# **🔥ShopSmart TikTok🔥**

# Team Kudasai

Members:

Si Ming Zhou

Teh Min Ze

Marcus Yeo

Khoo Yong Hui

# 

# **1. Problem statement 📋**

Our team have chosen the problem statement on “Enhancing Tailored Discovery on TikTok shop”

By choosing this problem statement, we aim to create a more intuitive, personalised shopping experience that can help users find exactly what they're looking for quickly and easily. Our goal is to enhance the user experience by improving the user interface, adding features to personalise recommendations and ensure that every user's journey on TikTok Shop is enjoyable and efficient.

# **2. Setup, installation & how to run 🏃**

## 2.1 Setup and Instructions

1. (Back-end) On a windows machine, run “wsl.exe --install Ubuntu” to install and run Ubuntu terminal. Install redis and run “redis-server” to start the redis server.
2. (Back-end) cd to “server” directory, run “npm install” to install dependencies and run “npm run dev” to start the node.js server.
3. (Back-end) cd to “model” directory, create a new python env and run “pip install -r requirements.txt” to install dependencies. Run “python app.py” to start the model’s flask server.
4. (Front-end) cd to “client” directory and run “npm run dev”.
5. Navigate to “<http://localhost:5173/>” to view the app

# **3. Features and functionality of ShopSmart 🕺**

## 3.1 Overview of ShopSmart

We want to find a way to integrate watching TikTok videos with product recommendations as we recognise that TikTok FYP videos reflect the user’s interest. Hence, our solution aims to use these videos to recommend products according to their preferences. This is achieved by implementing multiple features to improve the shopping experience of the user through hyper personalisation using artificial intelligence and real time data.

Our recommendation system consists of 2 tiers:

The first tier is a “star” button that the user can press on. If a user were to “star” a video, products shown in the video or products similar to that shown in the video will be prioritised. If users were to “star” a video, this signifies to the system that the products in this specific video are of interest, thus the system will prioritise this product in the tiktok shop recommendation. When the user visits the tiktok shop, the top few products will be from the videos that have been star-ed, the subsequent products will be from the second tier.

The second tier considers the duration spent on a video by the user. This will serve as the secondary rank for recommending products in the tiktok shop. For every video that the user watches, the system will take into consideration the duration spent on the video. For videos with longer interaction time, the system will deem those videos as favourable and will recommend products accordingly from the videos in the tiktok shop. Recommended products can be the exact products shown in the video or other similar products.

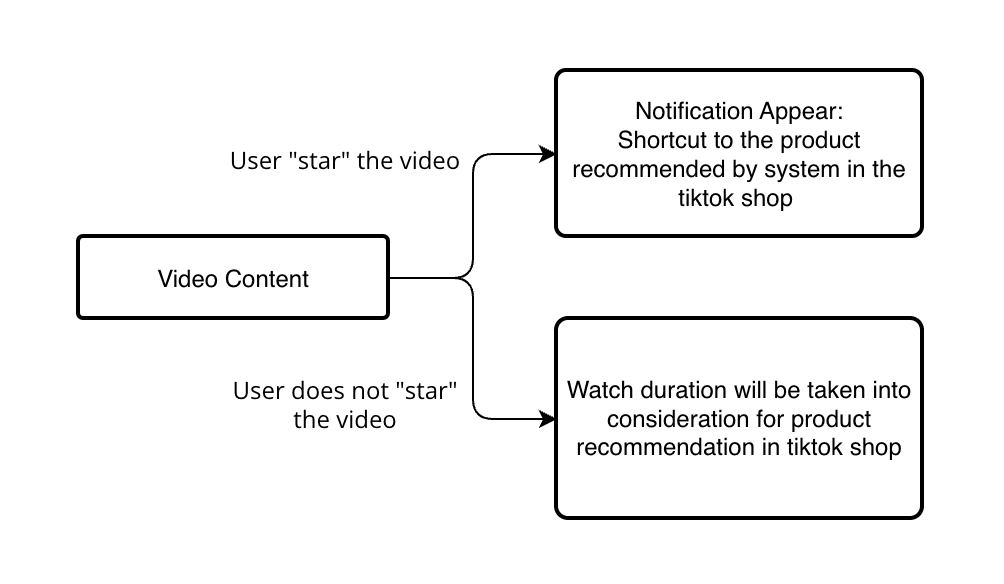


Fig 3.1.1: Flow for every video user interacts with

With reference to Fig 3.1.2, if a user were to “star” a video with content about tea, the “Nongfu spring fruit tea” will be recommended at the top of the user’s FYP in tiktok shop. Assuming the user had only “star” one video about tea and the same user had spent the most time watching a video on a decompression toy, the toy will be considered second in the recommendation. This means that “star” products will always take precedence, meaning if the user were to spend 5 seconds on the “star” video and 20 seconds on another video, the system will always recommend the “star” product first.

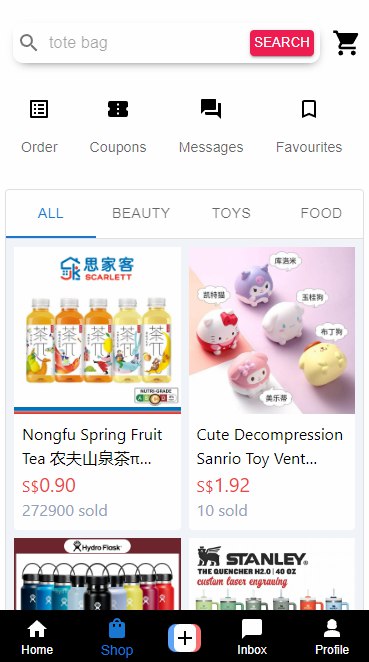


Fig 3.1.2: TikTok Shop

## 3.2 Star products in TikTok videos 🌟

### 3.2.1 Features

### 

Fig 3.2.1.1 Screenshot of “star” button

In our solution, we implemented a new button named “star”. Users will be able to “star” videos that are of their interest and have products in the video or products similar to those in the video recommended to them in the TikTok shop. This provides a seamless shopping experience for users as their TikTok shop will be a direct representation of their personal interest.

Keeping in mind that there are various reasons as to why users watch certain videos, such as for entertainment or learning purposes, there are occasions when users are not necessarily interested in purchasing content in the videos. Hence the implementation of the “star” button to only consider certain videos of interest to generate product recommendations.

Furthermore, a user watch duration will also impact interests in the product shown in the video. For example, if a user watches for at least 10 seconds, it would be deemed as only a slight interest in the product.

### 3.2.2 Functionality

When a user uses the button, it is given as a definite interest in the showcased product. Hence, a higher weightage is given to the “star” button, whilst watch duration is given a lower weightage, as it does not indicate definite interest.

The algorithm will slowly learn the interests of the user and recommend their personalised top 5 products.

In the case of cold start, where a new user had just created an account, the shop will choose the top 5 best-selling products instead.

## 3.3 Product showcase while watching 🛍️

### 3.3.1 Features

### 

Fig 3.3.1.1 Screenshot of product link while watching video

Sometimes while watching videos, creators will showcase or review products that they have used in their lives. Creators may sometimes omit information on the product, making it hard for interested users to find the exact product. Instead of commenting on the videos and waiting for a response. Users who are interested in the product would now be able to view the exact product in the TikTok shop that is showcased in the video. (Fig 3.3.1.1)



Fig 3.3.1.2 Screenshot of product page

By clicking on the link, it would bring them to the exact product page in the TikTok shop (Fig. 3.3.1.2)

### 3.3.2 Functionality

By using machine learning, we first determine what product is shown by providing keywords analysed from the video. These keywords are used to find the products showcased in the video, then we provide a link on screen without compromising on overall user experience.

# **4. Development of ShopSmart ⚒️**

## 4.1 Development tools used to build the project

### 4.1.1 Node.js and npm

To test our user interface and have reusable packages that can add value to our application.

### 4.1.2 VS code

Provides a better environment for writing code, ensuring it is legible and clean.

### 4.1.3 pgAdmin

## 4.2 APIs

### 4.2.1 PostgreSQL Database

To interact with PostgreSQL database to run SQL queries

### 4.2.2 Redis Cache

To enable the usage of Redis for caching to improve efficiency.

## 4.3 Assets used

### 4.3.1 Material UI

Material-UI is a popular React UI framework. It provides React components that follow Google's Material Design guidelines, allowing developers to quickly build attractive and responsive web applications.

In our application UI, we made use of

* Components and
* Icons

## 4.4 Libraries used

### 4.4.1 cors

### 4.4.2 dotenv

Loads our .env files that contain confidential information/credentials. This ensures that security is maintained throughout our application.

### 4.4.3 express

### 4.4.4 express-json

### 4.4.5 pg

### 4.4.6 redis

## 4.5 Front end architecture

The front end architecture of ShopSmart is designed to be modular, maintainable and scalable.

### 4.5.1 Model-View-ViewModel architecture

We use the MVVM architecture as it is able to facilitate a clear separation between the UI and business logic, making it suitable for modern front-end development frameworks like React.

The MVVM architecture comprises of three areas:

* **Model**: The model presents the data and business logic of the application.
* **View**: The view represents the UI of the application. It displays data from the ViewModel and sends user interactions (events) back to the ViewModel.
* **ViewModel:** The ViewModel acts as an intermediary between the Model and the View. It holds the presentation logic and state of the View. It handles data binding, automatically updating the View when the Model changes and vice versa.

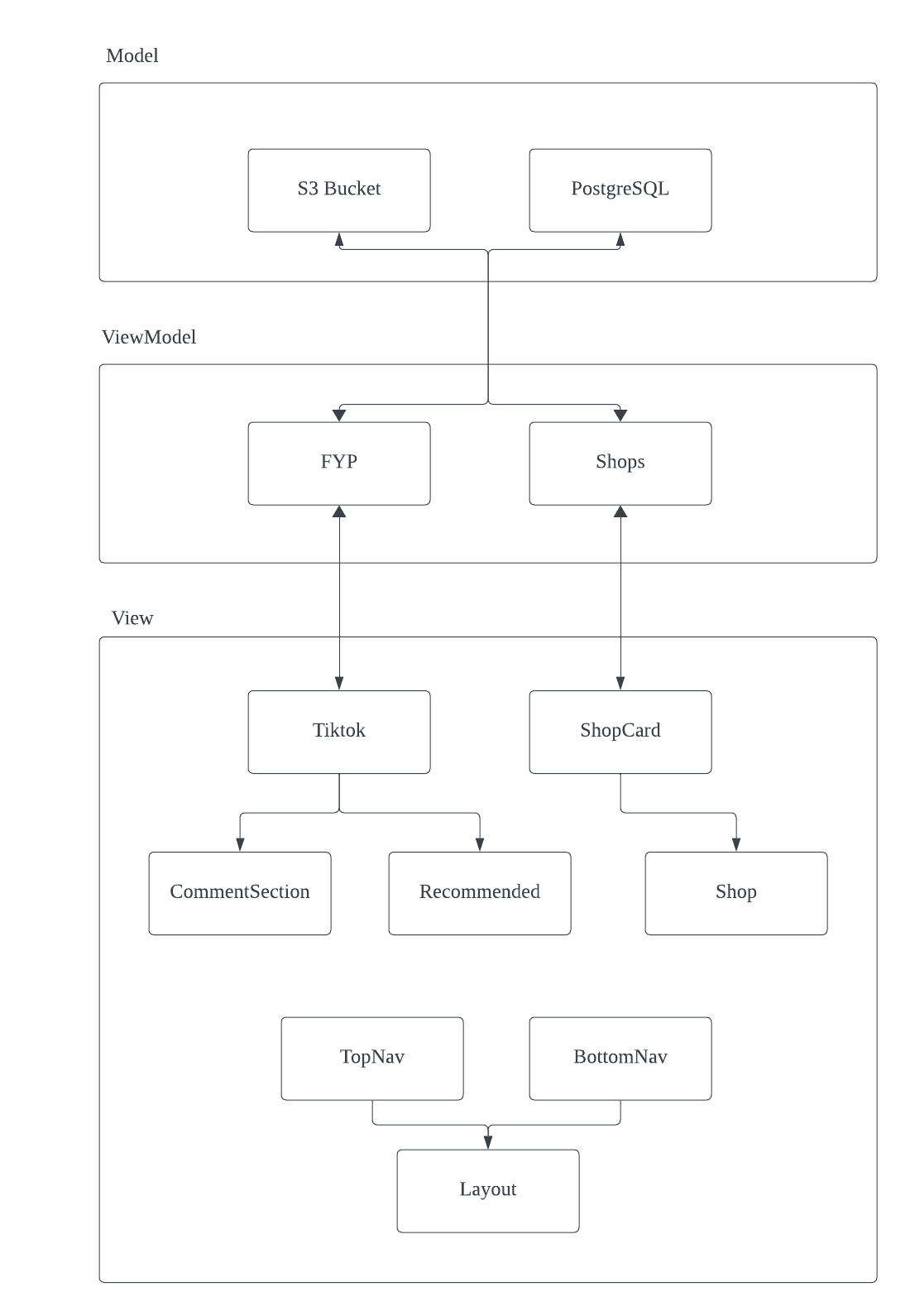
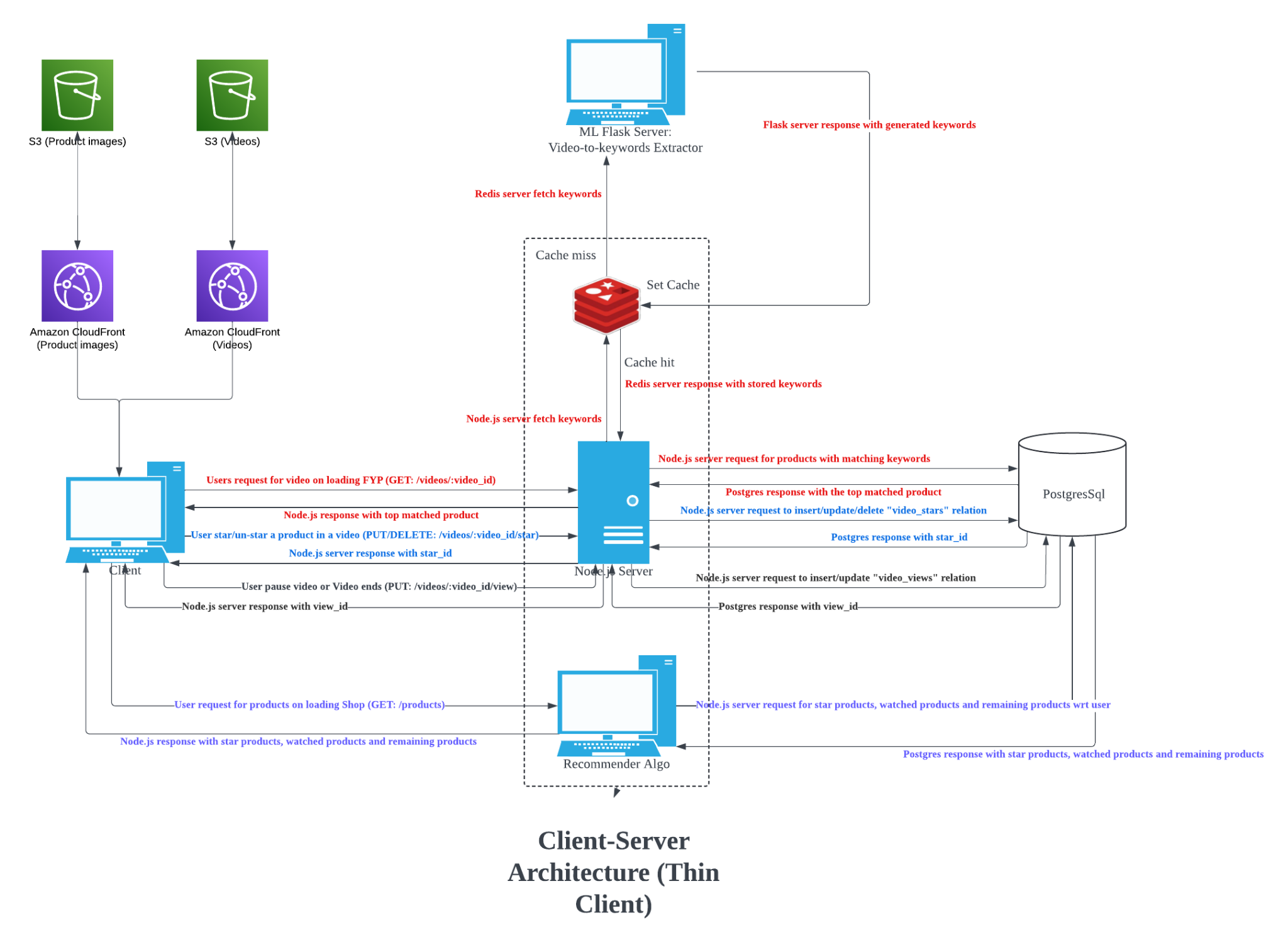


Fig 4.5.1.1 MVVM front end architecture of ShopSmart

* **Model**: The **S3** and **PostgreSQL** handles our application data and implements business logic to the data it stores.
* **ViewModel**: The **FYP** and **Shops** components handle all of the data fetching/posting and from/to the Model. The data contains all the video related information as well as product information that we need for display. This data is then passed on to the ViewModel components and vice versa when updates are made on the UI.
* **View**:
  + Data is passed to **Tiktok** and **ShopCard** for visualisation (parent views).
    - **Tiktok** represents each tiktok video and
    - **ShopCard** represent each product card in the Tiktok shop.
  + **CommentSection, Recommended** and **Shop** represent the sub-views where selected data is passed from the parent view for further visualisation
    - **CommentSection** displays all comments related to its parent video,
    - **Recommended** displays the recommended product when its parent video is starred and
    - **Shop** represents the individual product detail page.
  + **TopNav** and **BottomNav** make up our application layout, **Layout**, and stay separate from the application logic.

## 4.6 Backend architecture



4.6.1 Client-server architecture of ShopSmart

* S3 buckets: Store product images and videos objects
* Cloudfront: CDN to cache product images and videos at edge locations and serve to clients when requested
* Node.js server: Handle requests from clients by communicating with redis cache, ML flask server and Postgres database. Contains recommendation algorithms to serve client with ranked products.
* Postgres database: Store product and product-related data (ie product\_tags, product\_ratings, product\_reviews), videos and video-related data (ie video\_comments, video\_saves, video\_stars, video\_views), tags data, users data and shops data.
* ML server: Download video stream from CDN and processed by our model to return keywords based on frames extracted from video.
* Redis server: Cache keywords returned by ML server to serve multiple clients with minimal latency.

# **5. Model 🧑‍💻**

Our model consists of two different parts that together can detect the key product from a video: Object Detection and Text Extraction. We extract frames from videos and process these frames with our model consisting of the abovementioned 2 parts to output keywords relating to the products shown in the video. These keywords are then used to search for products within the Tiktok shop.

Object Detection  
The main model used here is the YOLO World Model. It is a real time based approach for Open-Vocabulary Detection tasks. It is employed as part of our model to detect objects of significant importance in the video.

Text Extraction

Tesseract OCR is used to extract the texts from the frames. The texts are then processed and tokenized using NLTK to remove noises.

Keywords from object detection and text extraction are combined and processed using TF-IDF to rank them from most to least important. The most important keywords are then used to find the products in the database.

# **6. Limitations ☠️**

## 6.1 Trade-off between accuracy and speed

To get better keywords from the video, more frames and more complex models have to be used. By employing more frames in our models or using more complex models, time to extract keywords will be exponentially longer. In fact, the duration will likely exceed the duration of most videos.

## 6.2 Accuracy of product identification

TikTok often contains complex visuals, rapid movements and showcases multiple products which can make it difficult for models to accurately identify products. Furthermore, identifying specific brands amongst similar-looking products can be challenging.

## 6.3 User privacy

Analysing videos and tracking interactions might raise privacy concerns.

## 6.4 Scalability

Analysing a vast number of videos in real-time requires significant computational resources, furthermore, tracking user engagement will use extra storage spaces and computation.

## 6.5 Advertisement fatigue

Users might be fatigued from constant prompts for products which may reduce engagement.

# **7. Future consideration 🤑**

## 7.1 Enhanced product recognition by using custom vocabulary libraries

The product recognition model can be improved to better handle a variety of video content and improve accuracy in product identification. One possible way is to use custom vocabulary libraries. Current constraint faced is that the corpuses readily available are outdated, they do not contain terms on trend and newer products. This poses a difficulty in extracting these keywords. By employing custom vocabulary libraries, the model will have an increased accuracy in identifying even the latest trendy products. Custom vocabulary libraries can possibly be obtained by using the products already in the Tiktok shop.

## 7.2 User privacy and transparency

This can be mitigated by allowing users to control their data via settings and being transparent by informing them of the usage of their data for product recommendation.

## 7.3 User experience optimization

Users can be allowed to turn off product recommendations whilst watching videos. At the same time, they can allow tracking such that when they navigate to the shop, it would still show their personalised recommendations.

## 7.4 Scalability considerations

Cloud computing can be leveraged to handle large processing loads and scale efficiently.

## 7.5 Bias mitigation

Use of a more diverse dataset to train models and regularly audit recommendations to ensure fairness and inclusivity.

User feedback is important here to identify and correct biases.

## 7.6 Model usage in recommendation system

We recognise the benefit of creating a model based recommendation system. However, due to computational limitations of having to host another server for the model, we had decided to opt for a simpler recommendation system in order to showcase our main features instead.

Hence, we have included a written model recommendation system that we would have integrated into our solution. This model uses TF-IDF and SVC. The idea is, after training the model, it will predict what a user would like based on the user’s actions. For example, watch duration and “star” likes are given different weightages. From the descriptions and title of products, the model will recommend products that are the most similar to the user’s preferences, thereby giving a more varied selection of products.

## 7.7 Taking user circumstances/preferences into consideration

Our idea is to use toggles, drop down menus, sliders and pickers for users to indicate their preferences. These preferences will be used to filter products on top of their personalised recommendations. (Fig.7.7.1)

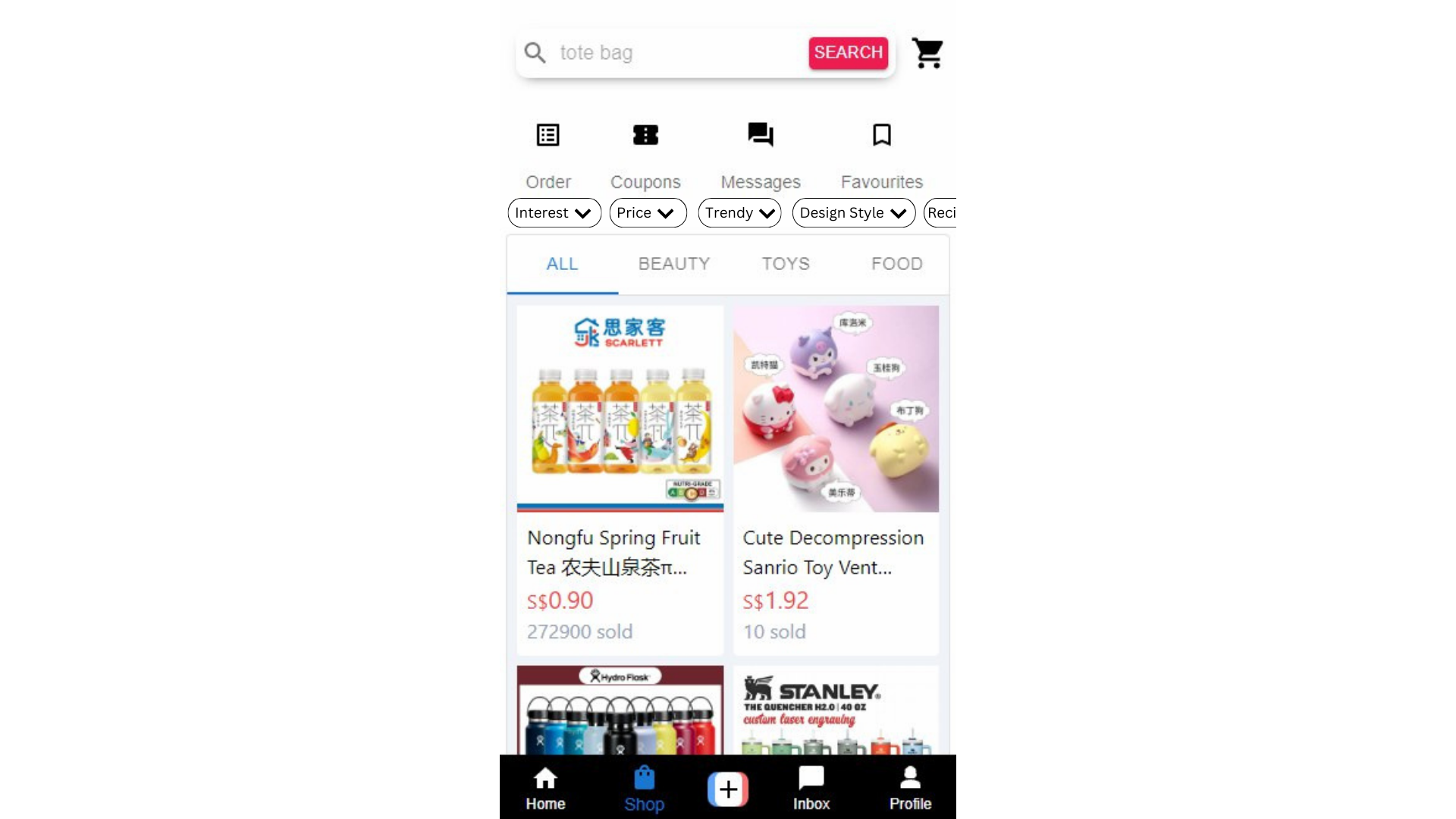


Fig 7.7.1 Example of how TikTok shop would look like with user circumstances extraction

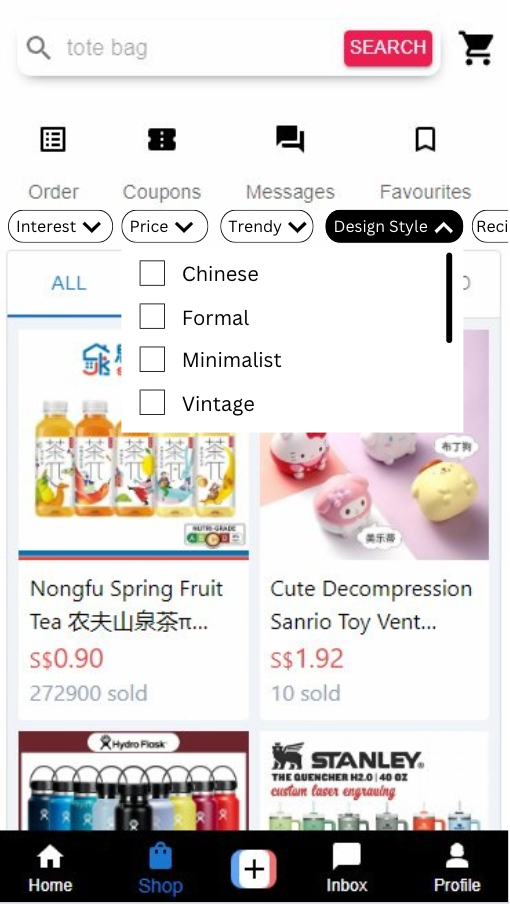


Fig 7.7.2 Design style selection

A multi-selection box can be implemented for users to select one or more design styles. Sellers who list their products can include design styles in their listing for users to easily filter through products. (Fig 7.7.2)

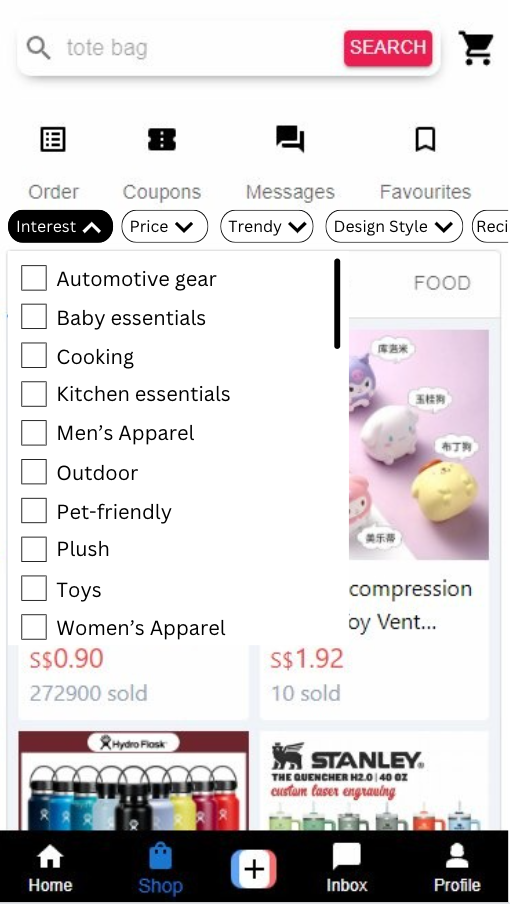


Fig 7.7.3 Interest selection

A multi-selection box can be implemented for users to select one or more of their interest(s). Sellers who list their products can include interests, such as cooking, in their listing for users to easily filter through more specific products. (Fig 7.7.3)

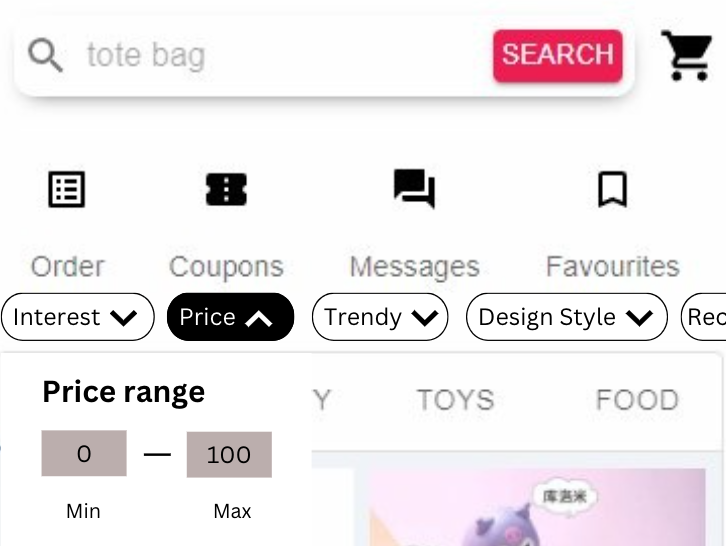


Fig 7.7.4 Price selection

Users can choose their price range by typing in their minimum price and maximum. These price ranges will be used to filter products to those that fit in the range. (Fig 7.7.4)

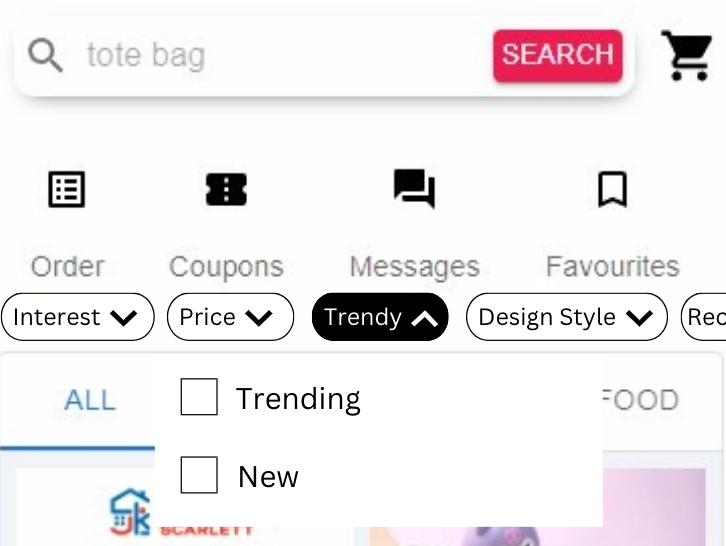


Fig 7.7.5 Trend selection

Users can select trending and/or new products. Trending are filtered products from a user’s recommended products that are currently selling the most in a week. New products are the newest posted products from the user’s pool of recommended products. (Fig 7.7.5)

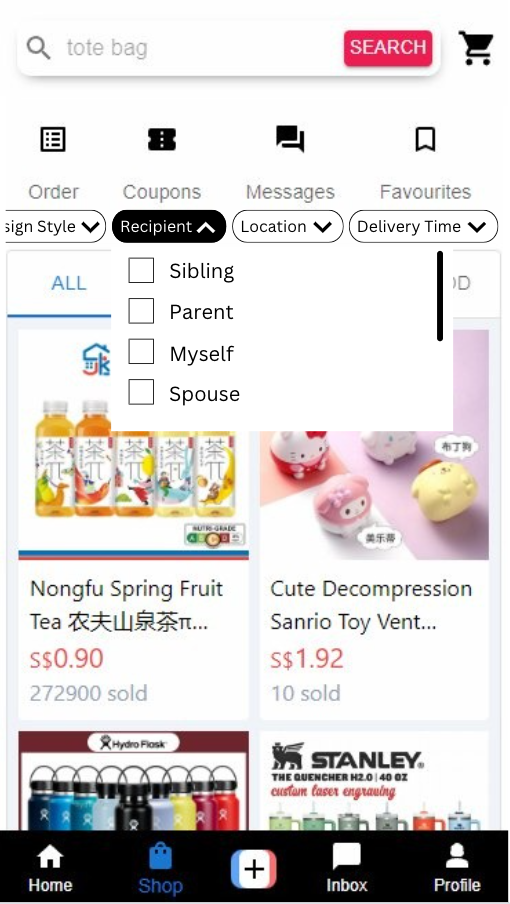


Fig 7.7.6 Recipient selection

Users can multi-select recipients. Listers can include these details to show who is perfect to receive these gifts. For example, if a user is currently interested in engagement rings, they can include the “Spouse” recipient to narrow down their search. (Fig 7.7.6)

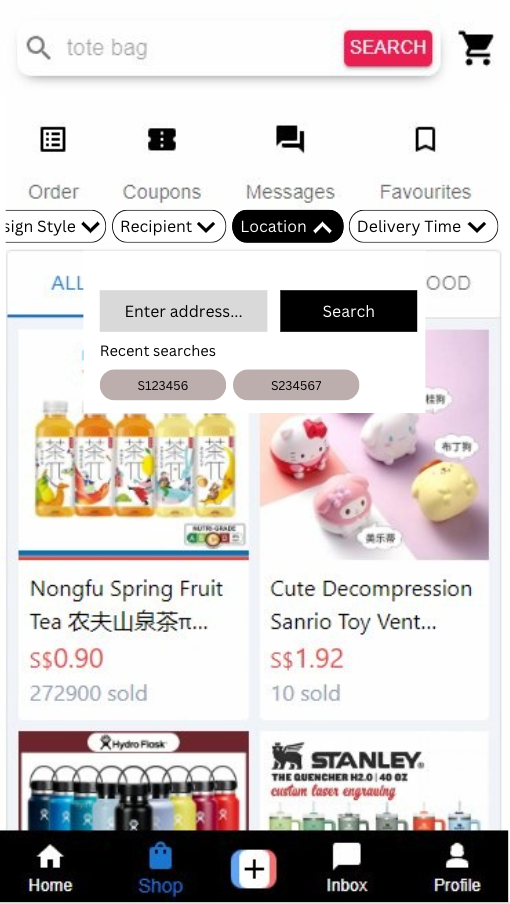


Fig 7.7.7 Delivery address search

Users can search for their delivery address to see filter products that can be delivered to their house. (Fig 7.7.7)

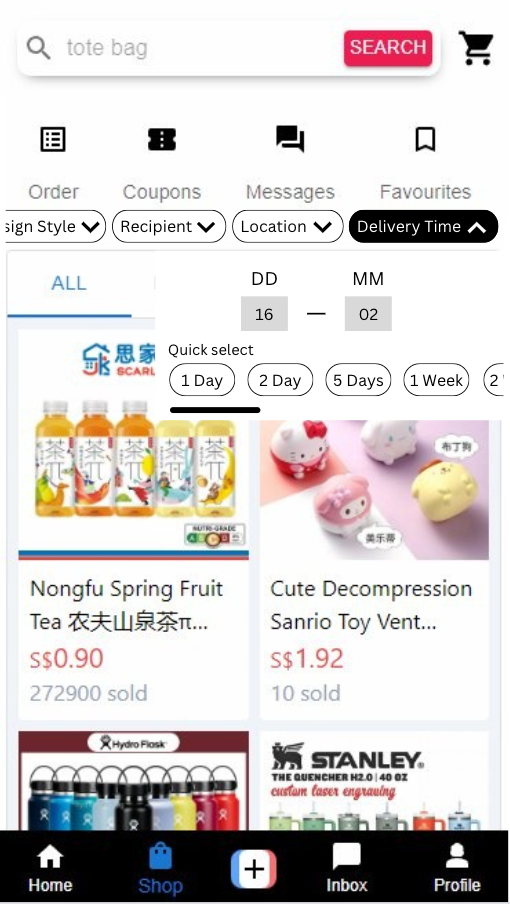
****

Fig 7.7.8 Delivery time selection

Users can select an estimation of when they would want their product to be delivered by. They are also able to quick select the estimated delivery time frame. (Fig 7.7.8)

# 

# **8. Link to github repository & Video 📹**

Github Link: <https://github.com/elginsimingzhou/tt_shop.git>

Video Link: <https://youtu.be/aJ0fZkOSlAg>