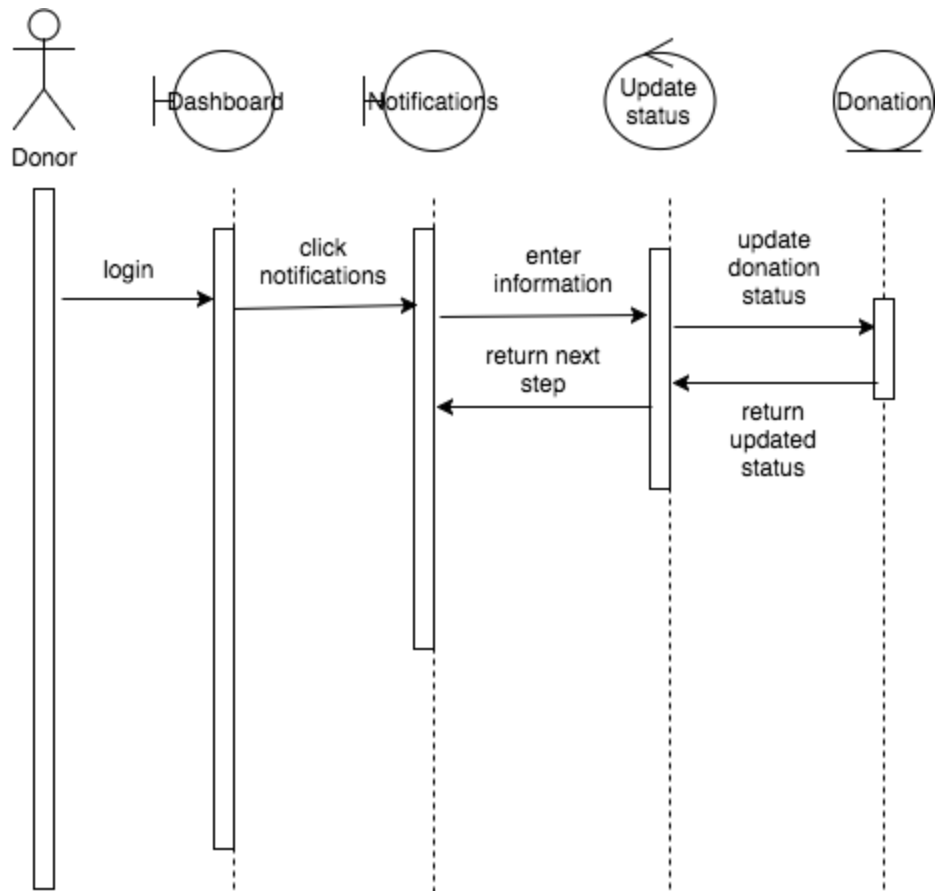
Donation Activity



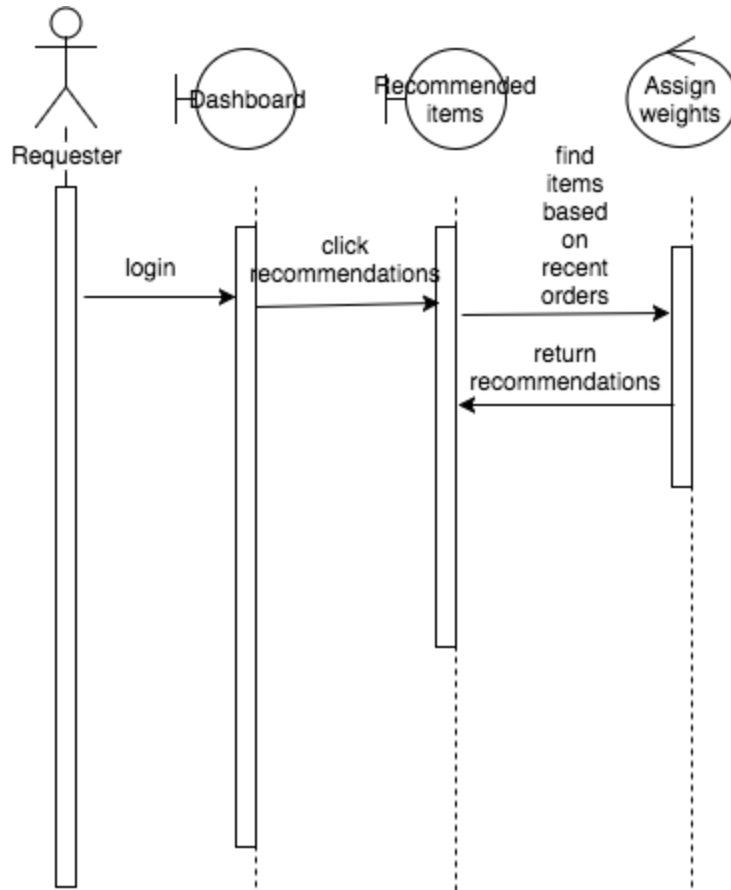
Donation Request Activity



Ship Donation



Recommendation Activity



2. UI Designs (Layouts below eight rules)

- 1) Consistency: The orientation of the logo/link to home, menu, and logout is consistent throughout the pages which they are applicable. Toast messages are only used for errors, and popup views are only used for major notifications related to the donation delivery
- 2) Shortcuts: There are shortcuts in the app bar for returning home (clicking the logo to the left), accessing pages in a menu tab, and logging out of the app.
- 3) Informative feedback: In the login screen, if the user enters the wrong username or password, a toast message pops up and states "Invalid username or password." This notifies the user of their mistake so they can correct it and properly login.
- 4) Dialogs to yield closure: There are dialog boxes and notifications that ensure the user they completed a task or need to complete an urgent action. For instance, there is a popup view that is displayed when the user confirms a donation request, stating that the donor has been contacted and that the user should check their notifications for the next step.

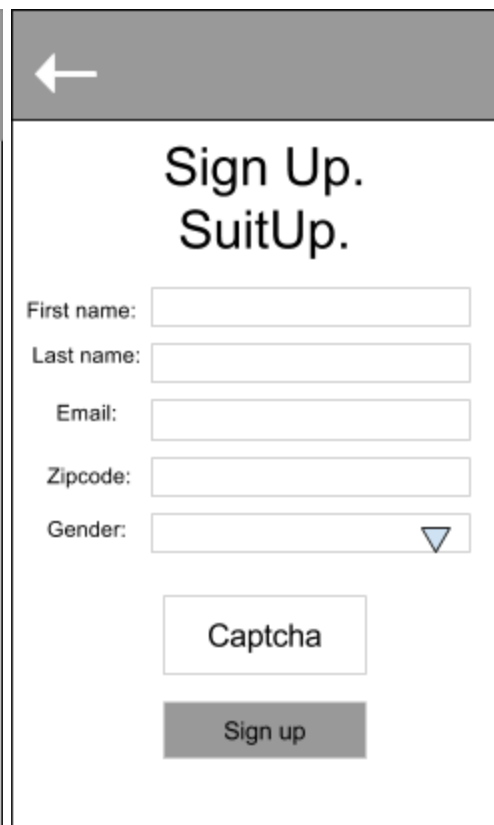
- 5) Error handling: If the user enters a wrong sized zip code or invalid format of input, the specific label for that edit text box will be highlighted or have a textview beneath it saying the input was invalid. The correct data in that form will be left untouched so the user won't have to enter information again. For the most part, there are dropdowns so users are less prone to mistakes.
- 6) Reversible actions: There are back buttons and a menu button that allows the user to go back to another page if they are at the wrong page.
- 7) Short term memory load: The user only has to remember their username and password for logging in. Additionally, they have to remember the tracking number for the package they sent, if they donated clothing. The app has dropdown menus with responses so users do not have to overthink responses or waste their time filling out forms.
- 8) Users are in control: Users have control over the system, as they are free to select which filters they want to search the database with. Additionally, their actions are necessary for the app to function, as the user needs to donate or receive a donation. The user, as a result, are the ones in control of the app.

Log-in Screen



The Log-in Screen features a dark grey header bar. Below it, the 'SuitUp' logo is centered, followed by the tagline 'Give a suit, make a difference.' in a smaller font. There are two input fields: 'Username' and 'Password', both with light grey borders. Below these fields are two buttons: 'Log in' and 'Sign up', both with dark grey backgrounds and white text.

Sign up Screen



The Sign up Screen features a dark grey header bar with a white back arrow on the left. Below the header, the text 'Sign Up. SuitUp.' is displayed. There are five input fields: 'First name:', 'Last name:', 'Email:', 'Zipcode:', and 'Gender:'. The 'Gender:' field is a dropdown menu with a downward arrow. Below these fields is a 'Captcha' box and a 'Sign up' button with a dark grey background and white text.

Toast message if login is invalid

If zipcode is invalid, a message will pop up underneath stating it is invalid

Gender is a dropdown menu (male, female, non-binary, prefer not to say)

Back button to return to login if you accidentally selected sign up

Captcha or “Not a robot” checkbox to ensure user is “human”

User Dashboard

logo

menu


logout

Hello, ____ ! Ready to
SuitUp?


Donate a suit or search for a donation

Donate

Search


Donation Notifications 

Mail out donation

Donation Cart 

Donation 1

Donation 2

Recommendations 

Donated 1

image

Clothing type

Donator name

Quality

Donator Reliability Rating

Donate and search are buttons

Donation notifications will update the user about what they need to do next if they are donating (ship package, provide tracking number, etc.) or if they are receiving a donation (ie: confirm they received the package, rate the transaction)

Donation 1 and Donation 2 contain description/clothing quality, image, and donator name of clothing items received by user

Donated 1 will contain the image, description/clothing quality and rating of product

Click notification to get to page with specific steps

Donate clothes page

The screenshot shows a web form titled 'Donate' with the subtitle 'Help someone SuitUp for their next interview or business venture'. The form is enclosed in a light gray border. At the top, there is a dark gray header bar containing the text 'logo', 'menu', and 'logout'. The form fields are as follows: 'Clothing Type:' with a dropdown menu showing 'Khaki Pants'; 'Gender:' with a dropdown menu showing 'Male'; 'Quality:' with a dropdown menu showing 'Good as new'; 'Color:' with a dropdown menu showing 'Red'; 'Size System:' with a dropdown menu showing 'US'; and 'Size:' with a dropdown menu showing '13'. Below these fields are two buttons: 'Take a photo' and 'Submit Donation', both in a light gray box with dark text.

Drop down menus

Take a photo -> access camera to take photo

Required to take a photo of product

Toast message if something isn't filled out when submit is clicked

Color drop down, with a custom fill in if a color isn't available

*Size options based on gender, clothing type and size system drop downs
Submit Donation button disabled until all fields are filled*

Address Request Form

X

Your order is ready to be shipped.
Enter your address below:

Address

Zipcode

*Once the donor receives the order, they request the address from the user
The user enters the address in the form*

Donation Reliability Rating

X

You just received a donation of:

- Blue blazer, Size 20, Male

From Bob C.

Please rate your transaction with Bob

☐ ☐ ☐ ☐

Popup on home dashboard after a recent transaction

Search for clothes

home menu logout

Search

Advanced Search ▲

Color ▼ Gender ▼

Size ▼ Quality ▼

Search Cart

Search Results ▲

Item 1

Item 2

Item 3

Drop down selections

List view to hold search results

Collapsible Advanced Search and Search Results Menus

Item cart (User's Donation Requests)

[home](#)
[menu](#)
[logout](#)

Donation Cart

[Empty cart](#)
[Delete](#)
[Select all](#)

[Request Donation](#)

Item 1☐

Item 2☐

Item 3☐

image

Clothing type
 Donator name
 Quality
 Donator Reliability Rating

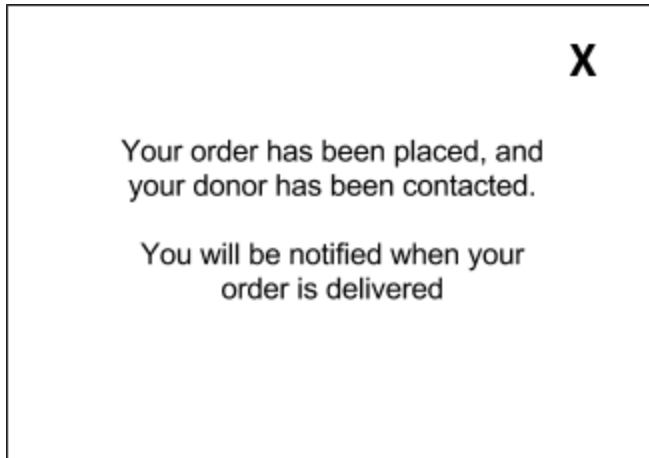
Checkboxes to select all items, or delete all items

Toast message to see if user wants to delete item(s)

Button to select all or delete all items

Request donation button activated once an item has been checked

Clothing confirmation



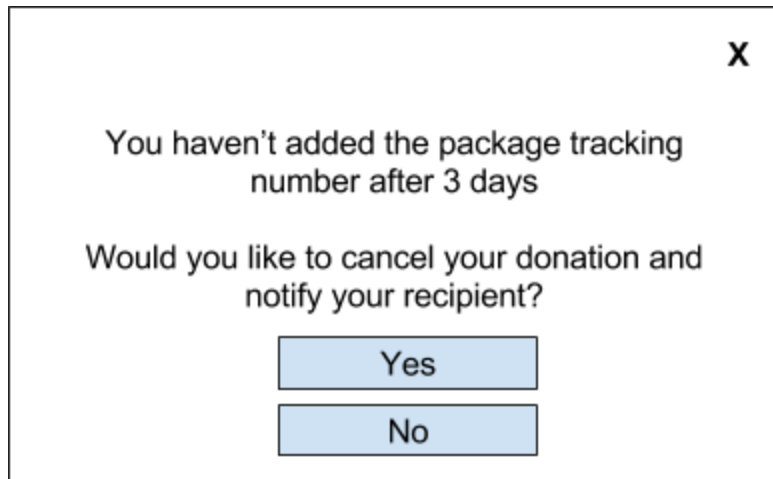
Popup window

Donator dashboard

[home](#) [menu](#) [logout](#)

Donation Status

- 1) Request address
- 2) Go to your local post office and ship the package
- 3) Enter tracking number here:
- 4) Donation received



Progress bar and steps highlighted to show process

Popup if the user doesn't ship out the package after 3 days

3. Modularity and Encapsulation:

My program encompasses modularity and encapsulation. The instance methods used in the classes of my classes are private, as I do not want these valuable pieces of information to be accessed freely/publicly. For instance, the attributes of a donation, such as the quality, clothing type, etc. will be private so it would be harder to modify these attributes -- which in turn, encapsulates the data. The receiver cannot modify the donation and the donor can only modify these attributes through modifier methods. I have created cohesive models, with minimal coupling. For instance, the methods in the Donation class involve updating the overall donation if a change is made, format the listview with all the attributes of the donation, and access each attribute of the donation. As for coupling, I intend to have the donation class to interact with the class responsible for the dashboard. This will help with displaying recent donations and other donation related operations. I believe that the coupling of the program will mainly involve the dashboard class, as it displays important information regarding donations made, shipping status, and donations received.

Elegance and efficiency of algorithms

I am using a recommendation algorithm, possibly in the user's dashboard, to help the user find what they are looking for based on their previous donation requests. I will be using a weighting scheme to help determine whether the user's previously received donations are similar. Once the user has received at least one donation, any item in the same clothing category are assigned one point. For each attribute, like color, size, and quality, that matches the item the customer received, the corresponding clothing item will receive a point for that category. The items with the most amount of points will be

displayed at the top of the listview. I intend on using SQL queries to help match similarities in the categories, and I will store the weight of the donation items in the database.

This weighting method is efficient, since it can be separately applied for each donation item in the user's recent donations. The weighting algorithm will only be called for the items that the user had recently received (ie: up to the 10 most recent items) to best match the user's current preferences.

Appropriateness of data structures:

I will be using databases to handle the donations, accounts and donation requests as SQLite databases are flexible and compatible with Android applications. I have found that it is common practice for programmers to use databases, as they offer better security options than arrays or hashtables. Databases will also allow for convenient searches, as you can filter based on the table and category names.

4. Testing

Unit Testing:

I would test each class, like the donation class and the account class by creating driver classes that call the methods of the respective class. For instance, a dummy test class for the account class would populate the account database with some users, call the names and information accessors, and modify user information using the modifier methods. I would also test cases where inaccurate data is passed into the account to see whether the parameter types have been set properly.

Integration Testing:

For integration testing, I will be using a bottom-up approach in which I will test the "lower most" classes first before testing the "top most" classes in the hierarchy. For example, I will test donation and shipping related classes before I test these classes with the dashboard class, as the dashboard class calls all important methods. For each module, I will create a driver class that will test both methods. An integration test with the donation and shipping classes, will not only test whether these classes work standalone but also whether the database marks the shipped item as unavailable once the item progresses in the shipping process.

System Testing:

For system testing, I will be stress testing the entire system by processing large transactions/amount of donations. I want to see if the system will lag due to the data and whether I would need to update the recommendation algorithm to handle a larger set of received items. My idea is to populate the database with 100 and 1000 donation items to see how system performance is affected. If the system is noticeably lagging, then I will have to alter some methods.

Usage of debugging or testing tools:

There are built in debugging tools within Android Studio, so I can “step-into” and “step-out” of my code if I want to isolate a bug. This is similar to the debugging tutorial in Assignment 2, where we used the built in IDE tools to detect and isolate bugs.

Test Cases:

Functionality	Inputs	Expected Output	Actual Output
Login validation	Username and password	Successful: redirects to dashboard page Unsuccessful: Toast message asking user to check what they have typed in	
Sign up	Name, email, zip code, username, password	Successful: Directed to dashboard Unsuccessful: prompted to check errors and submit again	
Search	Keyword, checked categories (if advanced)	Displays a listview of relevant results or outputs a message stating there are no results	
Order items in cart	Items in cart are checked and submit button is pressed	A confirmation message is shown	
Donate items	Clothing type, color, size, sizing system, gender, condition, image	Confirmation message and prompt for donating further Unsuccessful: Message to fill in missing categories before submitting	

		again	
Donation status	User address, Tracking number, Delivery confirmation	Successful: both user and donor have updated in a timely manner Unsuccessful: After a span of a week, if nothing is updated, the order is unresolved or cancelled	
Donor Reliability Rating	A rating on a scale from 0-5, with 5 being the best possible score for reliability	Successful: Button is clicked after a score is entered	
Recommendations	System uses previous user orders	Successful: at least 1 item is highly recommended based on the donation database size, category, and other qualities Unsuccessful: Unrelated clothing appears	