Table of Contents

[4 Introduction 4](#_Toc78925894)

[4.1 Motivation 4](#_Toc78925895)

[4.2 Contribution 5](#_Toc78925896)

[4.3 Real world application 5](#_Toc78925897)

[5 Project Design 6](#_Toc78925898)

[5.1 Project Overview 6](#_Toc78925899)

[5.2 Project Architecture 7](#_Toc78925900)

[5.2.1 Use Case Diagram 7](#_Toc78925901)

[5.2.2 Front end Design 8](#_Toc78925902)

[5.2.3 Backend Design 8](#_Toc78925903)

[5.2.4 APIs 9](#_Toc78925904)

[5.2.5 Database Architecture 9](#_Toc78925905)

[5.3 Data Pre-processing 11](#_Toc78925906)

[5.3.1 Exploratory Data Analysis - EDA 11](#_Toc78925907)

[5.3.2 Feature Engineering 11](#_Toc78925908)

[5.3.3 Data Engineering 12](#_Toc78925909)

[5.4 Application Wireframes 13](#_Toc78925910)

[5.5 Machine Learning Models Design 15](#_Toc78925911)

[5.5.1 Content Based Model Flow 16](#_Toc78925912)

[5.5.2 Collaborative Based Model Flow 17](#_Toc78925913)

[6 Implementation 18](#_Toc78925914)

[6.1 Backend and API 19](#_Toc78925915)

[6.1.1 Data Processing 19](#_Toc78925916)

[6.1.2 Database Construction 20](#_Toc78925917)

[6.1.3 Routes and Server 22](#_Toc78925918)

[6.1.4 API Implementation 23](#_Toc78925919)

[6.2 Machine Learning Model 24](#_Toc78925920)

[6.2.1 Content Based Filtering 24](#_Toc78925921)

[6.2.2 Tf-idf Vectorization 24](#_Toc78925922)

[6.3 Collaborative Based Filtering 25](#_Toc78925923)

[6.3.1 Correlation Clustering (Item-Item based CBF) 25](#_Toc78925924)

[6.3.2 Decision Tree (User-User based CBF) 26](#_Toc78925925)

[6.3.3 Natural Language Processing 27](#_Toc78925926)

[6.4 Frontend – Development 28](#_Toc78925927)

[6.4.1 Sign in Screen 28](#_Toc78925928)

[6.4.2 Home Screen 29](#_Toc78925929)

[6.4.3 Like List 30](#_Toc78925930)

[6.4.4 Profile Screen 30](#_Toc78925931)

[6.4.5 Recommendation List 30](#_Toc78925932)

[6.4.6 Search Screen 31](#_Toc78925933)

[6.4.7 Item Description 31](#_Toc78925934)

[7 Testing 31](#_Toc78925935)

[7.1 Automated Testing & Manual Testing 31](#_Toc78925936)

[7.2 Testing Methodologies 32](#_Toc78925937)

[7.3 Unit Testing 32](#_Toc78925938)

[7.4 White Box Testing 32](#_Toc78925939)

[7.5 Model Testing 34](#_Toc78925940)

[8 Analysis and Evaluation of Algorithms 35](#_Toc78925941)

[8.1 Introduction 35](#_Toc78925942)

[8.2 Experimental Setup 35](#_Toc78925943)

[8.2.1 Datasets 35](#_Toc78925944)

[8.2.2 Methodology 36](#_Toc78925945)

[8.3 Results 36](#_Toc78925946)

[8.3.1 Prediction Accuracy 36](#_Toc78925947)

[8.3.2 Top-N Accuracy 37](#_Toc78925948)

[8.3.3 Beyond Accuracy 37](#_Toc78925949)

[9 Conclusion and Future Works 38](#_Toc78925950)

[10 Bibliography 39](#_Toc78925951)

[11 Appendix 40](#_Toc78925952)

[11.1 Project Gantt Chart 40](#_Toc78925953)

[11.2 Singular Value Decomposition (SVD) Results 40](#_Toc78925954)

Table of Figures

[Figure 5.1 System Overview 7](#_Toc78925955)

[Figure 5.2 Use Case 7](#_Toc78925956)

[Figure 5.3 Project Architecture Overview 9](#_Toc78925957)

[Figure 5.4 Database Schema 10](#_Toc78925958)

[Figure 5.5 SQL view for generating user reviews sheet 12](#_Toc78925959)

[Figure 5.6 A part of the data pipeline to categorize similar data from multiple sources 13](#_Toc78925960)

[Figure 5.7 Home Screen Wireframe 14](https://qordatainc-my.sharepoint.com/personal/bilal_khan_qordata_com/Documents/Project%20Report%20V3.docx#_Toc78925961)

[Figure 5.8 User Flow 15](#_Toc78925962)

[Figure 5.9 Content Based Model Flow 16](#_Toc78925963)

[Figure 5.10 Collaborative Based Model Flow 17](#_Toc78925964)

[Figure 6.1 Implementation Workflow 18](#_Toc78925965)

[Figure 6.2 Tokenizing Reviews 19](#_Toc78925966)

[Figure 6.3 Important imports 19](#_Toc78925967)

[Figure 6.4 Python Scrappy Framework Script 20](#_Toc78925968)

[Figure 6.5 Database Connectivity 21](#_Toc78925969)

[Figure 6.6 Data Upload to Firestore 21](#_Toc78925970)

[Figure 6.7 Firestore Database 21](#_Toc78925971)

[Figure 6.8 Global Variable Declaration 22](#_Toc78925972)

[Figure 6.9 Global Functions Declaration 22](#_Toc78925973)

[Figure 6.10 Collaborative Recommendation Writing Function 23](#_Toc78925974)

[Figure 6.11 Server Start up 23](#_Toc78925975)

[Figure 6.12 Collaborative Filtering Route 24](#_Toc78925976)

[Figure 6.13 Tf-idf Matrix and Cosine Similarity 25](#_Toc78925977)

[Figure 6.14 Correlation Clustering 26](#_Toc78925978)

[Figure 6.15 Decision Tree with Gini index and Pruning 26](#_Toc78925979)

[Figure 6.16 Product Review Scores 27](#_Toc78925980)

[Figure 6.17 Vader Sentiment Analysis 28](#_Toc78925981)

[Figure 6.18 Firebase Connection and Authentication 29](#_Toc78925982)

[Figure 8.1 Product List Dataset Summary 36](#_Toc78925983)

[Figure 8.2 User Review Dataset Summary 36](#_Toc78925984)

[Figure 8.3 KNN 37](#_Toc78925985)

[Figure 8.4 Decision Tree 37](#_Toc78925986)

[Figure 8.5 Confusion Matrix for KNN 37](#_Toc78925987)

[Figure 8.6 KNN Hit Rate 38](#_Toc78925988)

[Figure 8.7 Decision Tree Hit Rate 38](#_Toc78925989)

[Figure 8.8 Tf-idf Cosine Similarity Index 38](#_Toc78925990)

[Figure 11.1 Gantt Chart 40](#_Toc78925991)

[Figure 11.2 Accuracy Measure for SVD 40](#_Toc78925992)

# Introduction

## Motivation

Skincare products have become an essential part of everyday lifestyle for a wide range of individuals. In 2020, it was recorded that there was a total of 31.3% consumers that have at least bought 1 skincare product (E. Martin, 2020). This is essentially the outcome of growing consumers interest in natural beauty and skincare consciousness in both men and women. Today people are desperately trying to accomplish this aim of having a picture-perfect skin by constantly searching for best skincare products that resonates with their preferences and objectives. Alongside, consumer interest in cutting-edge skincare products the variety and number of products provided by companies has also significantly increased. Sangeeta S (2019) outlined in her survey report that skincare has surpassed the scale and is now number one out of seven major beauty product categories. Furthermore, the global market of skincare products has increased to $180 billion, with an expected increase of 20-30% in coming five years (Treffis, 2018). The number of skincare brands world-wide that are selling multiple products is constantly increasing. With multiple vendors selling products instore and online the costumers are faced with such a vast variety of products whereas the number of items that are of interest is relatively small. Moreover, such a large number of alternatives makes it overwhelming for customers to identify the right product for themselves. Therefore, it became considerably important to develop systems that are able to intelligently utilise the data about the user’s preferences to recommend and display personally relevant skincare products (S. George, 2019).

The growth in e-commerce has transformed warehouse and distribution centres to mega fulfilment centres. From stores that sold individual items for customers to surf through and complete a checklist warehouse have evolved to compensate people and provide an omnichannel form of business for the merchant. This includes conglomerates such as Amazon, Alibaba and Souq. Supply chains have been administered to meet wnd demand. Instead of people suggesting each other items based on their personal interest, experience or criteria machines have become smart enough to realize just what the customer needs based on the individual’s characteristics. The smarter the algorithm the better the business evaluation. The first time Amazon made its name to the top 10 retailers was back in 2012. It came in 10th position with a revenue total of 34.4 billion US dollars. According to an analysis done by McKinsey & Company, 35 percent of Amazon’s total revenue in the year 2012 came from recommendation engines (Verma, 2021). While Netflix says, more than 75% of their sales are based on algorithmic recommendations alone (C. Sameer, 2017). The first ever recommendation engine dates back to 1992 at the Xerox Palo Alto research centre. It was designed for a mail system which was to suggest the users with the most loved and read document using collaborative filtering.

Therefore, in this dissertation the aim is to study recommendation systems to develop a personalised skincare product recommendation system. Skincare is an interesting domain to apply the concepts of recommendation system since there is a real benefit for the customers and not solely for the purpose of achieving business economics growth. Besides there not being a prior proper approach taken for developing skincare recommendation there have been no vital attempts at making a mobile application that could inevitably implement such a feature. The system would help users by recommending them personally relevant products that resonates with their preferences. Recommendation engines make the process of suggesting items to customer much more effective and efficient, plus they can help:

1. Drive the traffic
2. Engage Shoppers
3. Deliver Relevant Content
4. Convert Shoppers to Customers
5. Reduce Workload and Overhead

## Contribution

There have been a few approaches where the author has implemented either a content-based filter or made suggestion via collaborative based filtering, but not both. (Adebo, 2020) implements content-based filtering using Tf-idf with the concept of comparing ingredients, their active chemical contents and nature of effect they had on human skins. A similar approach has been opted by (Songsri Tangsripairoj, 2018) in the form of a mobile application. This application utilizes ***both approaches*** i.e. content and collaborative based filtering and provides suggestion independently of one another. Content filtering uses the Tf-idf vector to categorize item on the basis of their category, brand and ingredients. Whilst item and user based collaborative filtering implement KNN, Decision trees and sentiment analysis respectively.

Skincare Recommendation approaches taken previously represent their work in the form of a research or just algorithm results. (Junaidi, 2021) present their solution in the form of a web application which takes into consideration only the user characteristics for user based collaborative approach. This application on the other hand presents a ***native mobile application*** in the form of a complete solution that is scalable, responsive and presents the user with multiple results and approaches.

The contribution of this project has not only been on the basis of results acquired, but also the steps taken to prepare the dataset. The collection of the right format of datasets for skincare recommendation is sparse indeed. Kaggle only presents with a cleaned and formatted data for Sephora whilst (Songsri Tangsripairoj, 2018) take into regard just the chemical ingredients and their effects on the human skin. (Junaidi, 2021) describe their working with a data set that contains only user data***. Data pre-processing and engineering*** have played an entirely different criterion in this application. User characteristics were collected, normalized, regularized along with a similar procedure being carried out for user reviews and ratings. A list of 2000+ unique Products was accumulated alongside their reviews, ratings, URL and images. Previous approached all considered data from a single source with less complexity plus limited features to analyse making the algorithm smart, but their prediction was limited to the extent of its knowledge.

Finally, I’ve accumulated a result of ***multiple algorithms*** such as KNN, Decision trees, Vader sentiment and Tf-idf vectorization to discuss on the idea of the which is more superior based on the provided dataset and its features. These machine learning models have shed some light as to why many recommendation systems usually implement these specific algorithms such as why a natural language-based application is preferred for content based or review based recommendation system. The few literatures such as (Songsri Tangsripairoj, 2018) that exists in this domain only consider implementing a single model or algorithm.

## Real world application

Real world application for this project provides a widespread of opportunities for skincare brands. This application can be shipped to users and can provide with more valuable data for better outreach and predictions. It can exponentially increase user engagement and help make any visitor a potential customer. Recommendation engines are a widely used commodity now a days. Any e-commerce platform and or skincare brand can make the use of content and collaborative based filtering formula of the application to draw in more user plus reduce the time taken for customer to any decision. Making this an android application substitutes the need of marketing to reach a widespread audience. The application is developed as a smartphone application on the Android platform in order to cover the greatest number of users as Android dominates the market by more than 80 percent (Songsri Tangsripairoj, 2018).

Another use of the application would be to analyse the trend for uses with specific skin types, tones or any other characteristics in particular. It can prove to be an important feature to gather feedback from users that use a specific brand on a daily basis. Analysis can be gathered around a product whether it completes the purpose it has be designed for i.e. if users with a dry skin condition are really buying a skin lotion and what are their reviews around it.

The application doesn’t work on a predefined or short ranged dataset, it consists of multiple users with different characteristics and products from a wide range or brands hence, any suggestion made will not be limited to a specific product category or brand. It will deliver relevant content to the user’s screen and save their time in search of the right items. Many people hesitate to try new products or new brands as they have no knowledge of cosmetic ingredients, the chemical characteristics of each ingredient, and quality, but this application will help them decided based on the similar products they have used.

# Project Design

In this chapter of the report, the complete system architecture and design of an android application supporting the integration of a machine learning models that recommends skincare products to users is explained in detail and analysed. It also justifies the architectural choices and expresses the logical architecture with use of diagrams that were prepared to support the implementation of the project.

## Project Overview

The application has been designed as a ***3 layered architecture***. It consists of the **frontend** which allows the users to interact with the program, the ***backend*** makes it consistent by providing the algorithms that are needed to process the data whilst dealing with the oncoming user requests, and the ***database*** is where all the data is written and read from. The frontend of the project has been implemented as an android application programmed solely on ***Java using Android Studio***. The choice for backend and server implementation was ***Python*** due to its flexible nature which provided an ease and reliable way for implementing machine learning architecture whilst also providing great web frameworks. ***Firebase*** provides a serverless architecture and acts as a middleware for the application. Figure 5.1 shows the overall architecture of the application on a high level.

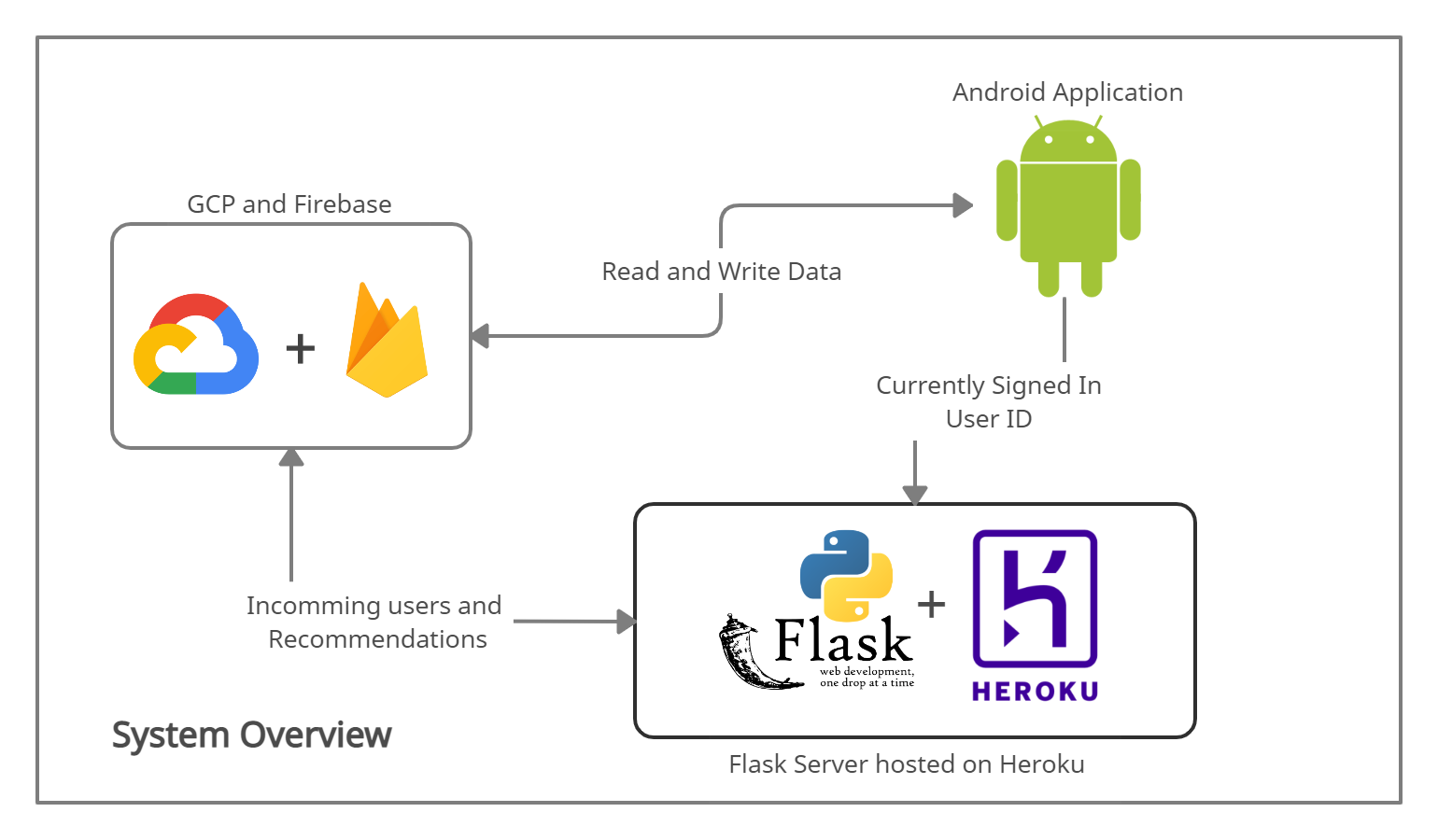


Figure 5.1 System Overview

## Project Architecture

### Use Case Diagram

Use cases carry an important aspect for the designing of the application. Figure 5.2 gives an overview to the use case created displaying the different interactions from a user’s perspective within the system. The application provides a basic login form, signup, and authentication. The user can see a list of products and brands on the home screen. From here they can traverse to every single item’s description and add it to their like list from where the recommendations are generated. Products or items can also be reviewed/commented on and rated based on a 5-star rating level.

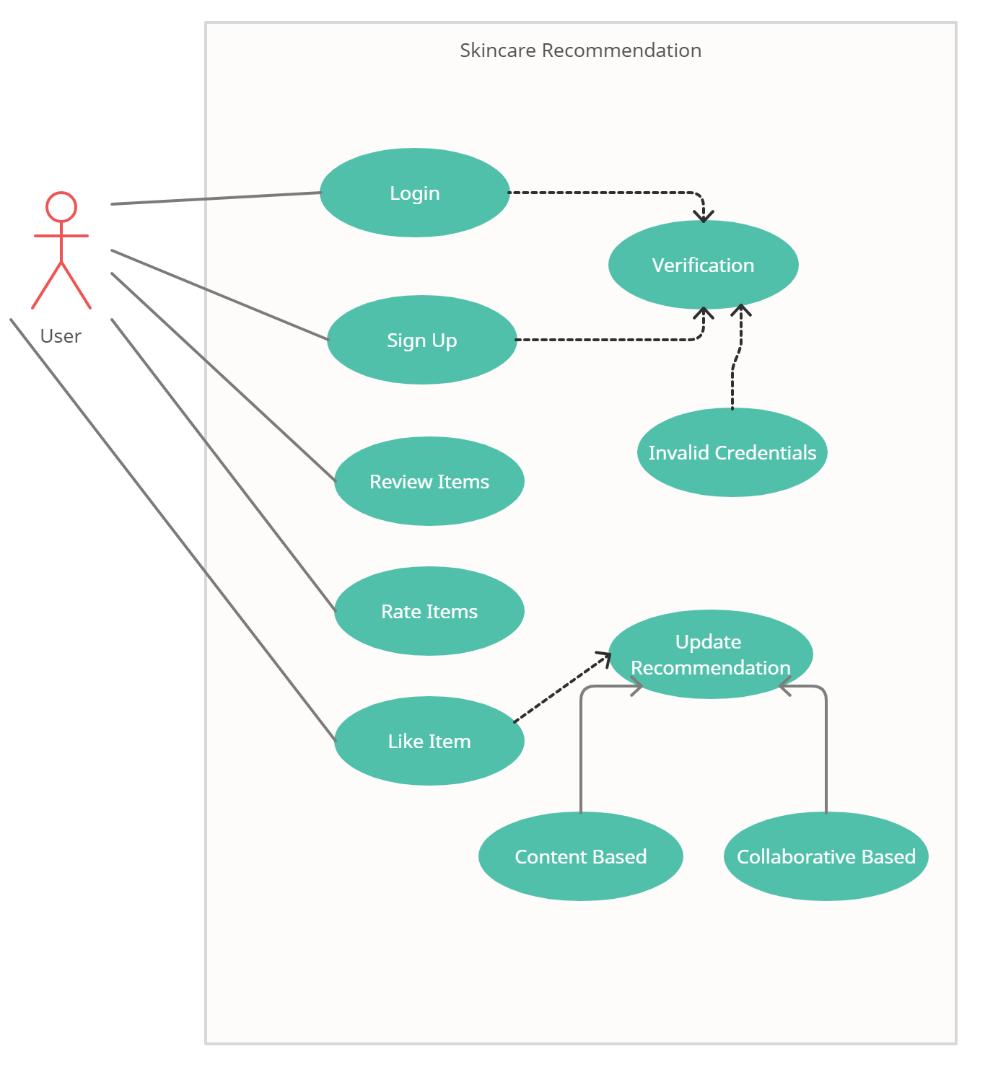


Figure 5.2 Use Case

### Front end Design

The application has been designed as an ***Android app*** to accompany the growing number of users and to provide easy accessibility for everyday use. Mobile machine learning or as it’s popularly known as ***Mobile ML*** is the forefront of modern-day machine learning, especially given the recent innovation in the field of 5G which fulfils the requirements for greater data transfer rates (ADERIBIGBE, 2020).

Popular technology stack for this domain consists of Java for android, Kotlin, Flutter and React Native for planning and executing the frontend part of the project. The major difference between these approaches is the programming language that these technologies represent and the cross-platform compatibility that they provide. While these did solve a bulk of problems, but what can’t be overlooked is the efficiency in these platforms. Applications that are developed in native framework outperform those with hybrid technology stack, this is mainly because the native code is executed as is in the application whereas, hybrid applications provide a wrapper around the code base which is in turn executed to be compiled as close as to native applications. Secondly, these hybrid or cross-platform technology stacks are still new to this paradigm and are in constant state of development where updates are still in progress and beta programs for new features are available for developers to enrol.

### Backend Design

Python is a ***dynamically typed language*** i.e. it provides the mean for the creation of a standalone server natively in to the application. A yearly survey held by JetBrains in collaboration with the Python Software Foundation PSF in 2020 outlines that 85% of data analysts and data scientists use Python as their main language (JetBrains, 2020). I have used ***Flask*** as the server for this project. It is micro service web framework making it light weight while providing full access to the developer.

Servers are a basic necessity for any web or mobile based application where information needs to be processed first and then displayed to the user. This is especially true for non-SPA (Single Page Applications). Such is the case in this scenario where a database needs to be accessed before routing information from one screen/page to another. Besides machine learning algorithms what Python also provides is web development frameworks like Flask and Django which are an entire solution in themselves for providing standalone server deployments.

Flask’s own server is only stable for debugging or development mode hence, whilst hosting the backend to a cloud platform such as Herokuthe server is redirected to a ***gunicorn*** multi-threaded server which can bare more load and is ideal for production-based deployments. ***Heroku*** is a cloud hosted platform that support frameworks like Python, Ruby on rails and Node.js servers natively utilizing the in-built AI to recognize technology stack and completely setup the requirements and multi-threaded servers by itself, making it ideal for this purpose. Python has been a choice for server-side development for quite some time now, YouTube being one of the major platforms to switch its entire technology stack to Python from PHP due to its flexibility, performance, and ease of use. Figure 5.3 describes the overall project architecture.

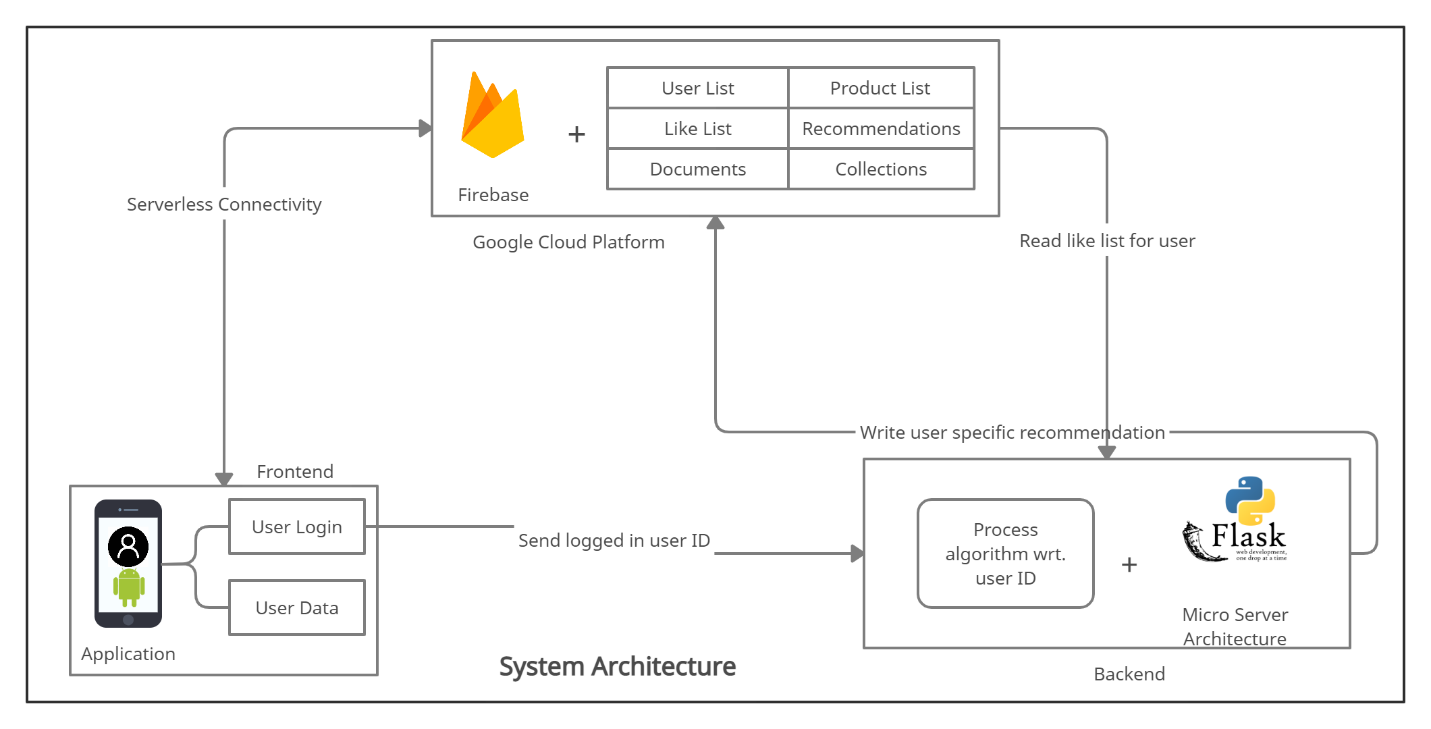


Figure 5.3 Project Architecture Overview

### APIs

In this project backend and frontend needs to be able to communicate with each other. To facilitate this communication a HTTP based API is selected to be developed in Python.

Application Programming Interface provides a sort of a middleware for the application to connect with the backend. These APIs receive information and are defined with the process of how to process them and return to their respective caller. APIs for our application have been implemented in Python. These are ***https*** based and when called from the frontend of the application returns the results in a ***JSON*** format using ***jsonify*** provided by the Flask framework. There are 2 major API call relative to the project, one for content-based recommendation, and one for collaborative based filtering. Along with this APIs provide a sense of ***abstraction*** between the two components of frontend and backend. What the user is required to see is available on their local storage while any or all of the process is done on the server with the help of API calls.

### Database Architecture

SQL databases have been a point of source for many production-based applications that can deal with enormous chunks of data, keeping the relation-based schema for tables in high regard. However, with the present paradigm of cloud-based software as a services ***SaaS*** many database applications have switched to pre-hosted ***NoSQL*** based database services such as Firebase and ***MongoDB***. As a comparison, NoSQL database have proved to be more viable and sustainable for content-based applications where instant delivery overrules the need of relations or foreign keys in a database. Plus, these databases come pre-hosted that provide an authentication of their own and providing an additional layer of security with keys and access shared specifically with those people who require it also making them highly scalable. These databases consist of documents and collections instead of table and columns because ***JSON*** is a natively supported format for these applications/services. Whilst considering the option between MongoDB or Firebase by Google the choice to conclude was ***Firebase*** since Firebase provides a real-time database connectivity option and multiple authentication routes. Firebase is also considered the ideal choice for mobile ML solutions for its seamless connectivity and data engineering solutions that keep data-oriented delivery in high regard providing a ***serverless connection*** to any mobile application which makes the application independent of a compulsory backend server.

Figure 5.4 describes the schema for the database in detail. There are 4 main collections that define the schema, these are users, brands, products, and recommendations. The user collection hosts further 2 more sub collection besides the document which contains the details for the user’ characteristics and login authentication details which are ratings and comments. The like list is embedded in to the user document using which the recommendation document is populated with the respective user ID key. All the details for the products are inside a document within the products collection, whereas the brands collection is solely for the purpose of displaying data on the brands home screen inside the app. Whenever a like list is updated the API is called which in turn populates the recommendation collection. The data for users is continuously updated whenever a user updates their profile and or a new user sign up with the application. This new data is fetched by the API and fed in to the continuous learning model for collaborative based filtering.

With a NoSQL architecture Firebase incorporates natively into an android application hence it was the ideal contender for this project. More and more applications are switching to NoSQL based databases that store data in the form of documents and collections providing a flexible schema. Firebase is hosted on the ***Google Cloud Platform*** which makes it highly scalable with zero to no down time.

The user details is the main collection which hosts the user details such as name, age, ID and characteristics. The user collection contains one or more like list documents which hold the product IDs for the item the user likes. The products collection contains all the products details including price, URL, brand, category and image url. Like list can contain a specific Product ID just once for a single user. The recommendation collection is divided in to 2 document which are collaborative and content based filtering. These documents can have each user ID just once.

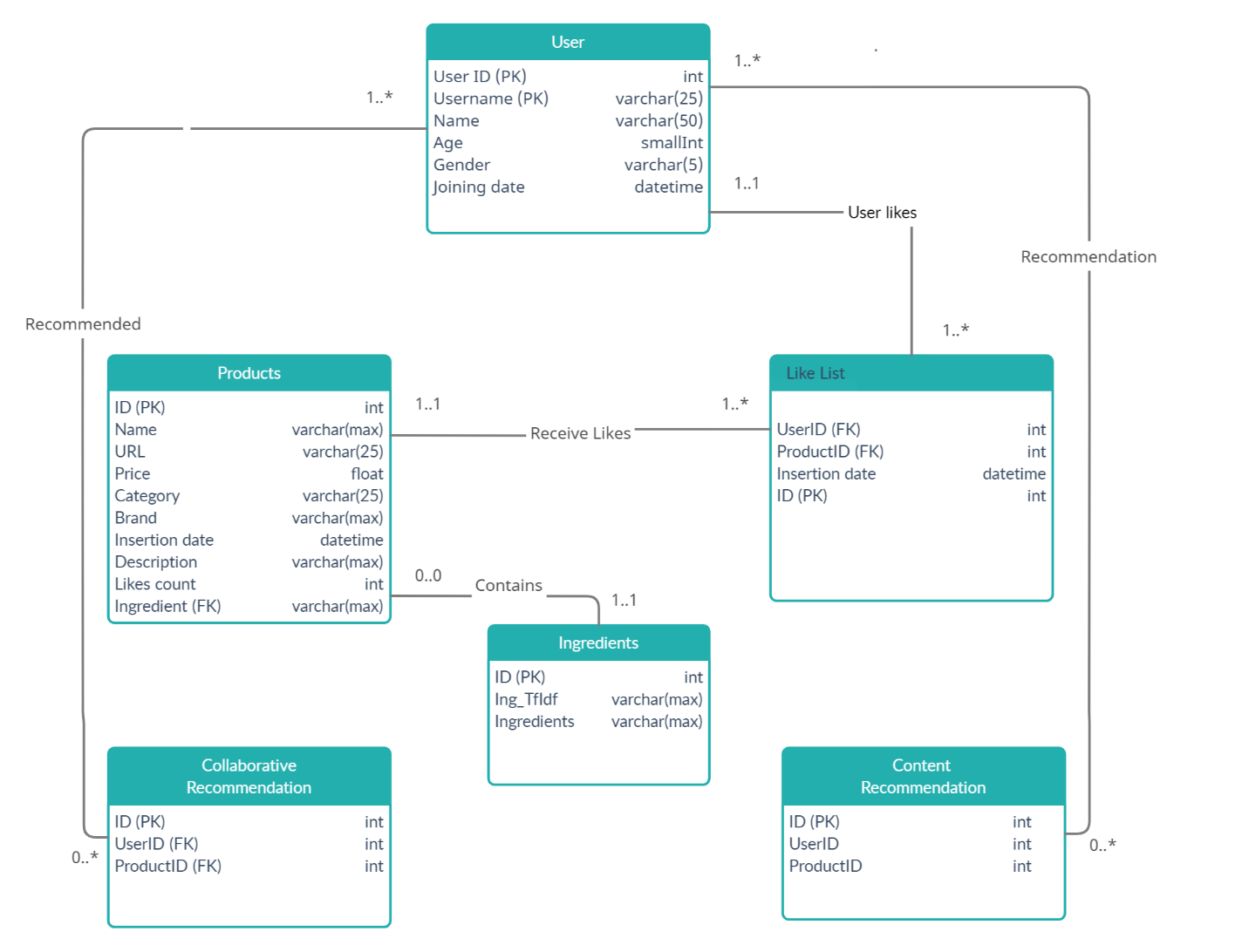


Figure 5.4 Database Schema

## Data Pre-processing

This section describes the design and format of the data that was to be processed by the machine learning models. Firstly, is was noticed that there wasn’t much data collected for skin care datasets. Whatever would be found was either incomplete or had missing features that were required for the model to process for the scope of this project. Hence data collection was a major part for this project which consisted of collecting dataset from website such as ***Kaggle***. However, just relying on the data from a single source meant it would give rise to an unbalanced dataset. Meaning there just wasn’t enough continuous data to learn from and the models would have a hard time catching a pattern in decision making. Even ***bagging*** and or ***boosting*** would not play an important role. Hence, to overcome this challenge websites were scrapped to achieve a better ***skewness curve*** in the data where multiple brands products had multiple reviews and user. Sufficient amount of data was scrapped from [look fantastic](https://www.lookfantastic.com/) website along with their user reviews, IDs, ratings, and product details. Scrapping data from a retailer rather than a specific brand gave the opportunity of diversifying the dataset for making prediction on a wide range of inputs and not just biased to a single brand and or retailer. The data was collected in an excel sheet format from a total of three sources. The data was collected ethically only capturing the information that is publicly available.

### Exploratory Data Analysis - EDA

EDA is an important aspect for implementing a smart solution. This gives us the sense of characteristics and any unusual outliers on a higher level. All of the statistics in a dataset can be observed here and corrected if deemed necessary for processing better results like slicing the dataset or duplicating it to highlight certain features that are important for the model to consider when learning from these datasets.

In the end the result came out to be a dataset containing products from around 191 different brands with a total of over 2600 unique products. About 13000 user reviews, from 9759 users who reviewed just over 1000 products. All of these users even had their skin type, skin tone, hair colour and other similar features also extracted from the website and if they were not present an NLP based algorithm was used to extract the features from their reviews with a standard accuracy.

### Feature Engineering

Data collection specifies the sources and format, but what the machine learning algorithm deems necessary are features which it is going to process. The raw data has about ***28 different columns or features*** (15 for the user reviews data sheet and 13 in the products list) that would overwhelm the model with unnecessary data and provide substantial results. The process of understanding the requirement for the model with providing it with specific features to learn from is what feature engineering highlights.

From a total of 12 features in the products list 3 have been deemed necessary for working with Content Based Filtering which include:

1. Brand
2. Category
3. Ingredients

These features are selected on the basis of what can be used to identify the relation between different products and what could be the desired effect for the user behind it.

Whereas, from the list of user reviews which was having 15 features out of which 12 have been utilized for Collaborative Based Filtering which implement multiple algorithms for driving out the results. These 12 features include:

1. Product ID
2. Rating
3. Reviews
4. Combination Skin
5. Dry Skin
6. Oily Skin
7. Normal Skin
8. Sensitive Skin
9. Skin Type
10. Skin Tone
11. Eye Colour
12. Hair Colour

These features provided Item-Item as well as User-User based relation. For a person with similar characteristics will want the same results as someone else while a product rated similar to another could also be rated in a similar manner by another user.

### Data Engineering

Not all the features and data organized can be put for processing. Much of it could be considered as ***noise*** by the algorithm. Data engineering is the process of bring the dataset in to a much organized and understandable form for both machine and human. ***SQL server*** was used for processing the data and identifying if there was a need to process it again. For example, if there was brand named slightly differently using a special character from one source and the character differed from the other source then they would be deemed as 2 different companies. Similarly, if one source had a unique product ID for one of the items and the case item would have another from the second or third source then this would render the data ambiguous.

For overcoming these challenges, the data was passed through a SQL server pipeline which filter the data and disregard any unwanted entry. This consisted of slicing the data where a user had reviews just a single product or a product was reviewed only once. Since would cause noise for the algorithm entries such as these were decided to be opted out from the original form of data which was to be processed.



Figure 5.5 SQL view for generating user reviews sheet

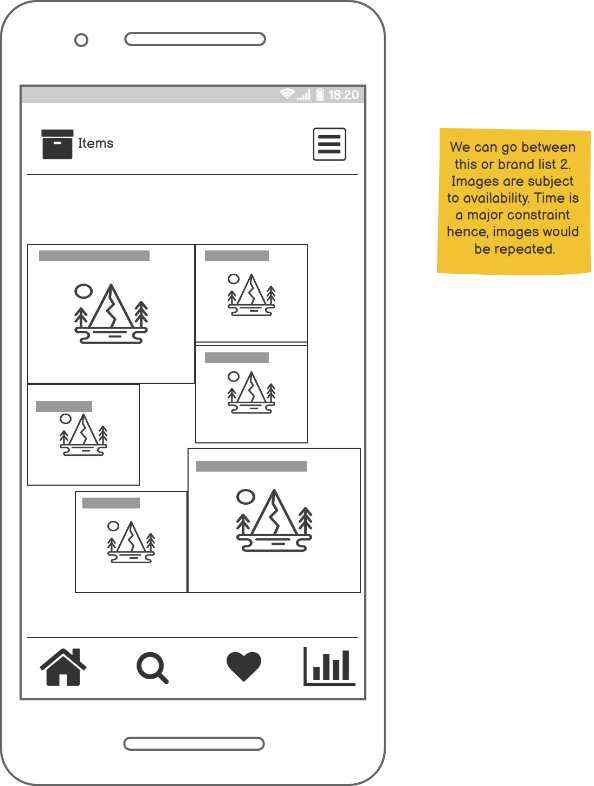


Figure 5.6 A part of the data pipeline to categorize similar data from multiple sources

## Application Wireframes

The user interface is always an essential perspective for any application. The application is to provide users with recommendation and usability of this application is an important aspect as if the users are not able to easily navigate throughout the app to find the information this application would fail to serve its purpose. This section concerns with the prototyping for the application with respect to wireframes and the user flow. The flow of the application is designed with user satisfaction and experience kept in mind. It follows the traditional concept of traversal, button and icons which keeps the learning curve to a minimum and requires a minimum demo for the user to learn how the application operates. It drives from the memory principle of human computer interaction making the application ***knowledge driven***. Figure 5.8 shows the user flow diagram for the entire project, design using ***Balsamiq Cloud*** on the basis of which following text defines the design of the application.

***Sign In*** screens hosts the usual email and password authentication form which validates the user’s credentials and on successful authentication redirects the user to the home screen of the application. This screen also has links for helping with forgotten passwords and sign up option for the user. Along with the application logo placed at the centre top.

After login a hamburger menu appears on the top right of the application allowing the user to navigate to their profile section or logout making traversing super accessible. Whereas the rest of the menu for the app will be visible as a horizontal traversing bar pinned at the bottom of the application. The Figure 5.7 next to the text here shows the home screen wireframe for reference.

***Home screen*** describes the brands available on the application, multiple stores for the user to explore and to their like list. Home screen is visible in a thumbnail view clicking on each will redirect the user to the brands list which shows them all the products from that specific brand.

***Sign Up*** and ***Profile screen*** are similar in ways that they both display the same form for filling up the user details. This maintains the theme for the application making user accustomed to the similar visual feels. This also helps them realize what’s on their screen with just a glimpse instead of reading through and understanding the form that is in front of them.

***Item description*** page describes the ingredients, description, rating and reviews of the product that the user had just clicked on. It also has a heart icon which can be tapped on to add/remove the item to/from their like list for getting similar recommendations. This feature has somewhat an Instagram flair to it helping the user realize its functionality without much explanation.

Figure 5.7 Home Screen Wireframe

***Like List*** screen can be visited by clicking on the heart shaped icon on the traversal bar. It displays all the liked item/products the user has reacted to. Any/All recommendation will be suggested on the basis of this list.

***Recommendation List*** can be visited by clicking on the graph icon. This screen hosts 2 lists, one for content-based filtering and another for collaborative based filtering respectively.

***Search screen*** is the final screen on the wireframe. This screen will present the user with a search bar to look up related item related to the text they enter there from the list of all the item in the database.

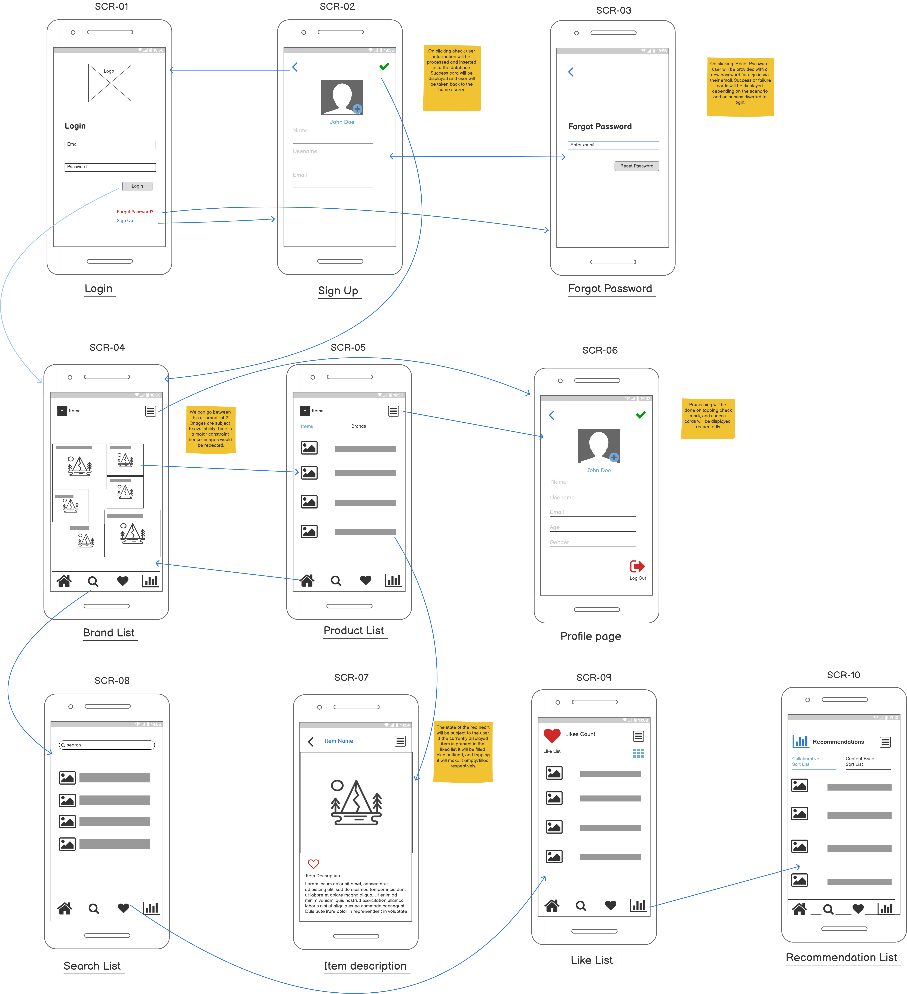


Figure 5.8 User Flow

## Machine Learning Models Design

There were a some of the machine learning models implemented for the prototype, out of which 3 have been selected for implementation. These all are based on ***model-based approach*** meaning they work on the principle of ***continuous learning***. Content based approach uses a *TF-IDF* vector approach to classify the similarity for a combined feature of brand, category, ingredients and recommend products based on the ***cosine similarity*** of the new feature. Collaborative filtering uses a hybrid of ***KNN clustering*** and ***Decision Trees*** for the purpose of recommending skincare products. KNN is used for item-item based recommendations whereas, decision trees output the results for user-user based approach. The data processed from the pipelines above feed the results here. Figure 5.9 and Figure 5.10 show the model design and flow for content based and collaborative based approach respectively. A sparse matrix has been utilized just as to keep only the needed information in the RAM in the form of a multi-dimensional array and discard any value that is not a number or does not contain a value. Since the data processing has been carried out prior to splitting them with a train/test split the architecture consists of 3 steps:

1. Data Collection
2. Model Processing
3. Recommendation

The data from the features discussed above an aligned with respect to model they’re being sent to, then the algorithm is processed for the recommendation to be populated into the database.

### Content Based Model Flow

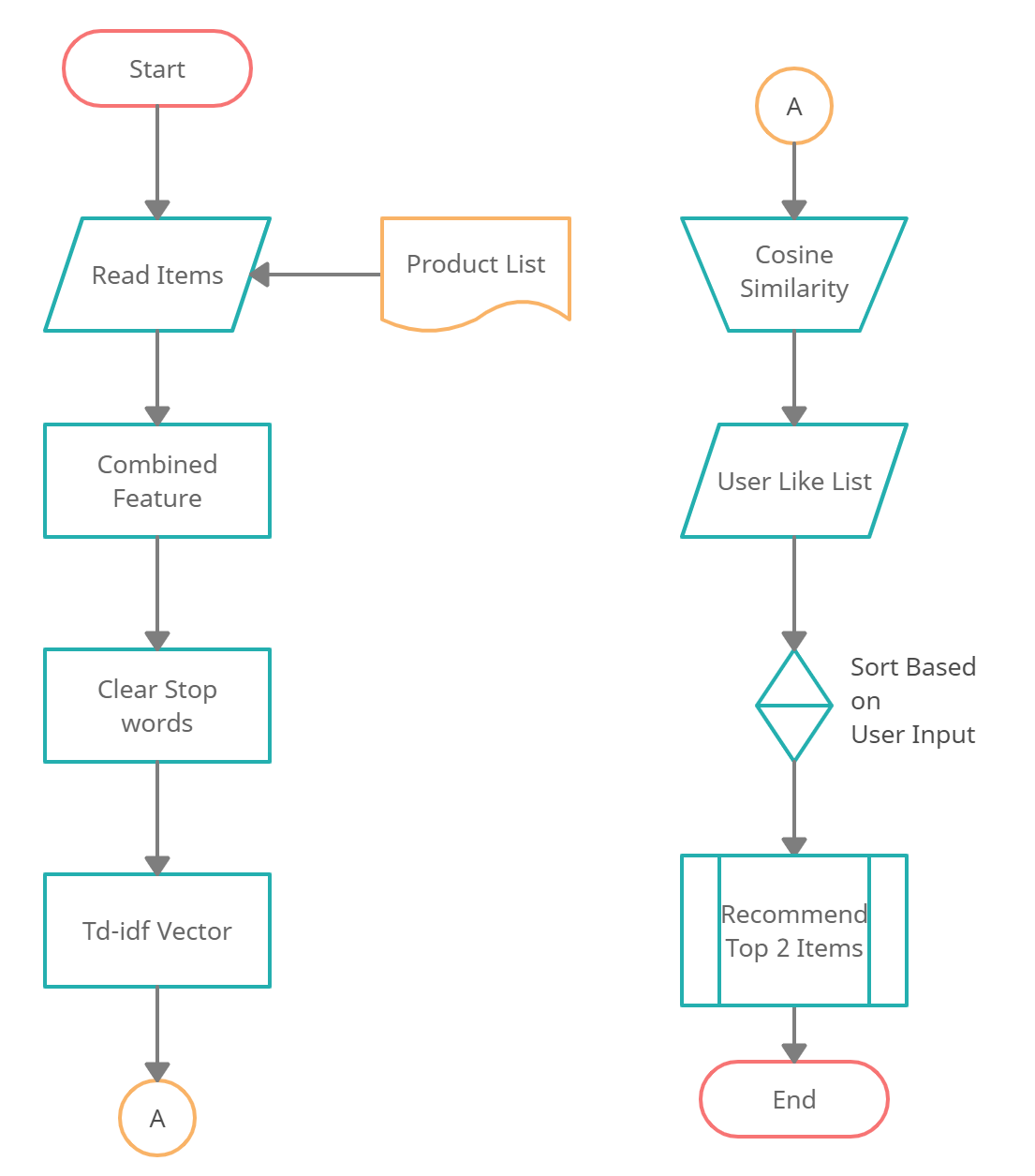


Figure 5.9 Content Based Model Flow

### Collaborative Based Model Flow

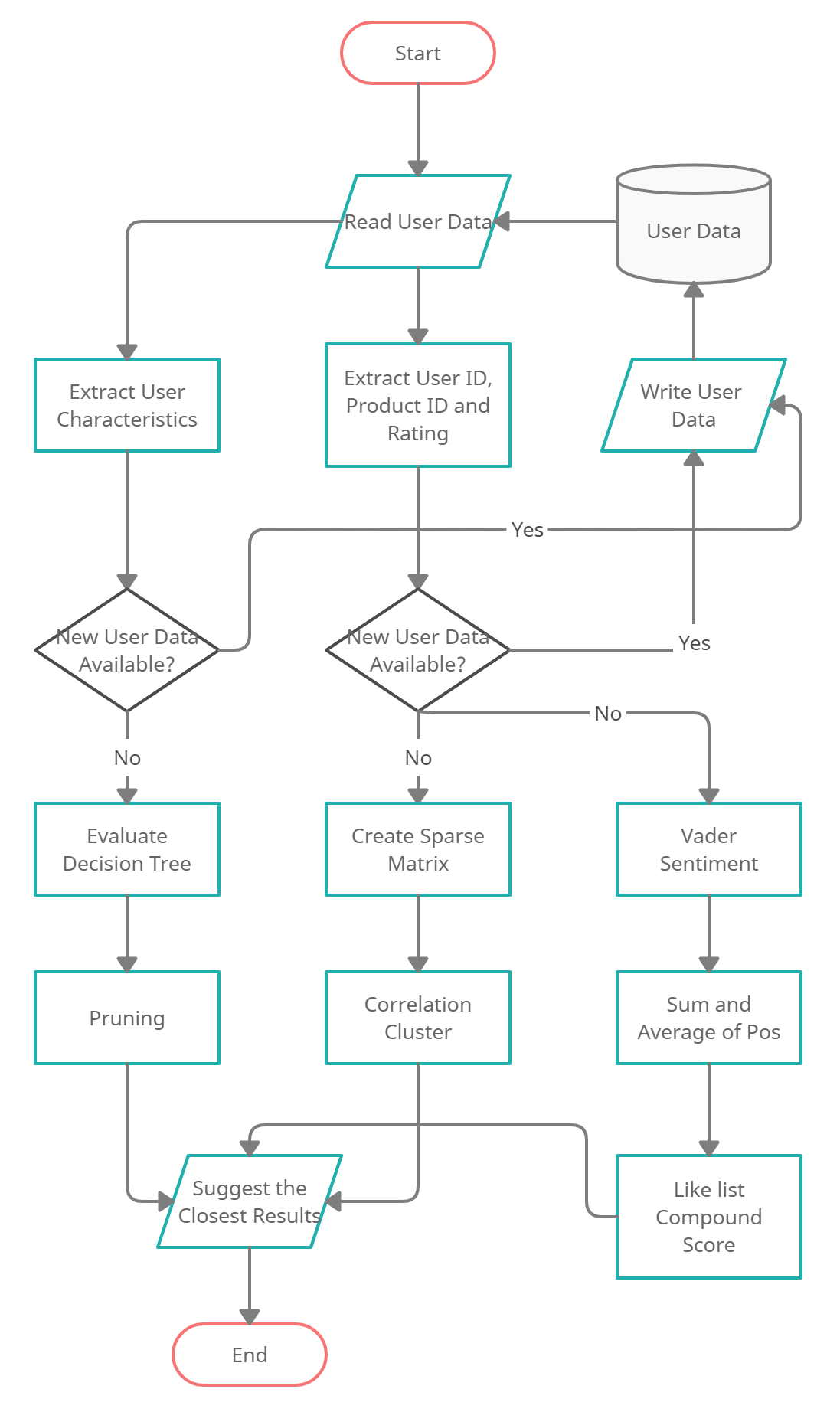


Figure 5.10 Collaborative Based Model Flow

# Implementation

Implementation section defines the interaction between all of the components and how data is processed in to a meaningful format to display to the user. Following the 3-layer architecture as discussed in the design section, Figure 6.1 Implementation Workflow shows the implementation and workflow of the application. User interaction is recorded in the database rather than being communicated to the backend first. Thanks to the real-time serverless connection just the updates that need retrieving are fetched via the flask server. On logging in to the application the recommendation is fetched from the server for the specific user. Similarly, on adding and or removing something from the like list will fire up an API that will fill up the recommendation for the signed in user. All the necessary processing and data fetching is done between these API calls. The server retrieves the data and sends it to the 2 functions i.e. content based filtering and collaborative based filtering respectively. These 2 API calls based on their specific features which were discussed in section 5.5.2 Feature Engineering execute the models which then generate the recommendations of products in the form of product IDs and send it back to Firebase in the form of an integer array/list. The data for content-based recommendation is processed through a ***Tf-idf*** vector and their cosine similarity whereas, collaborative based recommendations generate results with the help of a sparse matrix that calculates the ***correlation*** between the ratings of products by the user and return similarly rated products. The ***decision tree*** on the other hand takes in to account the users and their unique characteristics depending on which products are recommended to them. The details for the models, the data that they process, and the operations that are carried out have been discussed in the text that follows.

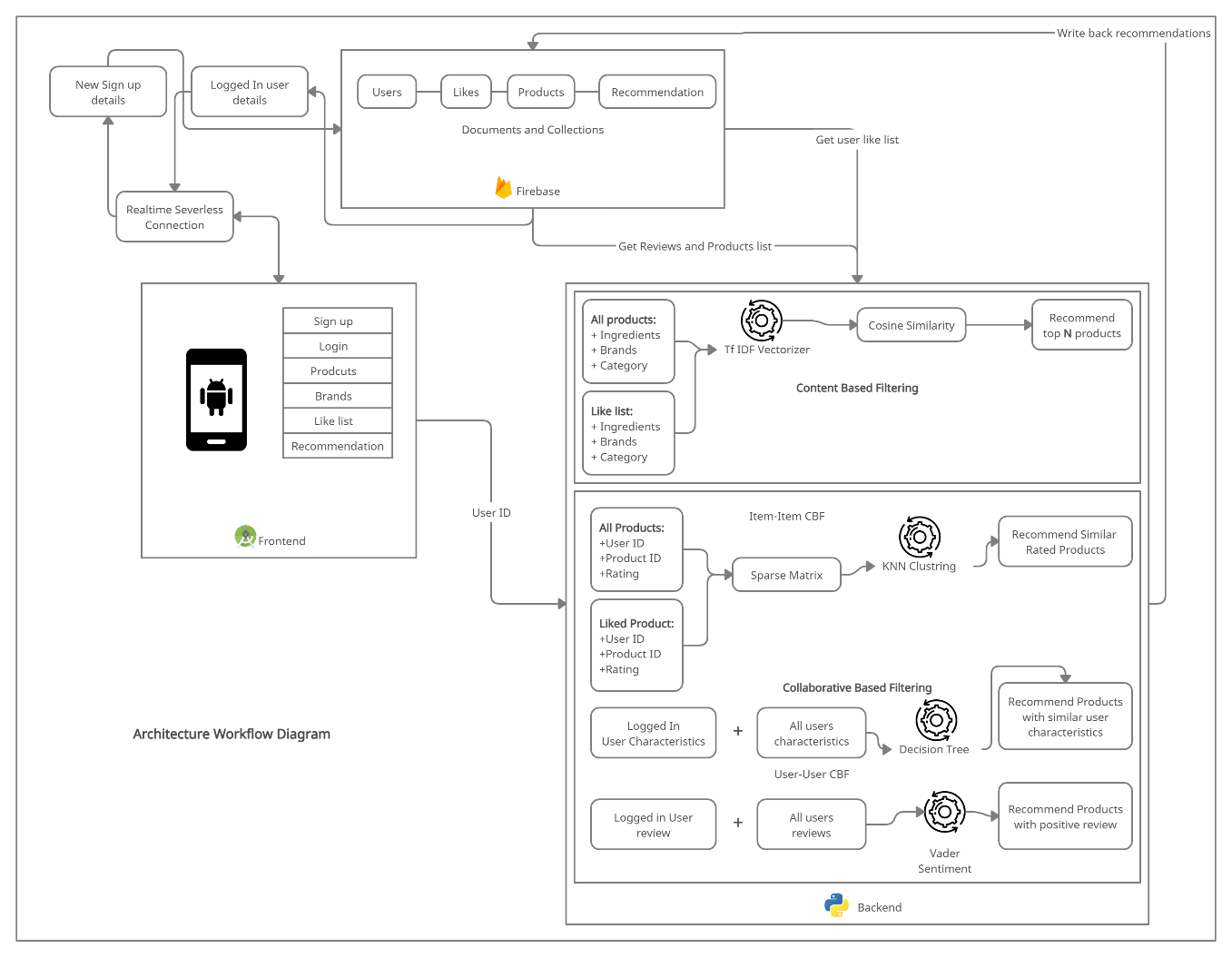


Figure 6.1 Implementation Workflow

## Backend and API

### Data Processing

Given that not much was available for processing as far as skincare data is concerned, data processing was a big part of the project. Following the design of the data discussed in section Data Pre-processing the data obtained from the source is unbalanced and represents ***skewness*** for multiple product IDs. This was overcome by duplicating the data, this makes the pattern occur more often and makes the algorithm recognize changes more accurately. Doing this also helps with ***stochasticity*** in the data, and any random occurrence of rows can be explained more logically now. Typically, any structured dataset includes multiple columns, a combination of numerical as well as categorical variables. Hence, another step that occurs in the processing part is ***label encoding*** where textual values are assigned a unique integer values which makes it easier for the algorithm to process. ***Scikit*** ***learn*** in Python provides a way of doing so based on alphabetical orders of the characteristics/features. Another form of doing this is called ***One hot encoding*** which produces binary vector of values so that the algorithm doesn’t process 2 characteristics as a dependency on one another, but One-Hot Encoding results in a ***Dummy Variable Trap*** as the outcome of one variable can easily be predicted with the help of the remaining variables. The Dummy Variable Trap leads to the problem known as ***multicollinearity*** hence I go forward with label encoding in this scenario. Besides this, the number of characteristics in our case is enough to avoid the reason for implementing one hot coding. Figure 6.2 shows processing of the user reviews that require their comments to be ***tokenized*** for the purpose of natural language processing, given this a separate column was created called ‘reviews cleaned’. The data was also cleaned by extracting a new data frame from the original with just the required features with a ***train test split*** of 70 and 30 which is usually the ideal amount for scenarios such as this. ***Cross validation*** was also applied to these algorithms with 5 folds.



Figure 6.2 Tokenizing Reviews

These scripts were first prepared with a prototype of the project using ***Jupyter Notebooks***. These required connection with the firebase admin, nltk corpus for python and the SQL server database. Important imports for this purpose can be observed below.



Figure 6.3 Important imports

#### Scrapping Data

As discussed, the scrapping part of the project was an important aspect. This concerned with using the link of the websites provided in the dataset to collect further information such as stars, reviews, username, and title for more products. Figure 6.4 below shows a part of the code that was implemented for this purpose.



Figure 6.4 Python Scrappy Framework Script

### Database Connection

Firestore database was created to get the API token to connect to the database. This token provides an Auth key to set up connection with the desired application as can be observed in Figure 6.5. In this scenario there were 2 application set up for access to the serverless architecture, the frontend application and the backend API. Figure 6.3 shows the firebase admin as one of the important imports for setting up the application.

Before integrating the database to the main server code base the data collected from the data engineering process carried out in section 5.3.3 needed to be inserted to the Firestore real-time database which was to be displayed to the user on login. Along with this a few users were to be created in the database for testing and verification. For this purpose, a script was created which uploaded all the 2000+ brands and 12000+ user reviews to the database.



Figure 6.5 Database Connectivity

Each time a new user is created the collaborative function fetches the new information from the database to incorporate the changes into the model for continuous learning. Figure 6.6 shows the query to fetch the new user data. The user IDs which have been specifically removed were integrated for testing purposes and have been discarded.



Figure 6.6 Fetching user data

### Routes and Server

The application takes in to consideration 2 routes one for content based and one for collaborative based filtering respectively. The input parameters, return statement and design are somewhat similar except the model and algorithm that are being called with.

Data frames and variables have been declared global as can be observed in Figure 6.8, to make them available in the application throughout. These are the data frames that will be processed by the algorithms.



Figure 6.7 Global Variable Declaration

Along with some functions that are required for the process, such as the item and recommendation function as seen in Figure 6.9 that log and print user recommendations to the console log and also return the specific product ID to the API.

The item function is used to get the item name from the data frame for logging purposes, whilst the recommendation function is set to return the top N stated recommendation product IDs to the API and for the logs.



Figure 6.8 Global Functions Declaration

Figure 6.10 describes the recommendation writing function from the content-based perspective, it returns the top best suggestion from the recommendation matrix generated by the model’s algorithm.



Figure 6.9 Collaborative Recommendation Writing Function

The server is executed when it encounters Flask’s main function. Figure 6.11 shows the application named app being executed by the Flask server. By default, the debug server is started at port 5000 however, this can be adjusted.



Figure 6.10 Server Start up

### API Implementation

API or Application Programming Interface provides a point of communication for the server with the frontend of the application. Any information is sent or received through here. All of the functionality and models have been implemented in these routes. Figure 6.12 shows the route implementation for the collaborative based filtering.



Figure 6.11 Collaborative Filtering Route

## Machine Learning Model

### Content Based Filtering

Content based filtering refers to making recommendations or suggestions on the basis of the characteristics of the product itself regardless of the user. This is the most common approach used throughout in recommendation engines. Content based filtering in this approach included multiple algorithms that were implemented and tested.

### Tf-idf Vectorization

Tf-idf vector work on the basis of calculating similarity between occurrences of a word throughout documents. Tf-idf now is the right measure to evaluate how important a word is to a document in a collection or corpus.

Equation 1 Tf-idf Vectorization

Tf – is frequency counter for a term t in document d

Df – is the count of occurrences of term t in the document

Idf – is the inverse of the document frequency which measures the informativeness of term t

Certain terms, such as “is”, “of”, and “that” or any other stop-words the like may appear a lot of times but have little importance. Hence, to counter this I need to weigh down the frequent term while scale up the rare ones. This is achieved by IDF aka ***Inverse Document Frequency***. It diminishes the weight of the terms that occur too frequently and increase the weight of the terms that occur rarely in throughout the document set the data for the Tf-idf vector balanced. Figure 6.13 shows the Tf-idf vector generation.



Figure 6.12 Tf-idf Vectorization

Now as I have all the product names in the form of a vector based on their occurrence I can use a measuring index to find out the similarity between what the user has entered/liked and suggest the top matching to them. For this I’ll be using ***cosine similarity*** index. It is a measure of similarity between two non-zero vectors. Non-zero, hence making it ideal for a sparse matrix where zero values are ignored. The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

Equation 2 Cosine Similarity

A combined feature of brand, ingredients and product category is passed through the Tf-idf vectorization model and the top 2 recommendations with the highest results for the liked list items are populated into the recommendations list for that specific user. Figure 6.14 below is the sample code that represents the implementation for cosine similarity between each of the entries after which a recommendation function is used to return the closest matching value from the matrix.



Figure 6.13 Tf-idf Matrix and Cosine Similarity

### Collaborative Based Filtering

However, how relatable content based filtering may seem this can cause rather ambiguous suggestions since not always will be a correlation between the user and the product as trends are not monitored by this approach and if for example a user used to watch horror movies and now seems more interested in drama scripts then anyone who is being recommended what this user like or dislikes not necessarily follows the same trend, this is where collaborative filtering comes in.

#### Correlation Clustering (Item-Item based CBF)

Clustering the process of grouping liked values together. In the process of ordering values with respect to the correlation between them on the basis of liked items, a matrix was prepared that had user ID on the y-axis and product ID on the x-axis. The user entered value column is extracted from the matrix and correlation is calculated with any other user in the list. Any other item with a similar rating is suggested to the user. To make the training process more viable ***cross validation*** was implemented for catching all the product IDs in the training sets. Exactly for this reason users who had bought only a single item and products that were reviews only once were dropped. This type of filtering is ***item based*** since there is no relation whatsoever with user characteristics. Figure 6.14 shows the implementation for training and testing the KNN model.



Figure 6.14 KNN Training and Testing

After the best fit for the value of K i.e. the number of items in a cluster has been identified it can be used to predict on the basis of the new value as shown below.



Figure 6.15 KNN Predictions

#### Decision Tree (User-User based CBF)

Decision trees and simple probabilistic models that compare the change of occurrence of each scenario with every outcome in a tree layout structure. There are many types of decision trees and multiple methodologies available. In this project I have followed a decision tree with ***gini index*** and ***pruning***. Pruning is a data compression technique in machine learning and search algorithms that reduces the size of decision trees by removing sections of the tree that are non-critical and redundant to classify instances. That is, it reduces the complexity of the algorithm whilst making better prediction results. The Gini Index varies between 0 and 1, where 0 represents purity of the classification and 1 denotes random distribution of elements among various classes. A Gini Index of 0.5 shows that there is equal distribution of elements across some classes.

Equation 3 Gini Index

The model is trained to classify product IDs based on user characteristics. It takes into the account the binary encoded values of hair colour, skin tone, skin type. as the X values and the product IDs that those respective users have bought based on their characteristics as the Y values. The results obtained were a mixed bag of values rather than perfectly accurate since the data collected is not so continuous to help the algorithm generate the right structure. Ensemble technique was also used in this case to try to overcome the lack of continuity of the data where a few users had only bought a handful of items. ***Bagging*** was introduced in this case by duplicating the data with a ***K-fold cross validation*** of 5.

The implementation of the decision tree itself was divided into different function calls which provides better re-usability. Figure 6.17 shows the split dataset function which when passed with the no of column (5 in case of collaborative) for the matrix return with a split of X and Y respectively. The second function return the hit rate of the algorithm while the third function is for training the decision tree based on the Gini index.



Figure 6.16 Decision Tree Training, Testing and Split



Figure 6.17 Decision Tree Prediction

### Natural Language Processing

Making conclusions is an extensive part of human analysis. Natural Language Processing or ***NLP*** is the study of making sematic analysis from textual inputs in the field of computer science. Python introduces 2 methods of making NLP analysis from within the library held with the ***PyPI*** (Python Package Index), *Textblob and Vader Sentiment analysis*.

In this approach I have utilized the positivity score from the Vader sentiment analyser. User reviews have been summed on the basis of the product ID and their average recorded into a data frame. Figure 6.16 shows a sample of the prepared product ID and their respective user review scores. Vader suits better to our needs it has a better library built for semantically analysing an E-commerce website reviews and recommending user products with similar reviews.

#### **Vader Sentiment Analysis**

***Vader*** uses a list of lexical features (e.g. word) which are labelled as positive or negative according to their semantic orientation to calculate the text sentiment. Vader sentiment returns the probability of a given input sentence to be positive, negative, and neutral. It is optimized for social media data and can yield good results when used with data from Twitter, Facebook, etc. Vader sentiment analyser returns 4 values to summarize the contents of the input sentence these are

* Positive ratio in percentage format ranging from 0 to 1
* Negative ratio in percentage format ranging from 0 to 1
* Neutral ratio in percentage format ranging from 0 to 1
* Compound that lies between [-1,1] just like polarity in Textblob

The sum of positive, neutral, and negative will sum to 100%



Figure 6.18 Product Review Scores

These scores are generated on run time hence making this a ***model-based approach*** rather the usual memory-based approach. Figure 6.17 highlight a bit of the code that was used to generate product recommendations based on an item-item collaborative filtering. The most similar score is then return from the list of which is what the next product the user might be interested in.



Figure 6.19 Vader Sentiment Analysis

Predicting based on the user review:



Figure 6.20 Prediction

## Frontend – Development

Frontend of the application was developed in Java with the wireframes as the base concept of the application. However, there were many changes that occurred during the implementation process, these consisted of visual changes as well as functionality changes which later either became vague or out of scope. The following section of the document shows the step by step process opted to achieve the completion of the application as design.

### Sign in Screen

The layout of the application started out with the sign in screen. A pink theme has been implemented throughout. User email and password is requested upon login and if it does not exist then the user can choose to create a new login password. By clicking on the sign-up button, the user is taken to the sign-up screen. The code snippet shown below gives an overlook of the implementation of firebase authentication for user sign in.

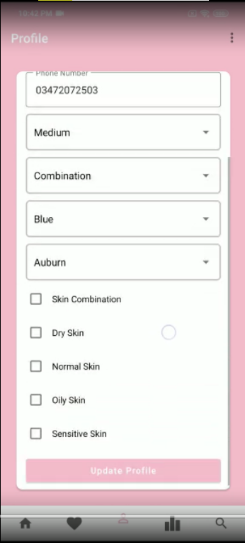


Figure 6.21 Firebase Connection and Authentication

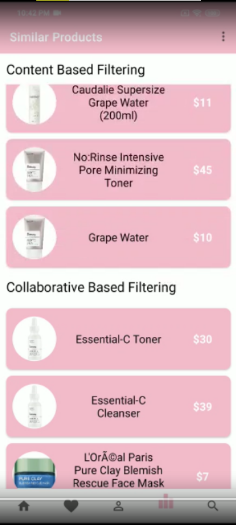
### Like List

The like list has been implemented as originally planned out. The list has all the item that the user has reacted to. The product IDs on this screen are what used to generate suggestion for the currently signed in user.

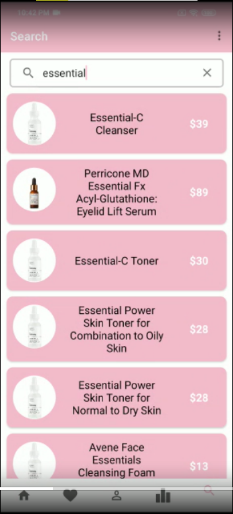
### Profile Screen

Instead of the original idea of having a hamburger menu for user profile change the three-dot menu on the top right shows the choice or logging out of the application and the user profile section has been moved to the bottom traversing bar. The profile update form allows user to update their credentials as well as the characteristics they originally entered whilst signing up for the application.

### Recommendation List

Following the similar from the wireframe, the recommendation screen hosts 2 tabs, one for content based and collaborative suggestions except this time they are vertically aligned instead of horizontally. The two tabs can be scrolled independently. While loading this screen the user might observe some delay due to the reason that product IDs to be fetched here are being generated by the model at the back via the API call.

### Search Screen

The search is where the user can look up from the extensive list of products a specific item that matches the words entered into the search/input bar at the top of the screen. Search screen has been kept a bit simple for an easy implementation since it only limits the search to products list and not brands list. Allowing brands list to be traversable would require a new schema to be prepared for the firebase database.

### Item Description

The item description page has been limited than what was discussed in the wireframe section. This page now displays only the current user entered comments and rating instead of the overall feedback. This restriction was introduced to reduce the complexity of the frontend of the application. Since requiring all the feedback would have to loop through multiple documents and collections. The description of many of the product may seem incomplete this is due to the scrapping process carried out which captured only partial text from the paragraphs entered on the website. Here the user can add the item to their like list, review the product, and rate the item from a total of five stars. Whenever data is updated here the models are executed to adjust to the dynamic nature of the application where continuous learning is the key.

# Testing

This section, describes the procedures that were followed when testing this application. This project uses ***unit testing*** for singular components and ***white box*** testing approach as a whole software approach mainly due to time constraints and to benefit from the fact having the code base ready for testing purposes without any other dependencies.

## Automated Testing & Manual Testing

On a higher level there are 2 types of testing that can be carried out for a project. They are automated testing and manual testing. Manual testing regards with the developers or the alpha testers going around with the code and working with the application that has been prepared up till then. It is done by clicking on and interacting with the features and the software APIs. This may be expensive and time consuming, but requires less time to set up the requirements or to set up an automatics tester agent/script to emulate a human going around the process. Automated testing is done via some sort of tool that defines the test cases pre-hand with a script. These require setting an environment and usually prepared for complex test and vast test cases that would ordinarily take humans a longer time to test the application. It is much more robust and reliable, but setting it up for a project that requires less complexity would be an overkill and make lengthen the work which was planned to be done quicker. Automated testing is usually setup for DevOps and similar procedures. Hence, for the sake of my application I’ve opted manual testing which is explained further in the following section.

## Testing Methodologies

Some other testing approaches include integration testing, black box testing and Beta/Acceptance testing. Integration testing is the approach of testing the different integrated parts of the application and see how they fare whilst communicating one another via an API. Integration testing is time consuming and requires an abundant list of test cases that require great understanding for the developed application. Black box testing is the process of overseeing the application from the perspective of someone who doesn’t have much information regarding the development process and procedure of the application. It is usually carried out by the QA team who testing structures laid out and can interact with the application at a higher level rather than debugging the code base. Beta/Acceptance testing is the testing part carried out by the users of the system, so is A/B testing. This type of testing usually takes long time to be implemented and the results need to be analysed first before completion.

## Unit Testing

Unit testing is one of the most basic and fundamental form of testing where individual/single components are tested for any particular software. The main goal of this type of testing is to involve and validate each and every unit of software program. Unit in a software can be considered as the smallest part of any software that is testable.

## White Box Testing

Clear Box testing or white box testing as it is generally known is a software test approach entirely based on developers working on their way through the program. The main function of this testing is to test the internal working or implementation of any program or software. High level programming skills are required to conduct this type of test.

Table 7.1 Unit Testing Review

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Description | Test Data | Expected Result | Actual Result | Details of Fix |
| 1 | User can review items | Input string in the review section | User can enter reviews correctly | Pass |  |
| 2 | User can like the products | Liked items will be added to like list | Like data will be present in the like list section | Pass |  |
| Items can be removed from the like list | Items will be removed from the list | Fail | Option to unlike items has been incorporated |
| 3 | Content Based Filtering | Tf-idf vector and Content filtering | Recommendation screen will show content-based recommendation | Pass |  |
| 4 | Collaborative Based Filtering | All the algorithms execute up to par | Collaborative Based recommendations will be displayed | Pass |  |
| 5 | Items URL link | Clicking on a button will lead to the products webpage | A browser page will be opened to display the original product and its description | Fail | URL links have been utilized now to redirect the user |
| 6 | Product Search | Enter product name | Entered product should be listed as the search result | Pass |  |
| 7 | Profile Update | User characteristics detail | The newly entered user characteristics should be updated in the database | Pass |  |
| 8 | Review Items | Add or remove user reviews | The added user review should be publicly visible | Pass |  |
| Added user review can be deleted | Fail | Review deletion feature wasn’t added |
| 9 | Collaborative based filtering update | User with a lengthy like list | User with 5 or more items in the like list should update the CBF results on adding another | Fail | The free tier server for Heroku times out. It needs to be upgraded |
| 10 | Item rating | Users can rate items | Add rating | Pass |  |
| Remove rating | Fail | Button added |

## Model Testing

Model testing have been carried out using Mean Absolute Error, hit rate calculation, precision and recall, cross validation and cosine similarity.

***MAPE*** has long been used as the standard loss function for regression problems. It explains the concept of error in a prediction with ease of understanding. Expressing the accuracy as a ratio it is defined as:

Equation 4 Mean Absolute Percentage Error

Where At and Ft represent the actual and forecasted value respectively. Multiplying the results by a 100 gives the percentage error. Problems can occur in MAPE when the actual values reach zero which produce undefined results or infinity, as an alternative each actual value At of the series can be replaced by the average of all actual values A−t.

In multilabel classification, ***hit rate*** computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in the test set Y.

Equation 5 Hit Ratio

***Precision*** is calculated as the number of true positives divided by the total number of true positives and false positives.

Equation 6 Precision

***Recall*** is calculated as the number of true positives divided by the total number of true positives and false negatives.

Equation 7 Recall

***F1 score*** was also the part of the measure along with Precision and recall.

Equation 8 F1 Measure

# Analysis and Evaluation of Algorithms

## Introduction

Recommendation System need to be evaluated before being used for production environments and making real world recommendations. This provides a testing ground for tweaking and verifying the working of the algorithm for providing better results and suggestion whilst also providing a firm standing for the future research. These systems are evaluated on the basis of how close recommendation they make based on the items liked by the respective users. They have a varied range of measurement since not only do exact matches or recommendation count as a viable hit, but also any similar items within the criteria are what counts. This provides a different criterion for evaluating the accuracy for a recommendation engine. ***Trust*** in RS is what defines the measuring criteria. Trust can be defined as the correlation between similar preferences towards the item that are commonly rated or liked by two users (Shambour, 2012).

Recommendation engines can be evaluated using not just accuracy, but also dependent on coverage (Shambour, 2012). Metrics type depend on the type of filtering technique used. This application utilizes both contents based and collaborative based filtering. Accuracy is just the fraction of accepted recommendation out of the complete possible recommendation of the coverage. I have divided the metrics in to statistical and decision support for further supporting my work. The suitability of each metric depends on the features of the dataset and the type of tasks that the recommender system will do (Friedman, 1997).

This section of the literature describes the evaluation criterion that was carried out to measure the results obtained for both approaches which have been discussed in detail in section 6 Implementation. These approaches utilize Machine Learning based algorithms such as KNN and Decision trees plus Natural Language process techniques like Tf-idf vectorization and Vader Sentiment Analysis. In accordance with this the relation between these algorithms and the data set prepared have been discussed as well.

## Experimental Setup

### Datasets

The dataset prepared in section 5.3 Data Pre-processing is what has been developed further for testing and evaluation purposes. The dataset in the product list consists of product ID, price, rating, good reviews and bad reviews. User reviews data contains user ID, product ID, rating stars and user skin characteristics. The data frame loaded in to the program is split in to a 70-30 ratio for training and testing purposes respectively. (Lee, 2020) uses a similar dataset to evaluate the quality of the algorithm, however it lacks is variance and continuity in the dataset since it only contains product from Sephora. The models trained when presented with any data from outside the domain will present a rather ambiguous suggestion for the user. Figure 8.1 below describes the summary of the dataset for the products list I have obtained along with Figure 8.2 which describes the same features for the user reviews dataset.



Figure 8.1 Product List Dataset Summary



Figure 8.2 User Review Dataset Summary

### Methodology

For measuring the results for the content and collaborative based filtering algorithms I have used mainly 2 quantitative indexes. The dataset split in the previous section is responsible for generating measuring results when passed through evaluation criterions discussed in the text that follows.

Firstly, the evaluation discussed is using the Mean Absolute Percentage Error ***MAPE*** Equation 4 Mean Absolute Percentage Error for both the KNN cluster and Decision tree. On the other hand, there is the ***Confusion Matrix*** which transcribes the precision, recall and the F1 score Equation 6 and Equation 7 respectively. MAPE is a measure of how accurate a forecast system is. It measures this accuracy as a percentage along with Precision and Recall that can help with measuring the actual and false positive from the predictions. Since, the language-based models such as the Tf-idf vector and Vader Sentiment cannot be measured using quantitative analysis they have been analysed with a beyond accuracy measure utilizing the ***hit ratio*** Equation 5 and ***cosine similarity*** index Equation 2.

## Results

### Prediction Accuracy

Figure 8.3 KNN describes the varying value of K for clustering the similar groups of item together. This has been calculated on the basis of hit ratio for CV value of 10. It was observed that the best value achieved was of 0.54 for a value of 3 for K, meaning that the ideal number of products that should be entertained are 3 items in each cluster. It should be noted that the value of MAPE ranges between 0-1 where 1 is the most erroneous value and 0 is the best value. Since, I have used the cross-validation score for KNN implementation, it converts the value of this measure to -1 to 0. Figure 8.4 shows the similar results obtained from a decision tress with pruning. The MAPE obtained was of 0.79 with a bagging classifier of 50 trees. This implies that the dataset was missing some clarity for the algorithm to be more decisive and could be improved upon for making better predictions. However, KNN cluster is uphold greater here due to its better execution time and slightly better accuracy.

|  |  |
| --- | --- |
| Figure 8.3 KNN | Figure 8.4 Decision Tree |

The precision and recall along with the F1 score for the KNN algorithm can be observed in Figure 8.5.

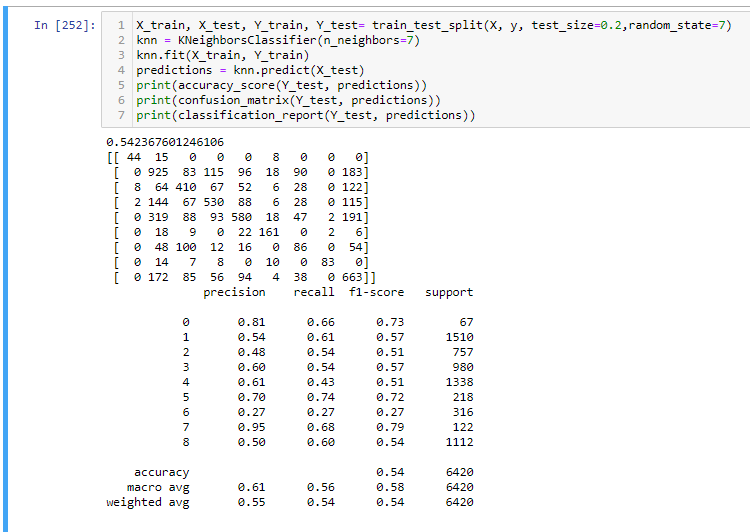


Figure 8.5 Confusion Matrix for KNN

### Top-N Accuracy

Measurements in RS is not just about the numbers obtained, but also keeps in regard the coverage of the prediction. Top n rated products could be the part of the prediction not the just considering the first value the algorithm suggests. Since the cluster advises the top 3 values to be recommended I’ve gone along with the top 2 since there will also be other algorithms that would be suggesting products.

### Beyond Accuracy

A user ID was used to verify the recommendations that were made. This required a user logging in to the system and adding products to their like list which would then in turn generate recommendation using that product ID. It was noticed that the suggestion made were closely related to the liked item. Besides this, Hit rate for the algorithm were also measured which shows that both of the collaborative models have a rate of more than half as can be observed in Figure 8.6 and Figure 8.7.

|  |  |
| --- | --- |
| Figure 8.6 KNN Hit Rate | Figure 8.7 Decision Tree Hit Rate |

Figure 8.8 shows the results from the perspective of a user having product 424 in their like list. It can be observed that values with similar ingredients have a similar name and hence a higher similarity index. However, since the data is not continuous and not many items have the similar description hence, I can see the results declining almost instantaneously after the first suggestion. It was observed that the first recommendation had a range of cosine index between 60 to 90 percent. It is also observed the recommendations are not limited to one brand and products from other brands are also recommended.

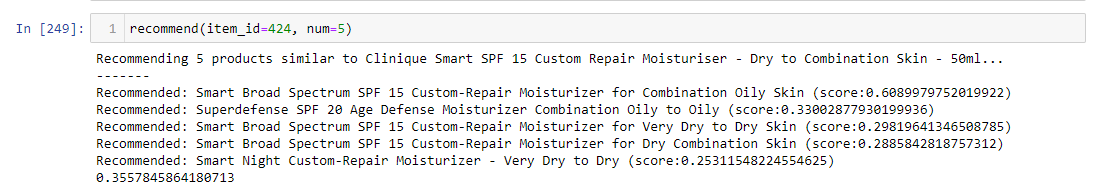


Figure 8.8 Tf-idf Cosine Similarity Index

# Conclusion and Future Works

# Bibliography

Adebo, A., 2020. *A Natural Language Processing Approach to a Skincare Recommendation Engine,* Dublin: National College of Ireland.

ADERIBIGBE, B., 2020. *Machine Learning for 5G Mobile and Wireless Communication,* London: London South Bank University.

Friedman, N. a. G. D. a. G. M., 1997. Bayesian network classifiers. *Machine learning,* 29(2), pp. 131-163.

JetBrains, 2020. *Python Developers Survey 2020 Results.* [Online]   
Available at: https://www.jetbrains.com/lp/python-developers-survey-2020/  
[Accessed 21 07 2021].

Junaidi, N. F. A. a. S. N. Z. M. a. A. N. a. K. H., 2021. A Personalized Web Cosmetic Recommendation System Based on Skin Type. *Applied Information Technology And Computer Science,* 2(1), pp. 225--234.

Lee, G., 2020. A Content-based SKincare Product Recommendation System.

Ojokoh, F. I. a. Y. F. a. B., 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal,* 16(3), pp. 261-273.

Shambour, Q. a. L. J., 2012. A trust-semantic fusion-based recommendation approach for e-business applications. *Decision Support Systems,* 54(1), pp. 768--780.

Songsri Tangsripairoj, K. K. P. P. Y. B., 2018. *SkinProf: An Android Application for Smart Cosmetic and Skincare Users.* Nakhon Pathom, IEEE, pp. 1-6.

Verma, K., 2021. *Muvi Blogs.* [Online]   
Available at: https://www.muvi.com/blogs/evolution-of-a-recommendation-engine.html#:~:text=The%20first%2Dever%20recommendation%20engine,to%20rate%20the%20messages%2F%20documents.  
[Accessed 29 07 2021].

# Appendix

## Project Gantt Chart

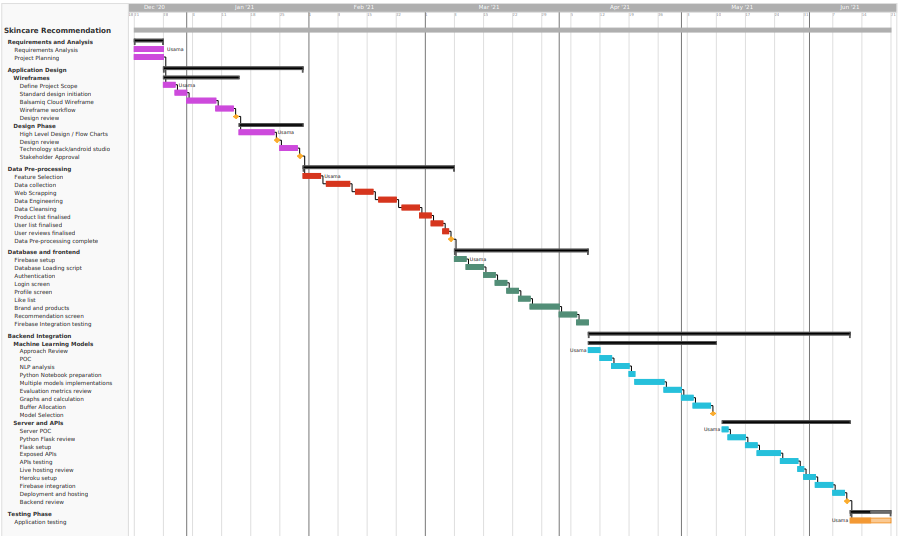


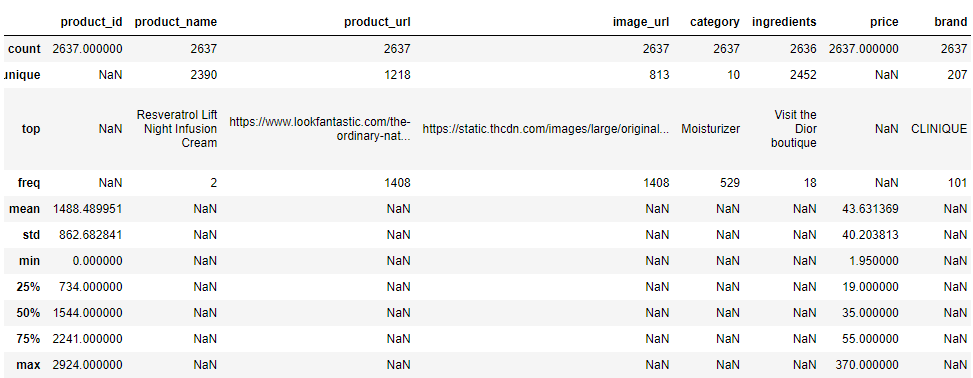
Figure 11.1 Gantt Chart

## Singular Value Decomposition (SVD) Results

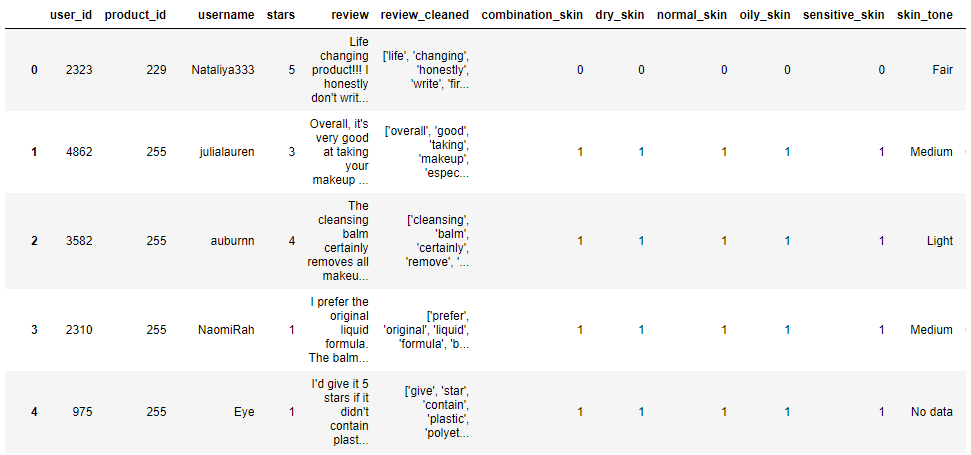
SVD algorithm was used in comparison to the KNN for predicting similar rating for a product and suggesting a new one based on the stars it received, but the results were not viable due to the reason that only a few products were bought multiple times and mostly were rated or reviewed only once which made the accuracy not worth using it for the production environment. Figure 11.2 shows the root means square error and the mean absolute error for the algorithm. It can be observed that due to only a single prediction class being available for training purposes the accuracy came out to be just 17%. Hence, a matrix working on the basis of correlation of similar ratings was used instead.



Figure 11.2 Accuracy Measure for SVD







1. Evaluation metric for Tfidf vector is cosine similiary (Equation 2)

2. Cosine similarity is a beyond accuracy metric

3. Language based analyser can't be calculated using any quantitative measure

4. A user ID has also been processed for comparing recommendations

5. Figure 8.8 shows the Tf-idf recommendation results

6. These results range from 60-90 depending upon the range of products available.

Firebase is a NoSQL database which supports JSON format which is what the applicaitons

API has been developed to receive and send results using the HTTP JSON format.

1. KNN and DT have been measured via a quantitative metric

2. MAPE (Equation 4) and Confusion matrix (Equation 6 & 7)

3. MAPE is a measure of how accurate a forecast system is.

4. It measures this accuracy as a percentage

5. Precision and Recall that can help with measuring the actual and false positive from the predictions.

1. The Mean Absolute Percentage Error (MAPE) is one of the most commonly used KPIs to measure forecast accuracy.

2. It ranges from a value of 0-1

3. 1 being bad and 0 is the best

4. MAE and RMSE can range from 0-inifinity based on the data being evaluated

5. MAPE is a negatively-oriented scores: Lower values are better

KNN

1. KNN takes user rating, userID and product ID as the input

2. It predicts on the basis of clustering the rating of the item

3. With 3 item in each cluster

DT

1. Decision tress consider the user characteristics

2. Skin specifications and such

3. The forecasting is made based on the product ID chosen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User 2323 | Rank | Tf-idf Vector | KNN | DT | VS |
|  | 1 | Umbrian Clay Pore Purifying Face Mask | Daily Cleanser Acne Treatment | Beauty Elixir | Pep-Start 2-in-1 Exfoliating Cleanser |
|  | 2 | Umbrian Clay Pore Purifying Face Exfoliator | Exfoliating Peel Gel | Exfoliating Peel Gel | La Roche-Posay Cicaplast Baume B5 Soothing Repairing Balm 100ml |
|  | 3 | Organic Aromessence Rose d'Orient Soothing Comfort Oil Serum | Deep Cleansing Exfoliator | The Clean Truth Foaming Cleanser | Calendula Deep Clean Foaming Face Wash |

To investigate further on the results produced by the recommendation algorithms I have manually reviewed their respective suggestions. Firstly, a random user is selected whose like list is populated based on the products they prefer. This is passed through the respective models to generate recommendations and finally the top viable suggestions are taken in to account.

On investigating the algorithms and manually verifying their recommendations it can be observed on the table below that the models have unsurprisingly reported significant variations between their recommendation. However, these suggestions have common characteristics and differ only on the basis of multiple algorithms working on different features of varying magnitude. The Tf-idf vector produces results on the basis of the ingredients, KNN take the user ratings into account whereas DT prioritizes user characteristics.

What communication service is used:

HTTP APIs have been used to communicate between the frontend and the server of the application

What Is incoming data:

User ID is received from the currently signed in user from the frontend

What is outgoing data:

These APIs drive the models that have implemented to predict the recommendations for the userID received based on their like list which is also received from firebase via the server.

What is processed behind the scene?

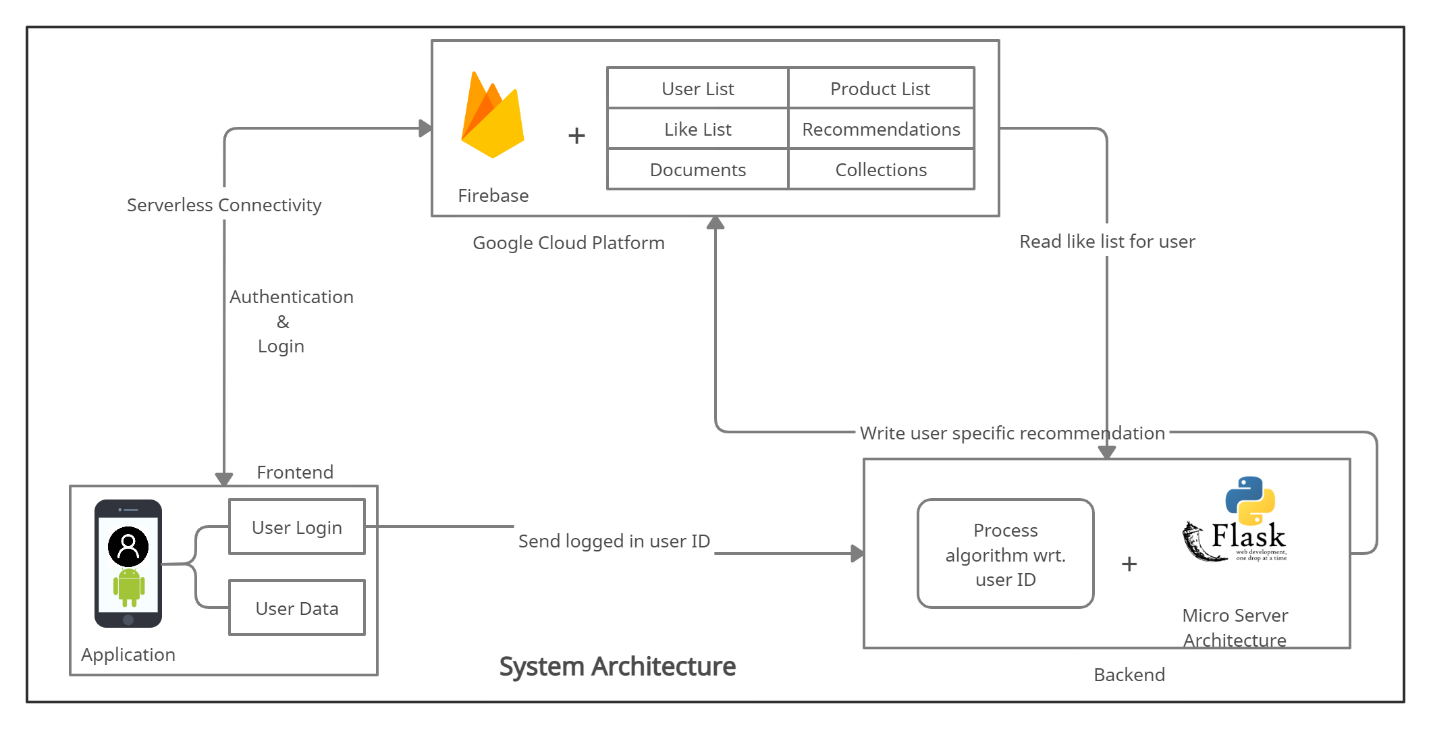
Currently signed in UserID is received from the front when a user logs in. API’s receive user like list, characteristics and user reviews based on that ID. The data is processed by the server by formatting it and preparing a data frame before passing it into the ML model. Respective data frames are passed through the models. The models generate results in term of product IDs that should be recommended to that specific user. These product IDs are populated back in to firebase in the user’s recommendation collection by the server which is then fetched by the frontend when the user visits the recommendation screen.

What is the format of the data taken in and going out:

The data is in JSON format which is natively support by HTTP unlike SOAP.

Why is this service a choice?

We used HTTP because it supports JSON format which is ideal for communicating between firebase and the flask server. Unlike REST and SOAP, HTTP does not require a proper structure or standard to be followed hence providing the developer the flexibility for creating a lightweight API that specifically suits the needs. SOAP works in XML format which was not required in the project scope.



Following a 3 layered architecture the Frontend, server and the database are interconnected. In this section I’ll be explaining how these parts of software communicate with one another.

Firstly, the user interacts with the front-end which is an android app where they sign up with an ID, login and browse the products list. Upon adding one of the items to their like list, the product ID is communicated to the Firebase database. Whenever a user logs in, the current signed in user ID is sent to the server to process their respective liked items, user characteristics and rated products for specific models discussed further in the literature. The APIs are responsible for catering this information to the models and once the recommendation have been generated the server then conveys this to Firebase and populates the user recommendation collection. Following this scenario when the user visits the recommendation screen the respective product IDs for the user is fetched from the Firebase’s recommendation collection and displayed to the user in 2 different tabs of collaborative and content-based algorithm.

The user first interacts with the front-end, where they create an account, login, and explore the product list.

The product ID is conveyed to the database when one of the products is added to their like list.

When a user registers in, the server sends their current signed in user ID to process their favourite things, user attributes, and rated products for specific models covered in the literature. The APIs are in charge of feeding this data to the models, and after the recommendations are formed, the server sends them to Firebase, where they are added to the user suggestion collection.

When a user enters the suggestion screen, the user's respective product IDs are retrieved from Firebase's recommendation collection and displayed to the user in two different tabs of collaborative and content-based algorithms.

