efficionado

Optimizing your clothing purchases

MIMS Capstone Project 2017 Report

Ganesh Iyer (ganesh.v@berkeley.edu)
Goutam Murlidhar (goutam.murlidhar@berkeley.edu)
Hadrien Renold (hadrien.renold@berkeley.edu)

Project advisor: Prof. Steven Weber

UC Berkeley | School of Information

INTRODUCTION

While there are many interesting trends in the design of e-commerce websites, there is generally a consensus on acceptable design practices at an interfacial level. These design practices come from core business needs that identify the consumer experience of their web, mobile and physical interfaces as a key factor on whether a consumer visits/uses/purchases products and services from that interface. A standout example of why user experience is crucial to stores is the case study of how Apple emulated the customer service practices of the fabled Ritz-Carlton hotel and implemented that into their Apple stores, codifying it as 'The Apple Experience'[1].

While design patterns in e-commerce sites have varied over time, there are a few key trends that have remained fairly consistent. These include an interface that allows for a user to search and browse through an inventory, filters to narrow down the inventory, a recommendation system that suggests other items from the inventory, a checkout flow through which a user is seamlessly funneled through without aborting the process as much as possible. At a bird's eye view of *all* e-commerce websites, the design decisions are fairly justified considering the underlying assumptions of user behavior.

However, user behavior does not remain consistent while shopping across different products and services. When shopping for books or flights, the search interface suits users' needs because of a valid assumption that the users are more aware of what they want to read or where they want to fly. In case they do not, the recommendation interface steps in suggesting other books based on the user's purchase history or hotels based on the user's destination.

We find from our personal experiences that this awareness drastically reduces when shopping for clothes. Tracing our journey from start to finish in looking for *a shirt*, we observe the process is not seamless thereafter. As opposed to books and flights which are reused at a much lesser frequency post-purchase, shopping for clothes becomes in most

---

[1] How the Ritz-Carlton inspired the Apple Store: https://www.forbes.com/sites/carminegallo/2012/04/10/how-the-ritz-carlton-inspired-the-apple-store-video/#2d1af8063449

cases, a *strategic* decision - one that we argue is not well supported by today's search and recommendation interfaces.

EXPLORING AND VALIDATING ASSUMPTIONS

"Imagine that you are looking for a shirt."

Using this starting point as an investigation into interfaces for clothes shopping, we did a walkthrough of the process of buying a shirt. Our starting point into any interface, physical or online, was first met with a befuddling array of possibilities that the store presents to you when you enter the interface. Thoughts that we expressed in this walkthrough were to the tune of '*this red shirt could look good on me*', updating our desires from the time we entered, based on the inventory presented. From this experience, we hypothesize that with a wide range of choices available, not just limited to one clothing store, the user has to undergo a certain degree of search cost in eliminating less desired possibilities from the inventory.

The strategic aspect of buying clothes becomes stronger when choosing between a handful of strong candidates for the final purchase. In cases of making a *single* final purchase, we hypothesized that the user might prefer to select the candidate that would add the most value to the user's wardrobe. In simpler terms, pick the one that would go well with what you already own.

To evaluate our hypothesis, we conducted a short qualitative survey to users online over common pain points that they experience when shopping for clothes. We find in our study that users expressed a need for understanding and evaluating what they already own (**"I need to consider what I already own"**, **"I need to remind myself that my closet is full"**, **"I need to check my wardrobe"**) and also to know from an expert or a friend what would work well (**"seek opinions, confidence in what I'm getting for myself"**, **"know that I am buying something that will make me stand out"**, **"I'm on my own :-("**).

In our study we also note a few other observations that give us a deeper insight into user's pain points while shopping for clothes. Observations like

"look for something new, but I usually wind up going to the same stores and buying the same things"

indicate a desire for something more refreshing than their existing wardrobe but we argue that existing interfaces are not equipped to handle this desire as users swing back to their default buying patterns.

In terms of needs not expressed directly in our survey, thoughts like

"I feel bad when I spend all day shopping but didn't find anything I like",
"a high quality classic item that I'll have for a long time is always worth it",
"often I feel I just want to get it over with and end up with something I don't really love"

point strongly to cognitive dissonance, leading up to buyer's remorse. Shopping for clothes thus could be emotionally taxing, lead users to question the wisdom of their purchase and express concern over being taken advantage of in the situation[2].

If we look at the findings from the perspective of someone buying books or looking to book a flight, it is difficult to imagine that these same quotes will apply to them. We argue that shopping for clothes is a vastly different experience from shopping for books or flights and hence, deserves a shot at a more specific needs-driven interface.

Other concerns expressed in our survey were related to sizes (XS, S, M, L, XL, XXL), fit (slim fit, baggy fit etc.) and availability in the store. For the alpha version of our proposed interface, we delegate addressing these issues to a later version.

---

[2] Buyer's Remorse: https://en.wikipedia.org/wiki/Buyer%27s_remorse

PROBLEM STATEMENT AND EARLY INSIGHTS

As a segue into our problem definition, we aim to build a search interface that addresses the following user needs:

While shopping for clothes,

- **a user needs information on what she already owns**
- **a user needs to know what goes well together**
- **a user needs to know what they would or wouldn't regret buying**

In relation to the problem of cognitive dissonance, there was a peculiar insight into our survey on *wastefulness* as a human characteristic that users wished to get rid of. We also aim to target this additional need through our proposed interface.

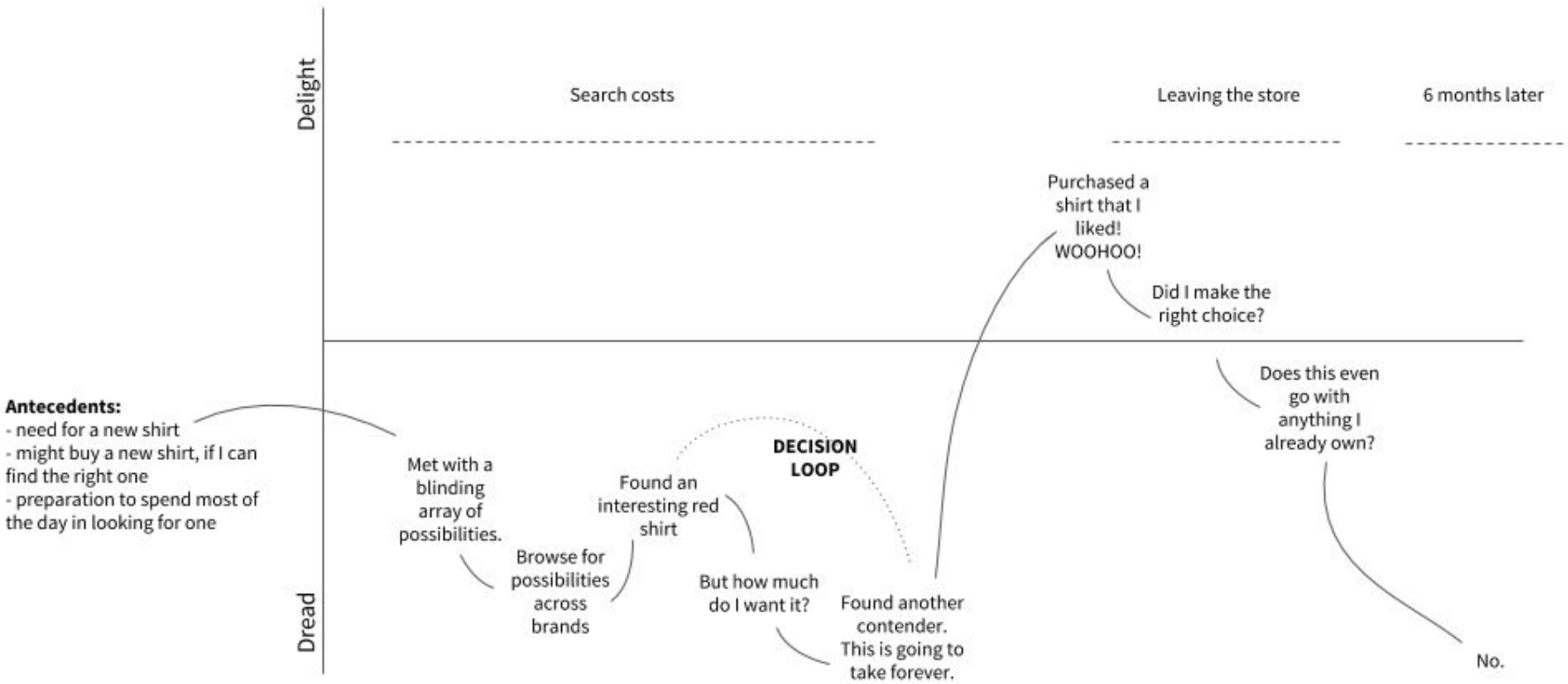
Our approach began with creating a journey map (see next page) and a value proposition canvas (and the page after) to generate insights for the ideation process.

One of the key factors that raise the search cost for a user in looking for the right purchase is the fact that they have to first ***query the eventual purchase itself*** *(search for a red shirt before wanting a red shirt)*. This might be true for use cases where a user is specifically looking for a solid red shirt to wear for a certain occasion, however in cases where a user *eventually decides* on purchasing a red shirt among the huge inventory at the store, what should the 'search query' look like?

A second key insight from our analysis of the survey responses was understanding how much users ***evaluated a piece of clothing in isolation***. All three needs listed above indicate that a piece of clothing is rarely seen in its own merit (*for example, a red shirt*) and current interfaces (both physical and online) that place it against comparable choices of the same category (*shirts of other colors and variations*) do not inform the users enough on how *wise* a purchase the red shirt itself would be for that user, keeping in mind the user's wardrobe and preferences.

Buying a shirt

Beneficiary: User walking into a clothing store



The Value Proposition Canvas

Value Proposition An interface to search better

Customer Segment A user looking for help in making a clothing purchase.

Products & Services

- Search interface with its own intelligence
- Personalized recommendations
- A feature to take a picture of a clothing item and getting feedback
- A feature to take a picture of an item in the closet to generate suggestions for later

Gain Creators

- Generating suggestions with what they already have or would like to buy.
- Have a machine learn their patterns and determine their own styles

Pain Relievers

- Allowing users to recall what they might already own in their wardrobe
- Providing a way to make sure that a purchase now is not something they'll regret later
- Cutting short the time to pick what kind of clothing item to purchase

Gains

- Getting a second opinion
- Knowing what goes well together
- Monitor my fashion tastes
- Extend my tastes to be more ambitious

Customer Jobs

- Browse items of clothing and decide which one to buy
- Personalize appearance

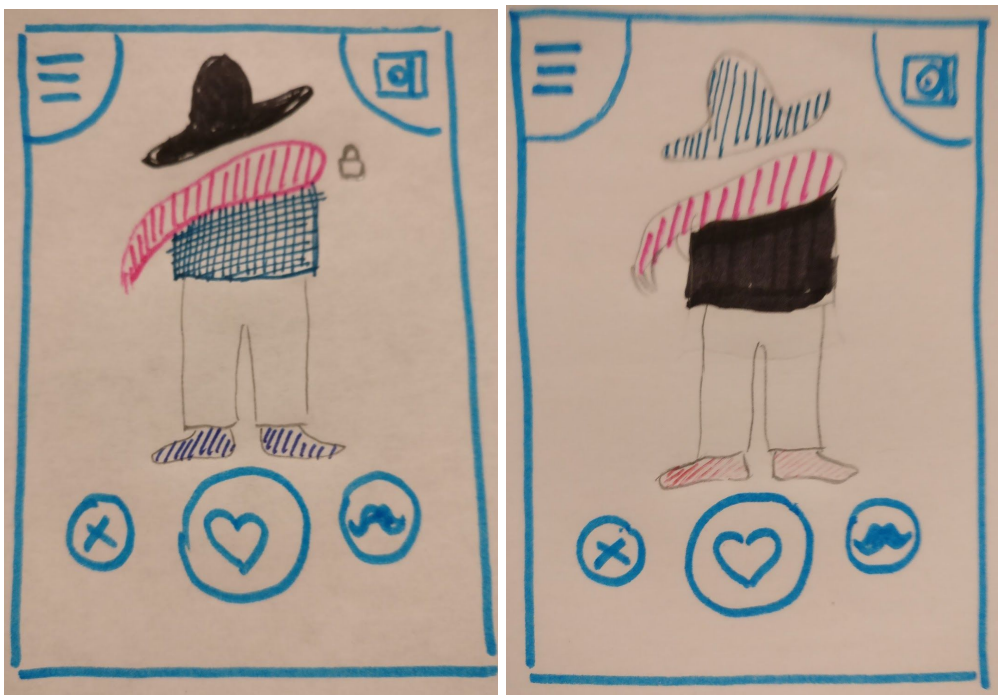
Pains

- Not knowing what is in my closet
- What do I put in this search box? :-/
- Spending money on something I didn't need
- Not being able to choose between outfits
- Time consumed in making a choice that ultimately wasn't great

Another insight was that the system needs to address the cognitive dissonance by being accessible to users as they shop for clothes in physical stores.

APPROACH

Our ideation process began with paper sketches (shown below) north-starring[3] a potential flow for the MVP.



In the picture above, the three key design decisions are detailed below:

1. A *result* would be presented as an entire outfit *that goes well together*. This serves a few key user needs. Firstly, giving an idea of browsing potential outfits will enable users to be more adventurous outside their default buying patterns. Secondly, users can also glean from the interface how well individual items go together and decide for themselves if any item in the outfit is worth further

[3] Design's North Star by Julie Zhuo: https://medium.com/the-year-of-the-looking-glass/designs-north-star-d469193063c5

exploring into.

2. As an additional feature on top of the possibility to determining good outfits, the user can also interact with the outfit. The user can *lock* a certain item in the outfit and browse other outfits with that locked item persistent throughout all outfits. We hypothesize that this has a twofold advantage.

   Firstly, **a user can test the versatility of a clothing item she is seriously considering to buy**. If the outfit is not versatile across many outfits in the store, the user can deem this outfit to not be a wise purchase considering other purchases from the same store in the present or in the future. As the user browses through other outfits with the locked item, the user can also decide if the locked item is versatile or goes really well with an outfit that strongly resembles something from her wardrobe, thus having more clarity in deciding if it is a wise purchase with what she already has.

   Secondly, the user can actually lock an item **to test the versatility of a clothing item she already owns**. If the user finds a pair of blue trousers that strongly resembles something in her wardrobe, she can get a ready set of recommendations by locking that blue item and generating outfits that go well with the blue trouser.

   We label this pattern - ***the anti-search***. We see conventional search as 'you query a red shirt to get a list of red shirts'. However, the anti-search is 'you query a red shirt to get results from matching trousers, shoes and accessories'. Since a conventional search UI is more likely to be used in the former manner, we decided against incorporating it in our interface, instead opting for a browse-only interaction that allows users to pick a query based on something they see.

3. The third key design decision was to incorporate the flow into a playful Tinder-esque interface allowing the users to judge the merit of one outfit at a time. Through the lock function the user can judge the versatility of one item by recognizing how many outfits the item goes well with, however not all outfits that

are deemed by the system to go well with the locked item are equal. The *one-outfit-at-a-time* approach helps the user to bring in their personal tastes and styles and also see an outfit that they think extends their style and be more adventurous.

From a point of view of *wastefulness*, it may at first, seem counterintuitive to expect users to be economical with their clothing purchases when presented with an entire outfit as opposed to an individual item. However, we argue that this deserves a second thought as users might now have a greater grasp on their own inventory before making a purchase for a versatile clothing item.

Now to address the big elephant in the room. How does the system decide **what goes well together?**

GENERATING OUTFITS

The abstract concept of style is fundamental to generating what goes well together in an outfit. There are two approaches to tackle this problem that can be performed in conjunction or independently on a raw inventory of clothing. The first approach is to analyze users' buying patterns on a store to generate a model. The second approach relies on experts seeding a model based on what they see as good combinations. For the latter approach, stores can choose to recruit an in-house expert or hire the services of an external expert. However, we felt that for the solution to gain traction in the market, we first needed to evaluate the merits of our solution to justify investing in experts in the first place, a classic Catch 22. For this reason, we chose to proceed with the former approach - analyzing user's buying patterns on a store inventory to generate a model.

We generate outfits based on the approach presented in **Image-based Recommendation on Styles and Substitutes**[4] by Julian McAuley at UCSD. The goal of this algorithm is to

[4] McAuley, Julian, et al. "Image-based recommendations on styles and substitutes." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2015.

learn an abstract concept of style from a very large “raw” dataset of user interactions with items. By “raw” we mean a dataset that was collected based on normal user browsing and purchases but not specifically related or curated to style. The assumption is that the scale of the dataset will filter out the noise and bring out relations related to visual features[5] and hence, hopefully style. To train the model we used an Amazon dataset consisting of 180 million relationships between 6 million objects. Relationships are defined from the browsing history of Amazon viewers and are of four types - “also viewed”, “viewed then bought”, “also bought” or “bought together”, with the assumption that the first two represent cases where a user substituted one item for a more preferred option (*substitution*) and the last two represent cases where a user made complementary purchases (*complements*).

Each of the 6 million objects contains metadata which includes the image url and categories as per the Amazon category hierarchy. From the images of each object we also have 4096 image features generated with a neural net. Based on the image features and the relationships, the algorithm assigns weights to the relationship between image features[6], thus creating the learned model.

Breaking down their image features allows to generate associations between objects based not only on the user’s browsing or purchase history but also on their visual features. This also allows product purchases that are apparently unrelated, for example different categories (sports outfits v/s. formal outfit) to both participate to reinforcing the same association of image features. For example, if Bobby purchases two items of a sport outfit, one white T-shirt and a pair of black pants, and Ashley purchases two items of a formal outfit, a white shirt and a pair of black shoes, this would reinforce the weight associated with an abstract rule of white and black items fit well together. The uncertainty is because it is very hard to interpret exactly the features learned from the neural net and the “rules” learned by the algorithm.

---

[5] This to a certain extent has been validated in the paper.
[6] For simplicity’s sake the exact formulation of how weights are computed and the computational tricks to make this efficient have been omitted.

We can now query this model to extract associated items for an existing image in the dataset: based on the feature weights we collect the best matching results. We can also query it using a new image, extracting its features and comparing it with images in the dataset and finding the closest match based on those. This allows us to get recommendations for an object that has never been seen before and expand on the existing model. Currently the algorithm takes a lot of time to run, because the data files are very big (in the order of dozens of GB when compressed) and the computations are time intensive so that it is not possible to run this process online, but an offline processing is possible and with more work the algorithm could potentially be optimized to run online. To generate clothing outfits we use the categories metadata associated with each object to extract items in the "Clothing, Shoes and Accessories" category. We then used a definition of outfit as a set deriving one item from each of the following categories:

Men: '*Pants*', '*Shirts*', '*Shoes*'
Women: '*Accessories*', '*Pants*', '*Tops & Tees*', '*Shoes*', '*Dresses*'

We made a slight adjustment after noticing that in a single outfit for women, '*Dresses*' do not go along with both '*Pants*' and '*Tops & Tees*'. A single outfit now would be based on:

Men: '*Pants*', '*Shirts*', '*Shoes*'
Women: '*Accessories*', "*Shoes*', '*Dresses*' ***OR***
Women: '*Accessories*', '*Pants*', '*Tops & Tees*', '*Shoes*'

These outfits were determined both from *a priori* perspective on what could fit together and on result inspection on what makes sense from an outfit perspective and from a search interface perspective. As we will discuss later, there are many different ways to define outfits, this is just one possible way to do so. Finally, from the model we extracted the top 2000 outfits for men and women and use those as the basis of service.

IMPLEMENTATION

A. Design decisions

We had initially intended for the system to be accessible when users are in physical stores. The underlying use case behind pursuing a mobile interface was for users to be able to enter a query by taking a picture of a clothing item with their phone camera. However, since the computations are time intensive and the captured image would take a long time to be 'featurized', we opted against this platform with the hope that with improving technical feasibility, we could incorporate it into the product. For the alpha version, we focused on a web application.

Based on our initial sketches we also implemented an interface which serves *one outfit at random*. The outfit contains a list of images for each clothing item that can be *locked*.

The user can choose to filter down items by gender, however we did not have an informed sequence in which the user prefers to browse through outfits. We ran into a few obstacles here. The images from the database sometimes returned an 'image unavailable' image making part of the outfit irrelevant. A second obstacle was multiple clothing items being advertised in one image which came in the way of clarity of which item is being advertised. A third constraint was that the visual styles across all images of clothing items in one outfit were not consistent as they need not have come from the same manufacturer, or had a model wearing them.

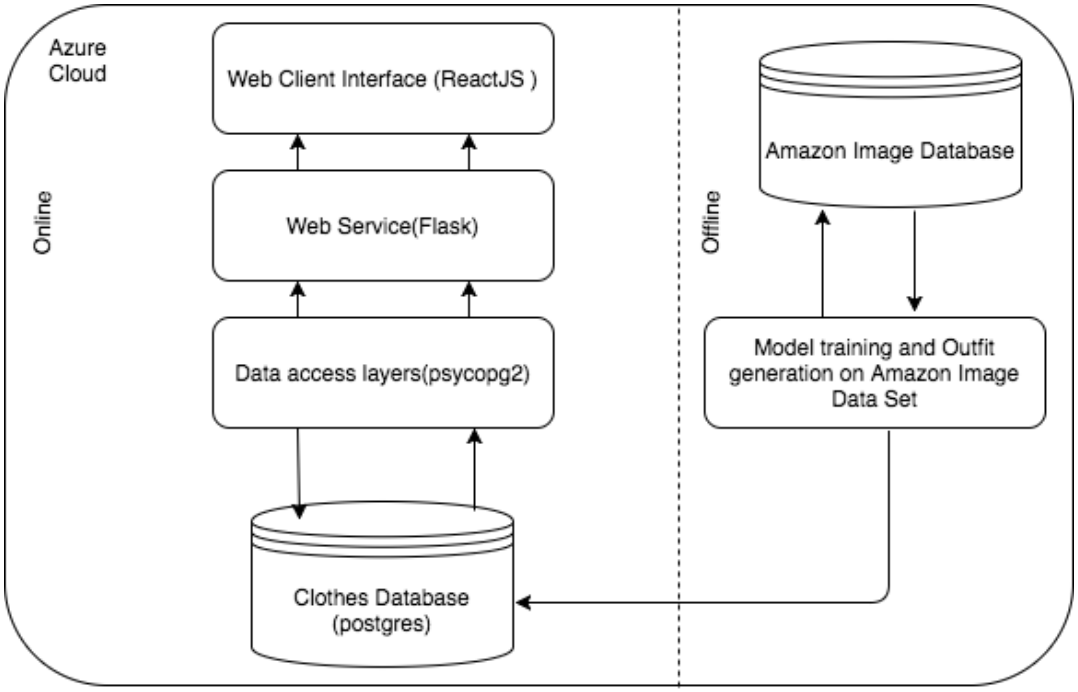
Locking an item allows the user to continue browsing other outfits with the locked item persistent throughout. As explained previously, this constitutes the user's query to see which other clothing items go well with this locked item. On disabling the lock, the user can continue to explore outfits randomly under the constraint of gender, if selected.

To validate if users like the outfits returned to them, we incorporated a 'Like'

functionality for each outfit. This allows a signed in user to retrieve a list of liked outfits at a later time to evaluate potential future buys. For a v2, this could be extended to a user setting a price tracker for a liked outfit so that a purchase could be initiated at a time when the desirable outfit is available at a low price. A more ambitious direction would be to evaluate a user's personalized style through further machine learning and suggest not only those outfits that strongly align with the user's tastes but also those which adventurously extend it.

The 'Like' feature also allows us to augment the results of the machine learning algorithm with end user input. This could potentially be integrated further to be fed back into the machine learning algorithm to improve its results.

B. Technological Stack



The diagram above provides an overview of our architecture, which is composed of an offline layer and online layer. All our service is hosted in the cloud, on Azure linux virtual machine instances. The offline layer is for training the model and to generate outfits as described in an early section and is written in C++. The online

section is largely based on a python infrastructure. It is made up of three major components: the database, the data access layer and the web service layer.

The starting point for our clothing databases was the set of men and women's clothing which are loaded into Postgres object relational database. We chose to use a relational database over a NoSql solution for the enhanced querying support that a traditional relation database solution provides. To interact with the database we use the psycopg2 module which provides us the necessary abstractions to efficiently use the database. Once outfits and items were loaded in our database our entire application was powered from it. The technical architecture we chose for our application was to make the data model structure transparent to the application.

The Data access layer provides an interface to the underlying database. Any access to the database goes via the data access layer. This architecture allows any code changes required for changes in the data model to be isolated in one component. Since every user interaction required us to go to the database either to serve data or insert it we made a design decision to use vanilla SQL with a cursor interface provided by psycopg2 and not use an ORM layer. As we do not have a caching layer we believed the performance hit of an ORM layer would adversely impact interactions of the user with the system. While in some cases the data access layer simply acts as a pipe to fetch and push data, in other cases we chose to process the data there and pass it to the web service layer, which forwards it to the frontend as JSON.

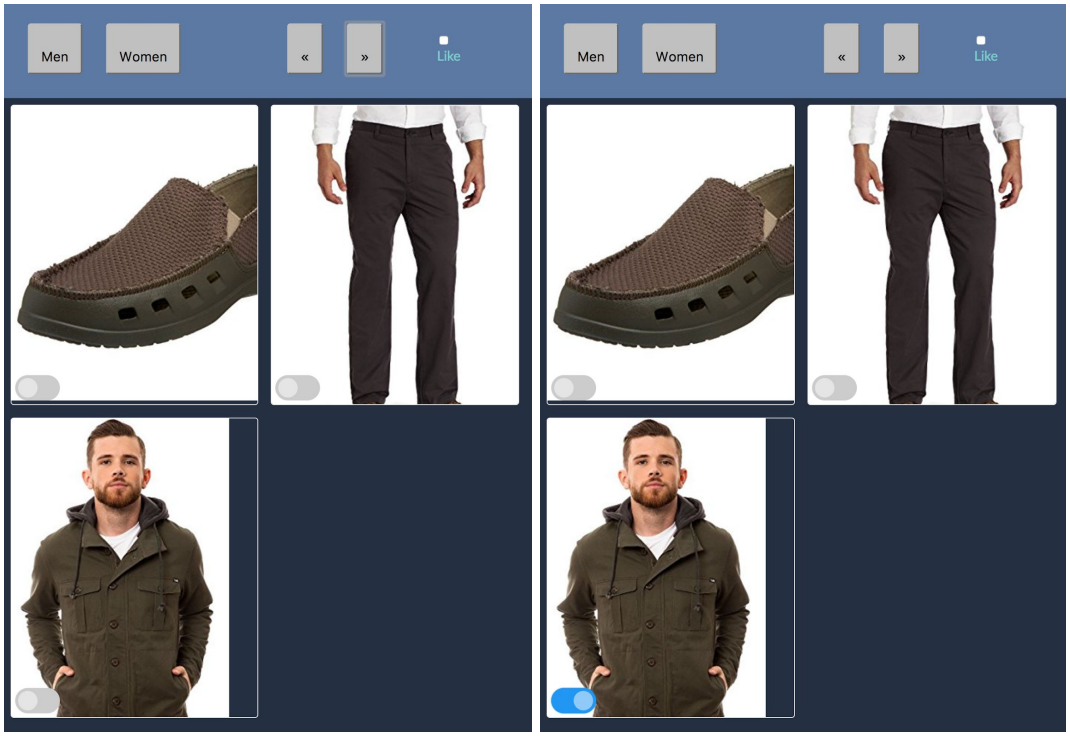
The Web service layer is the external interface for our system.It is designed such that any application not just our user interface can make requests to our system. All services are exposed as HTTP Rest endpoints. While some services require authentication to work all service inputs are generic and are either in the form of query string parameters or json message bodies.This allows potentially other external applications or our own alternative front ends like for instance a mobile app to use the same web services to push and pull data. In essence the web service layer is the external world facing endpoint for the data access layer. As we

described in the data access section in some cases the data access layer does the heavy lifting and in some cases the processing is done by the web service layer.

The frontend uses Facebook's library React to populate web pages through modular components. The components are written in ES6 (ECMAScript, an advanced version of Javascript that current browsers do not support) and JSX (a mix of HTML and Javascript in the same segment of code). At the front end, Webpack bundled all the files into individual CSS, HTML and Javascript files with the help of the Babel transpiler that converts ES6 and JSX into vanilla Javascript that current browsers *can* read. For the styling, we wrote SCSS code and supported it with the mixin libraries, Bourbon and Neat. For the development, we also ran a Webpack dev server that helped us to reliably test our changes.

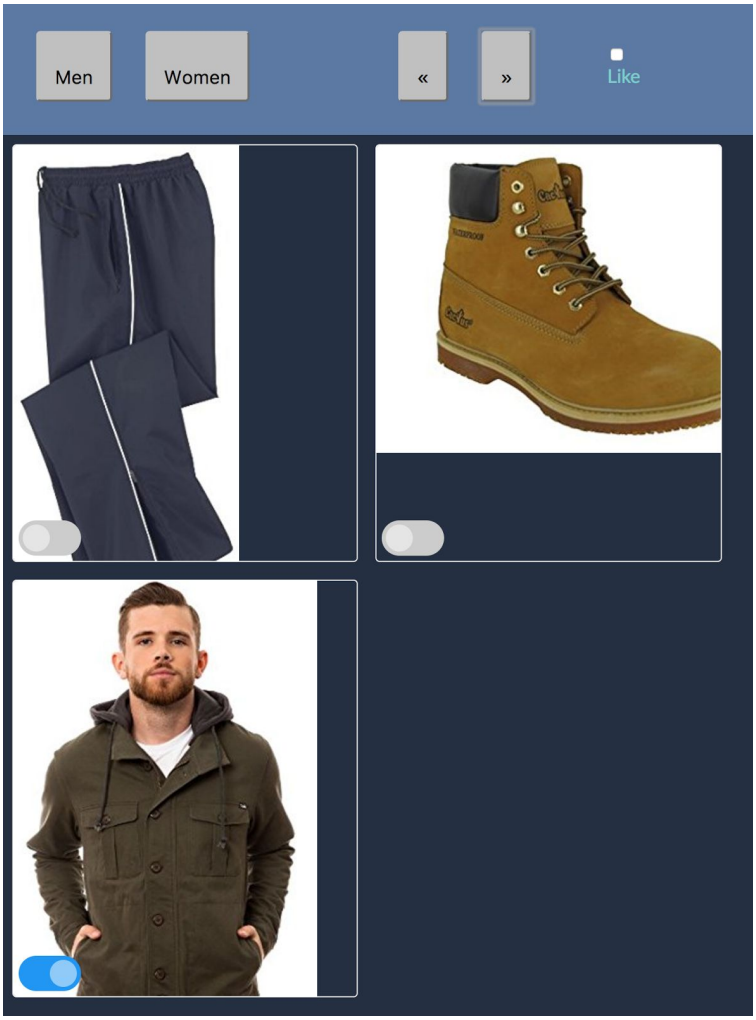
RESULTS AND EVALUATION

On implementation, the interface returns outfits consistently and reliably. Attached below are screenshots of a sample outfit.



This was followed by locking an item (right) to let it persist as the user scrolls through other potential combinations with this outfit in mind (below).

In order to evaluate the quality of the outfits generated solely by the algorithm we presented the platform to users and asked them to like an outfit based on whether they considered at least two items to fit together, i.e. if they would wear it or if they would see someone else (potentially with a different style) wearing it. From the 10 users, 5 women and 5 men, they liked on average 37 ± 18 percent of the outfits. Considering that those are purely generated from the machine learning model from a raw dataset this is an acceptable rate. This experience also allowed us to generate a starting set of liked outfits to present to end users.



At a glance, we can notice some discrepancies which stand out, such as problems with the categorizations of certain outfits, for example presenting multiple items of the same category, sometimes totally unrelated objects. We realised that we should not be presenting the result as a complete outfit but rather as multiple suggestions of items that could potentially fit together.

FUTURE WORK AND REFLECTIONS

While we are confident that there is a core user need and subsequent value in sharpening and iterating over the design of the search interface, we acknowledge that machine learning and analyzing image features to consistently generate outfits that the user would like is an extremely hard problem. To be more effective as a recommendation system, there needs to be additional curation of the inventory. It would also be worthwhile to explore patterns of usage in a more specific dataset such as a store which exclusively deals in clothing goods. In terms of testing the product, it would be an important step to implement this as an alpha and do a longitudinal study on whether it prevents a user from making a potentially regretful purchase.

An important intervention that needs to be made for future outfits once there is a reliable learning algorithm in place is to design for multiple accessories and more ambitious outfit styles - such as a jacket, t-shirt, trousers, shoes, hat *and* goggles.

To round it off as a product, bringing in input from other users (including fashion experts) to see a list of 'Liked' outfits can feed back into the learning algorithm to derive patterns from community behavior.

With companies like Amazon, eBay, StitchFix investing in this domain recently, there is a promising market that lies ahead when looking to strongly reinforce a sense of confidence in users in making these strategic purchases.