Team 28 - Product Backlog

Aaryan Srivastava, Aditya Patel, Atharva Gupta, Rishabh Pandey, Sanjhee Gupta

Problem Statement

Efficiently managing a wardrobe, crafting the perfect outfit, and discovering new fashion pieces can be challenging and time-consuming for many individuals. Our project aims to revolutionize these processes by developing a sophisticated management system for clothing selection. Our solution is designed to alleviate the overwhelming nature of a growing wardrobe, making fashion-forward choices more accessible and enjoyable for everyone.

Background Info:

Audience

Modern day closets often hold various clothing for different events ranging from partying to business formal events to everyday wear. It is surprising how despite a huge target audience, existing virtual wardrobe apps lack the fashion-forward choices to make choices more accessible and convenient. Our target audience includes people who want to create a better lifestyle and have a virtual organization of their own closet. Additionally, with the rise of teenagers on social media, friends can interact and like their planned fits.

Similar Platforms

Current Smart Closet applications on the market, like GetWardrobe Outfit Planner, OpenWardrobe, and Acloset, have a similar management system when displaying a user's clothes, but they often require links to brands to upload the clothes onto the smart closet. Our aim is to simplify this process by simply adding a scanner feature.

Limitations

Our hurdle to acquiring users is the initial time investment of inputting clothes into our application. One of our main goals for this project will be streamlining this process and making it feel natural and easy. This will be one of our main differentiators from our competitors, and will set us apart.

Requirements

Functional Requirements:

As a user, I would like to:

Settings friends calendar shop closet

1. Create an account that I can sign in and out of.
2. Have a biometric login with Face ID/Touch ID.
3. Have secured entry with login and password.
4. Input clothing data (size, preferences)
5. Get shopping recommendations based on user size, style, and preferences
6. Have the option to change my account's credentials.
7. Get photo album/camera authorization.
8. Take pictures of my clothes and upload them to the app.
9. Optimize and standardize all clothe pictures on the app to have a white background.
10. Include an experimental outfit planner, where the user can scroll through different clothes you own to find a desirable combination.
11. Be able to categorize my clothes based on tops, shorts, jackets, pants.
12. View an outfit of different clothes I've selected and see how they fit together.
13. Create a calendar feature which allows users to plan/save outfits for certain days/events for more organized planning.
14. Plan my outfits in advance for future events – Add outfits to the calendar future dates to plan outfits in advance
15. Create a outfit from clothing items inputted by the user
16. View clothing items inputted
17. Sort clothing items by type, size, and color
18. Allow users to favorite clothes.
19. Randomly generate outfits based on different clothing types (recommended)
20. generate outfits based on machine intelligence and style
21. Set up a database that stores clothing information + pictures
22. Be able to access my saved outfits.
23. Create a recap that replays all previously saved outfit
24. See my past outfits and what day I wore them.
25. Get a reminder at the beginning of the day about my planned outfits.
26. Send friend requests / accept friend requests
27. Be able to block users from seeing my profile
28. Be able to remove users from my profile
29. View/edit all the friends I've added to my feed
30. Incorporate a feed into the app which shows posts from your friends and yourself.
31. Allow users to react to one another's posts.
32. See what my friends are wearing by encouraging interaction through posting.
33. Share a picture of my outfit once a day, having the clock to post reset at midnight.
34. Be able to add URLs to my shopping cart to see how well they pair with my current closet.

35. Be able to search for clothes needed to match an outfit
36. Be able to navigate to company's shopping page when selected an item not already owned
37. Swipe left and right based on desired clothing recommendations
38. Pin my favorite outfits onto my profile
39. Be able to view fits on a virtual mannequin
40. Wishlist feature for the future fits through the shopping page
41. Search for types of clothing in the shopping page
42. Add a marketplace which imports clothing items from stores.
43. Marketplace features brands and uses affiliate links to raise money.
44. Add a review system for the marketplace that will allow users to leave reviews on items in the marketplace.
45. Include a settings option with a marketplace that allows you to search for specific brands

Non Functional Requirements:

Organization

We plan to develop the application with a separate frontend and backend. This will allow to effectively divide our work, and make it easier to avoid compatibility problems between frontend and backend. The backend will model a RESTful API written in Javascript, using the Node.js and Express.js frameworks. Node.js is known to be highly efficient for building scalable network applications. Express.js is a minimal and flexible web application framework. This stack is known for its performance and scalability, particularly in handling asynchronous operations and real-time applications.

The frontend will be developed using the SwiftUI framework, and will pull data from the backend via requests to the API. Separating the front and back ends will make it easy to expand to other platforms in the future should we choose to do so.

Architecture and Performance

Our architecture's front-end and back-end will be divided during development. The backend will be developed using Python on an event-driven style of architecture. Having our backend operate on requests makes scalability and reusability of code easier. Our database will be hosted by Google Firebase. Firebase provides several services for IOS development such as metrics, image labeling/scanning APIs, and notification setups that we can use to better facilitate the creation of our app. Monitoring the app for unexpected crashes is also provided by Firebase.

The front-end will be using SwiftUI to create all the screens and interactive elements of our app. SwiftUI is also highly compatible with Firebase, making integration of Firebase tools with our front-end much easier. This front-end will send and receive requests to our back-end for integration.

Security
Proper security is essential for OOTD, as we want to make sure all users are comfortable sharing their data and information. Client side security will include secured login with authorized biometrics. All user data stored on OOTD servers will be encrypted using AES Encryption. Server side security will include OAuth 2.0 flows for all RESTful API Calls, and hidden env variables.

Usability
The interface should be simplified and easy to navigate for users to upload, pick, and scroll through clothes. With our features, we would have to ensure that the interface is well designed such that all the clothes are not cluttering the main page. We are going to centralize this on an iOS based platform where users can access our application and be able to fit within the constraints of any iPhone display.

Hosting/Deployment
We plan on using DigitalOcean to host our server side code. For the client-side application, we will deploy with the iOS App Store for production and Testflight for development.

Quantification
Google Firebase allows 1 GiB of stored data for free, along with 50k document read requests and 20k document write/edit requests. Our app can handle 200 users with the current rate. DigitalOcean provides automatic HTTPS and custom domain deployment free of cost, and 4 GiB of memory and 2 vCPUs for 6 months for hosting our server.