Design Document

Team 25: Shantanu Jha, Arka Acharya, Anqi Yu, Kevin Jiang, Minyi Lu, Richard Jung

Purpose

Today, people seek to know what clothing is in-style verses clothing that is on its way out. With our software, we hope to simplify this process by compiling user data into one place. Modeling after Instagram/Pinterest/Billboard/Spotify, our software will provide users with the information and customizability that will allow them to create their best digital wardrobes.

Our functional requirements include the following items:

1. Adding a new clothing item
    - Users and clothing companies can upload pictures which will be shown to the users who are following them. Picture can either be uploaded from device storage or taken from the camera itself.
    - The user will then add tags to the their picture. That picture will be uploaded to our server and added to the database under "user's uploaded pictures"
    - The tags help us distinguish different clothing items in the picture. For example, if a user uploads a new outfit, they can add tags to each clothing item which will take them to that item.
2. Adding an item to the user's closet/wardrobe
    - Users will be able to add clothing items to their wardrobe from clothing suggestions on their home feed from clothing companies.
    - Users will also be able to look for articles of clothing by using the filters in the search tab and add it to their wardrobe.
    - Users will also be able to follow tags used in user pictures and add the selected clothing item to their wardrobe.
    - All of the above will be done if the user actually has the particular clothing item in their physical wardrobe at home.
3. Users can be notified through push notifications
    - Users will be notified if another user likes or comments their post.
    - Users will be notified if another user follows them or requests to follow them if their account is private.
    - Users will be notified if a company they are following posts a new item.
    - Users will be notified if an item they like, but don't own, goes on sale or is not available any more.
4. Users can search through the database for specific item
    - The search option in no sense is a strict search option. The tags added by the users will be used to filter through the database and display the reduced results.
5. Users can "follow" and "get followed" by other users
    - Users can "follow" other users in order to track his/her activity

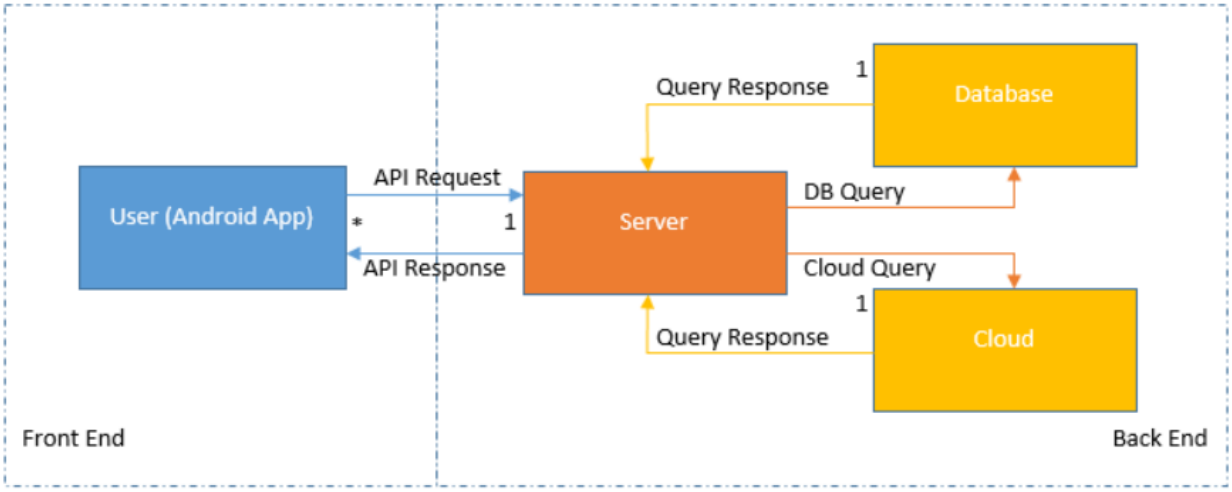
- Users can be "followed" by other users in order for his/her activity to be seen
- Users can check the number of users that are following him/her
- Users can check the number of users that he/she is following

6. Users can provide feedback on clothing
    - Users can either "like" clothing articles uploaded by companies giving them feedback for that particular item.

Design Outline

High Level Overview

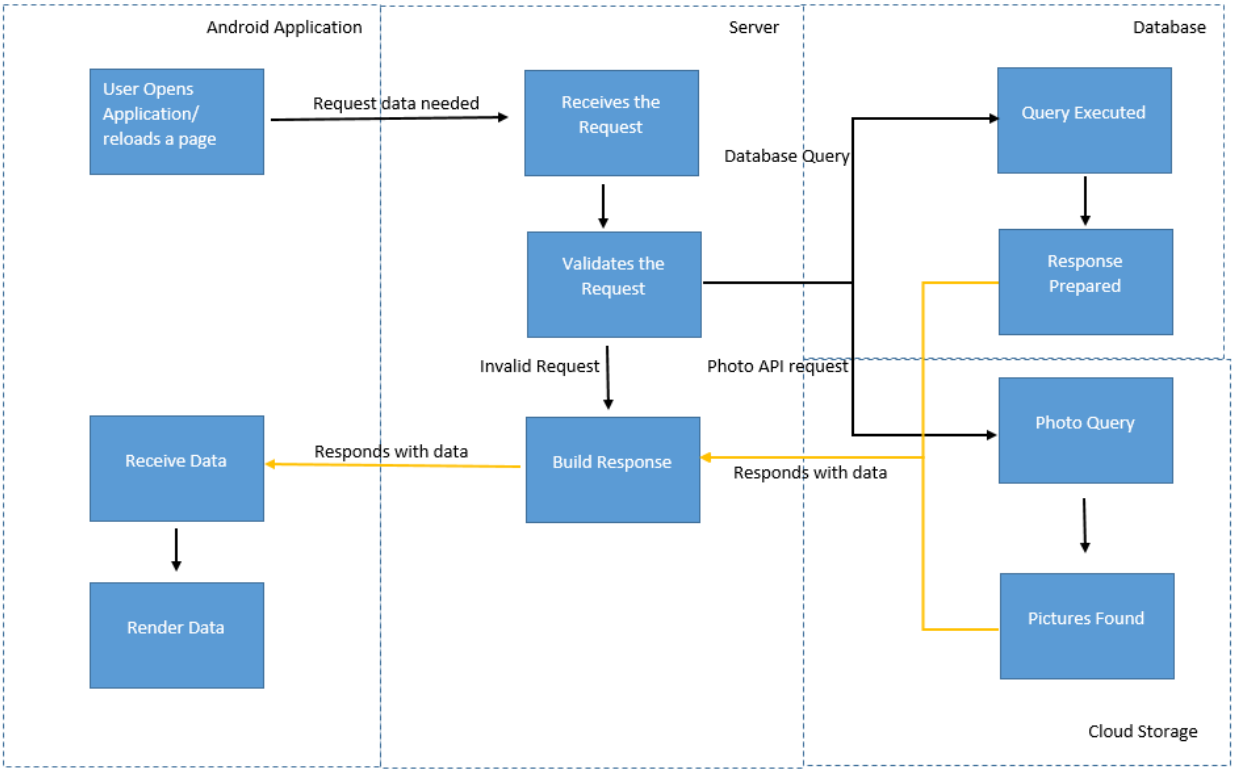
Our project will use the client-server model. The server will receive and respond to the client's request, and the client will parse and render the server's reply using Java. The figure below demonstrates a high-level overview of our system.



1. Client
    a. Client sends Java request to server API.
    b. Client receives Java response from server API.
    c. Response data is interpreted.
2. Server
    a. Receives and handles all client requests.
    b. Validates requests before processing.
    c. Queries the database and/or cloud and generates appropriate response.
    d. Sends the response to the client.
3. Database
    a. Stores all system information, such as users, companies, administrators, etc.
4. Cloud
    a. Stores all pictures/images and corresponding id tags.

Flow of Events

The typical flow of events begins after the user logins into the application. When the android application requires data from the server or cloud to complete a page or updates something in the database and/or Cloud Storage, there is an AJAX request that is sent to the server. From there, the Server validates the fact that the user may access or update the requested information. If the request is valid, the server will query the database and/or the cloud storage. There, depending on whether there was a get request or post request, the response prepared will be different. The Server will parse the data that is received into a JSON format and send that back to the android application. There, the App will parse the JSON token to render the pages on the application.



Design Issues

Functional Issue #1: What type of architecture should we use?

Option 1) Client-Server Architecture

Option 2) Unified Architecture

We chose the Client-Server architecture. This is because separating the application into a front and back end will make it easier to divide the work for our team, and we will be able to make changes to our back end without majorly affecting the front end functionality. If we were to use a unified architecture, the front end team would be dependent on the back end team for progressing their work. With the client-server architecture, the two teams can work simultaneously. This is because the front end can be linked with the back end once the back end components have been completed.

Functional Issue #2: What back end framework should we use?

Option 1) NodeJS

Option 2) PHP (Laravel)

Option 3) Python (Django)

We decided to work with Django. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Some reasons that we didn't work with NodeJS and PHP is the fact that Amazon Web Servers support Django much better than the latters.

Functional Issue #3: What database software should we use?

Option 1) NoSQL

Option 2) MySQL

Option 3) SQLite

After carefully comparing the three most popular relational database systems, we decided to go with MySQL. MySQL is a database software that is popular and widely supported by various server operating systems. SQLite is more suitable for embedded applications, and is typically not accessible over a network, which is critical to our web app.

Functional Issue #4: Do users need to login to use our service?

Option 1): Allow login using Facebook, Google+ account

Option 2): Create a username and password that is unique to our service

We chose to let users register through existing Facebook or Google+ accounts using OAuth for authentication as it will make registration easier for us and for the users. Most of the apps, irrespective of whether or not it is a social networking app, use Facebook or Google+.

Functional Issue #5: What front-end framework should we use?

Option 1) Java/XML

Option 2) C/C++

We chose to use Java/XML primarily for our previous experience in using this language. Furthermore, we feel as though Java/XML is the most supported, easy-to-use, and widely-used language for front-end framework. Therefore, we decided to use Java/XML for its simplicity and universality.

Functional Issue #6: How are we going to host our backend services?

Option 1) Amazon Elastic Compute Cloud

Option2) Virtual Private Server providers

We chose Amazon Web Services for hosting our project's backend components because the Elastic Beanstalk product provides an abstraction over a bare Linux server environment that allows us to easily deploy a wide variety of applications without needing to perform server maintenance tasks.

Functional Issue #7: How should we handle user permissions?

Option 1) All users are treated the same and clothing data would be uploaded by the developers.

Option 2) Company accounts have access to the database and add clothing themselves.

We decided that companies should be able to have full access control regarding the clothing database. This would be much better for the long-term maintenance of the system. While the first method would thins a lot easier for us. In long term the companies editing their data would cut down in costs for maintenance teams, and it made more sense to change the restrictions and abilities of accounts that had no purpose in adding people as friends.

Functional Issue #8: How should users be notified of updates regarding user activity?

Option 1) SMS notifications

Option 2) Email notifications

Option 3) Push notifications

We decided that having push notifications for users would be best for being notified when important activity is occurring on our software. We feel that most users of our software are people who readily have access to Wi-Fi or data services. Email notifications seem out of place in our current society and not all users have SMS complete paid for.

Functional Issue #9: What time of the design scheme should we use?

Option 1) Adaptive

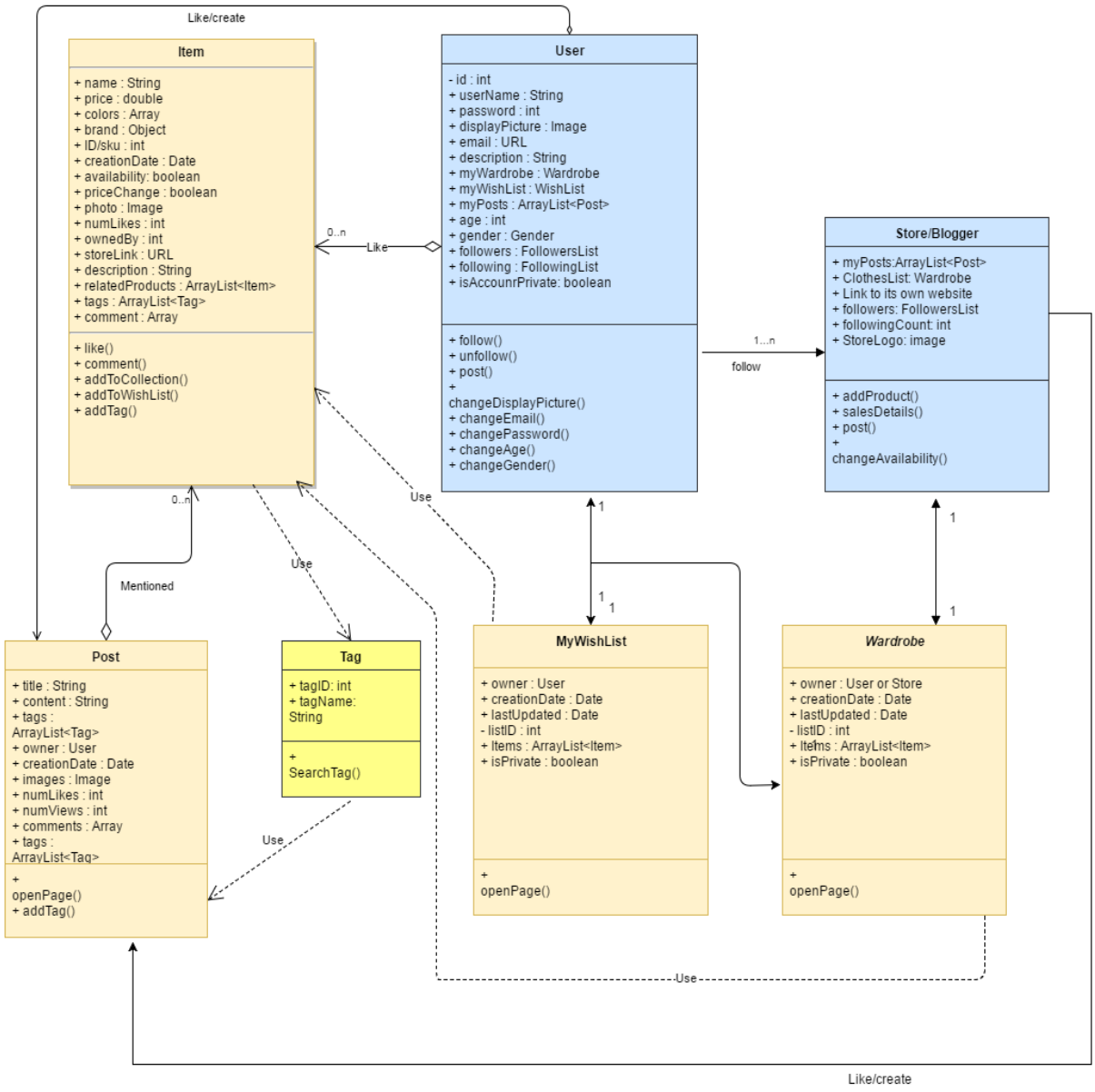
Option 2) Responsive

As different android devices have different screen sizes, we would like our app to be consistent across all the devices. Making the UI responsive achieves this goal while helping us to maintain one responsive front end instead of having two or more different front end versions.

Design Details

Database Design

We will use MySQL implemented relational database. Six tables will be created to store the information of clothes, items, users, Stores, Posts, Tags, and wardrobes. SQL foreign key constraint is used to associate information in different tables. A primary key specified by either an ID or a unique username will be used as an identifier.



Description of Database Tables

Shown above, we have provided a database schema that represents how we plan to utilize our functional requirements. Although this schema is not final, this schema should cover all of the main functions of our software.

SQL Tables

User (UserID, Facebook or Google account, username, gender, age)

Store (StoreID, account, username, Description, WardrobeID)

Clothes Item (ClothesID, Descriptions, Likes#, UserID, Tag Name, StoreID)

Wardrobe (WardrobeID, UserID, ClothesIDs, Timestamp)

Posts (PostID, UserID, Photo, ClothesIDs, Timestamp, TagName)

FollowerList (UserID, Followers#, FollowersUserIDs)

Tags (Tag Name)

Table names:

User

All user information will be stored in this table. This includes username (unique), full name, age and information about the user.

Clothes

- All data for clothes will be stored in a database.
- Each article of clothing will have an ID and tags.
- Clothes are the central focus of user activity. Users can post clothes, add clothes to his/her wardrobe, and like clothes in order to add them to his/her WishList.

Posts

- Posts table will store posts of the clients (Users and store).
- The table will store post ID, ID of client, number of likes, link to the image in the cloud.
- Our users can share their outfits or other fashion inspired photos through uploading photos as Post. Each Post will have a basic description, tags(if applicable), a photo, clothes items worn in the Post.
- Numbers of likes is shown for each Post
- All the clothes items' tags will appear in the POST detailed description.

Store/Blogger

- Store/Blogger table stores the stores using our application.
- The table will store information like store ID, store name.

Followers

- Followers table will store the followers of the user. The table will have a username field which will be used as the key to link to the User table using JOIN
- Using this table, the users can see their followers.

Following

- Following table will store the people/stores the user is following. The table will have a username field which will be used as the key to link to the User table using JOIN
- Using this table, the users can see whom they are following.

Wardrobe

- Each user will have a wardrobe to store the clothes item they owned.
- Other than MyWardrobe function, MyWishList function will be stored through this table in our data base system.

Tag

- Tag table stores all the tags used in posts and clothing items. This table will be used to filter searches

Navigation Flow Map

Our navigation flow map focuses on the ease of access. The navigation bar at the bottom of the mobile screen will allow users to access any page with one tap, irrespective of which page they are on at the moment. The login page gives fast access to login through Facebook or Google so the user doesn't have to spend time typing a username and password. This eliminates the time wasted in re-typing passwords and usernames due to spelling mistakes.

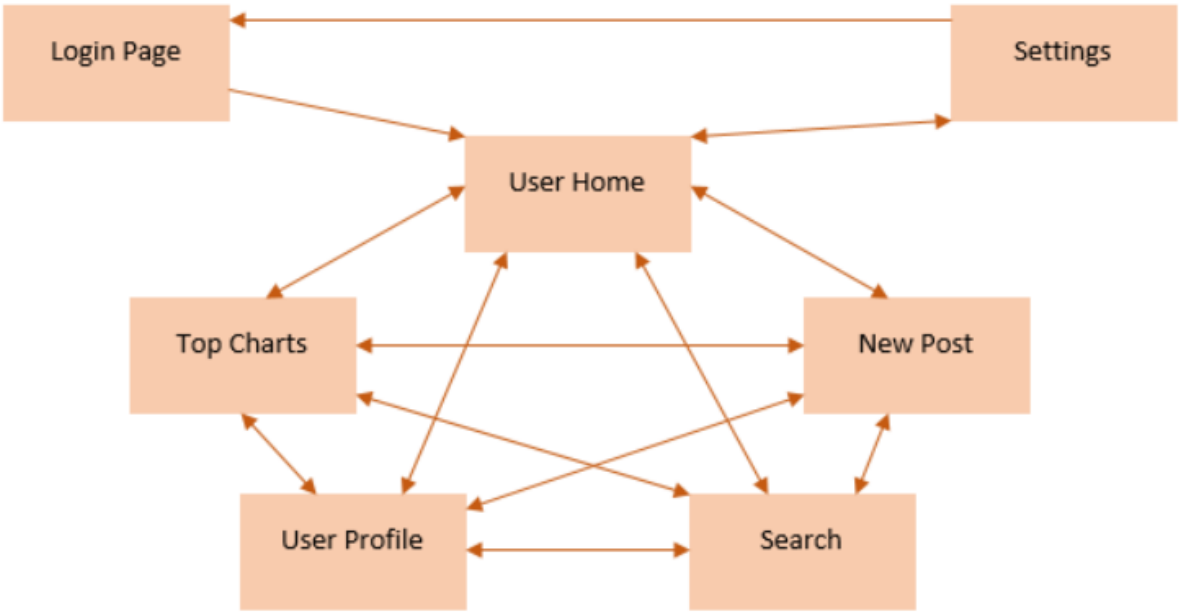
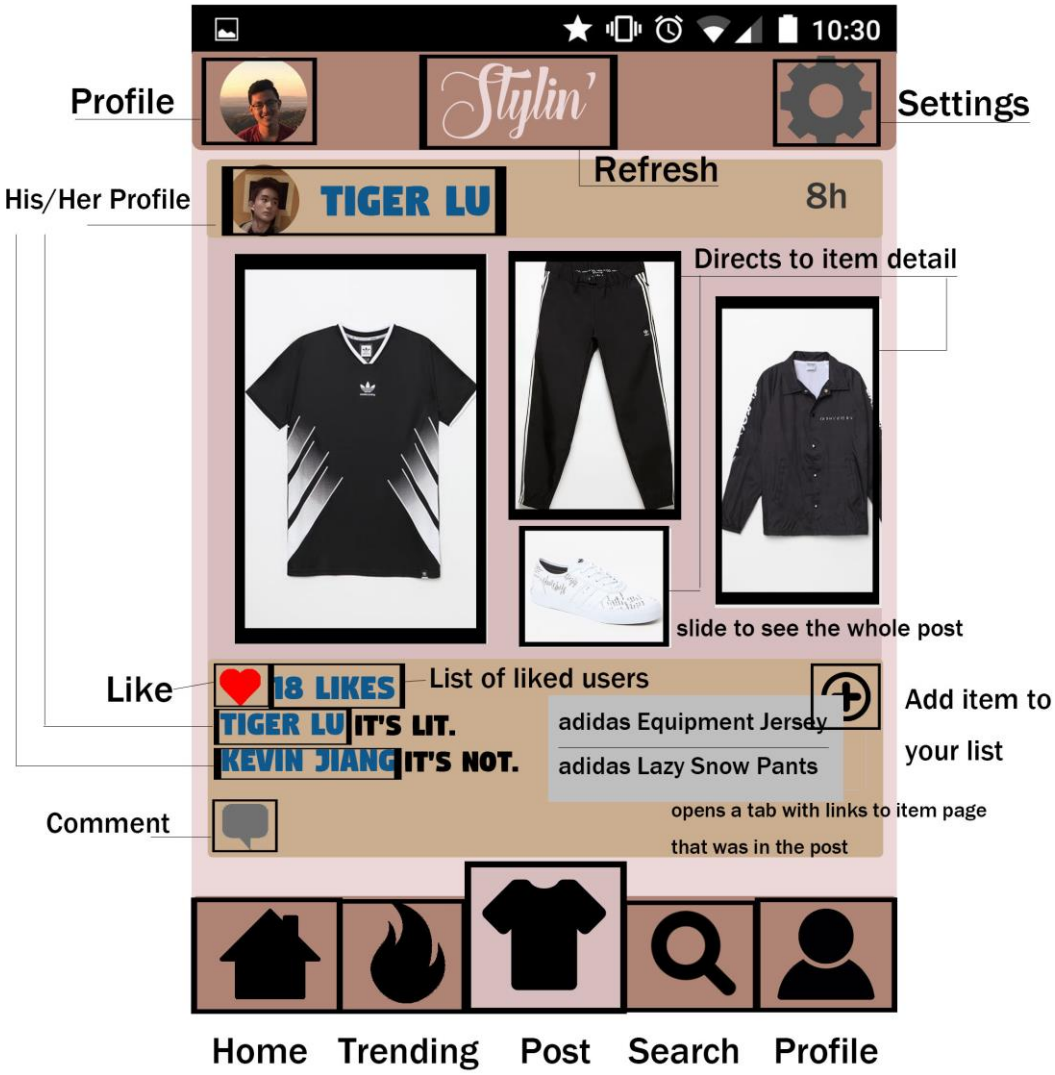


Diagram illustrates ease of access provided by the app. The central pentagon can be accessed via the navigation bar at the bottom of the user's screen. All the tabs are available to the user at all times and doesn't prevent the user from visiting any of the pages at any time in any order.

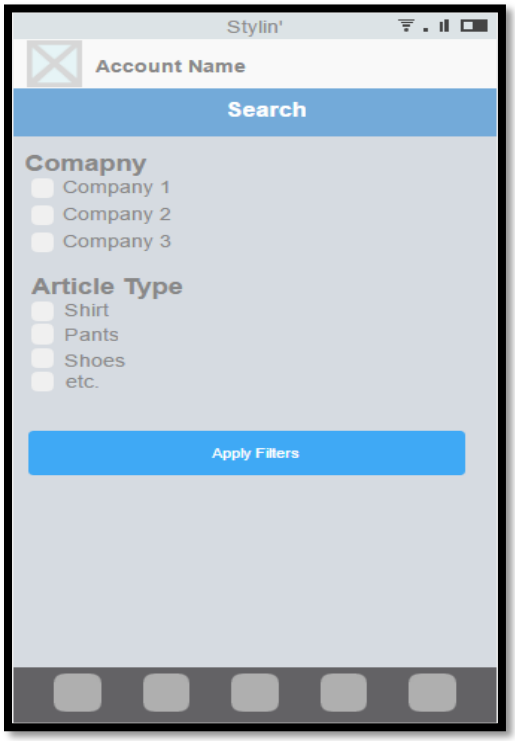
UI Mockups

We designed the UI to be consistent through the app. A finished example of the User Home along with the navigation bar at the bottom. More simple designs of the Login, Top Charts, User Profile, and Search.

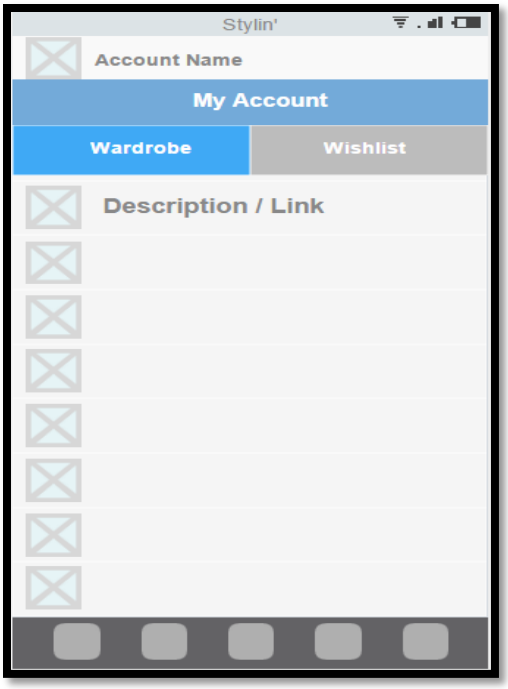


This is the landing page after you log in. It includes your profile picture, profile name, posts from people you are following, and the bottom navigation bar.

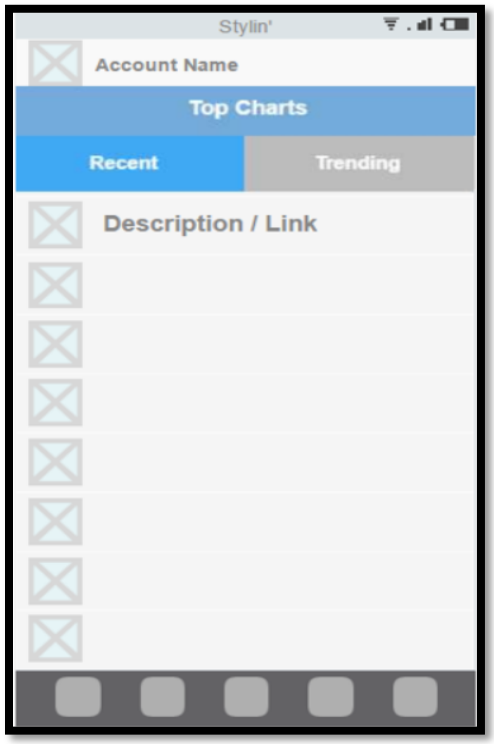
Mobile Login Page



This is the search page. As simply searching "shirt" will display unwanted results, we implemented a filter based search system. The filters will help narrow down the search parameters and speed up the searching process.



This is the account page of the user. Over here they will be able to view their wardrobe, i.e., the clothes they own, and their wishlist, i.e., the clothes they liked.



This is the Charts page. Over here the user will be able to see the most recent clothes in the market, and the most liked clothes in the market.