

Database & Solutions Integration Plan

using HUAWEI CLOUD GaussDB for MySQL Database
for high performance, flexibility and reliability



Team EcoShop

Teo Kai Xiang, Ho Wing Yip, Theodore Lee Chong Jen, Billy Cao Yuxuan

Technical Summary

EcoShop is an **innovative sustainability marketplace** which allows sellers and service providers to **list both used items and repair services** for certain items (like electronics) respectively. This helps to **mitigate climate change** by both **reducing waste** and the amount of **goods manufactured**.

To enhance the **user experience**, several features such as a **video-oriented social media network**, **natural language processing-powered question and answer functions**, and **real-time chat** are required.

As these are **critical functions** of our platform which require **complex data manipulation** and **high performance**, we chose **HUAWEI CLOUD GaussDB for MySQL** as our database for platform metadata due to its **unparalleled performance and reliability**, to deliver a **stellar user experience**; we will use **HUAWEI CLOUD's other services** (such as **Media Processing Center**, **FunctionGraph**, **Image Recognition** and **Object Storage**) for other aspects of the platform. **Connections and data manipulation** can use **standard MySQL** libraries and features, taking advantage of **GaussDB for MySQL's compatibility with MySQL**.

In the queries below, a '?' (question mark) represents a parameter that will be substituted in, using MySQL prepared statements.

Our queries and structure have been tested using MySQL, as we do not have access to GaussDB for MySQL yet.

Table Definitions

Solid underlines denote primary keys, and dotted underlines denote foreign keys.

user (user, pass, plan, reputation, created)

user_product_bookmark (bookmark, user, product)

user_video_react (id, user, video, react)

product (id, name, price, tags, quantity, owner, description, boosted, created, impressions, points)

product_attr (product, attr_name, attr_text, attr_int)

product_image (product, obs_image, order)

social_video (id, obs_location, impressions, created)

social_video_product (video, product)

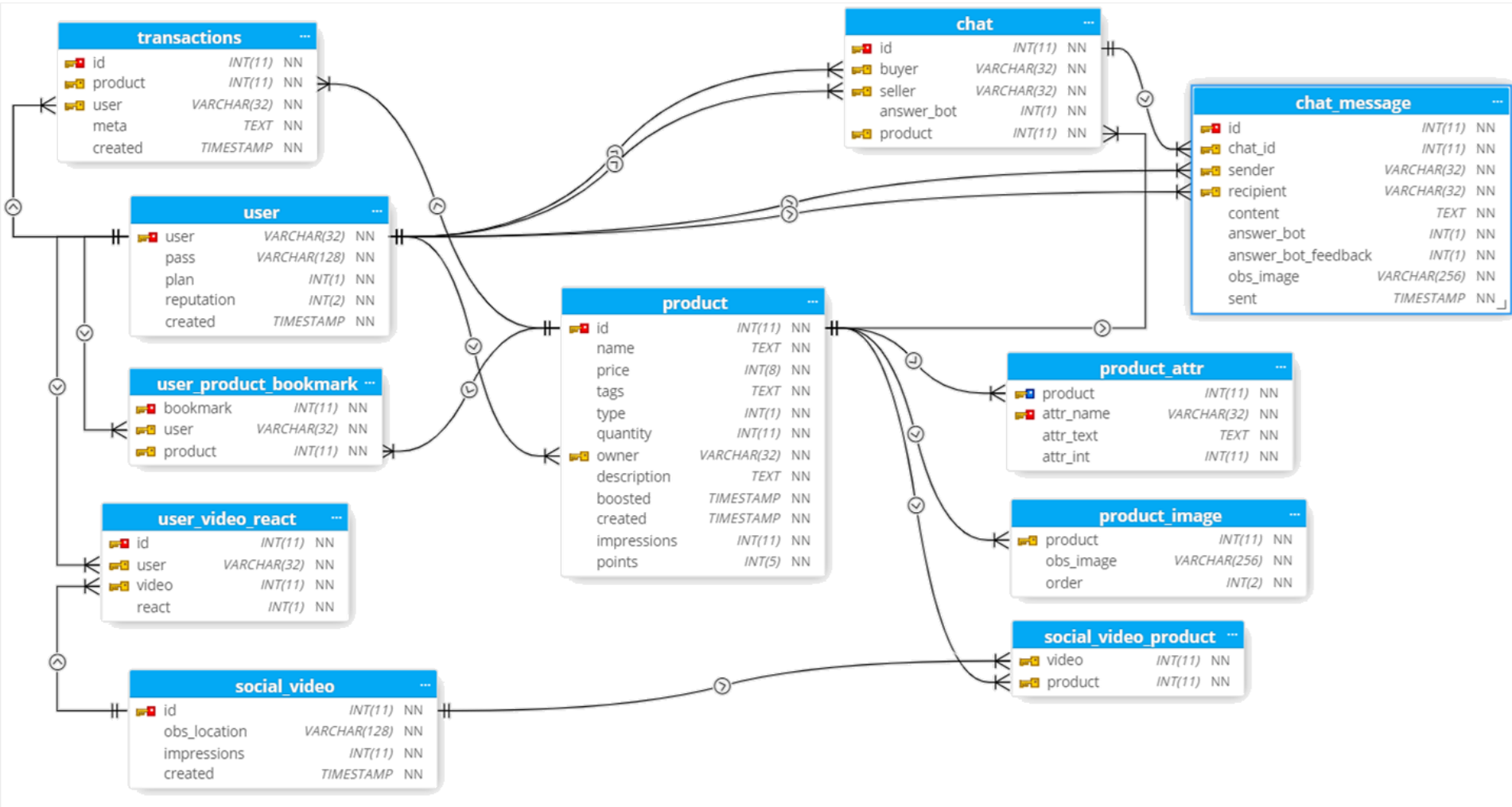
chat (id, buyer, seller, answer_bot, product)

chat_message (id, chat_id, sender, recipient, content, answer_bot, answer_bot_feedback, obs_image, sent)

transactions (id, product, user, meta, created)

Table Relationships

Relationship Diagram



Key SQL Statements

The following statements are going to be used by our API to fetch data from GaussDB. Our API will be invoked by our frontend to gather and present data to our user in a user-friendly, understandable and insightful manner.

Our product's database design will take advantage of GaussDB's high performance by creating indices on the necessary entity attributes.

In the queries below, a '?' (question mark) represents a parameter that will be substituted in, using MySQL prepared statements. The queries are tested against MySQL as we do not have access to GaussDB yet.

Purpose	SQL Statement	Remarks
<i>User-related Tasks</i>		
User creation	<code>INSERT INTO `user` (`user`, `pass`, `plan`, `reputation`) VALUES(?, ?, ?, ?)</code>	Value of <code>pass</code> field is hash of user's password.
User login	<code>SELECT `pass` FROM `user` WHERE `user` = ?</code>	Gets user's password hash to verify against entered password.
<i>Product / Service Listing-related Tasks</i>		
Create product / service listing	<code>INSERT INTO `product` (`name`, `tags`, `price`, `type`, `quantity`, `owner`, `description`, `impressions`, `points`) VALUES (?, ?, ?, ?, ?, ?, ?, ?, 0, 0)</code>	Tags generated by CTRLsum running on HUAWEI Cloud FunctionGraph or Cloud Server
Create product attributes	<code>INSERT INTO `product_attr` (`product`, `attr_name`, `attr_text`, `attr_int`) VALUES (?, ?, ?, ?)</code>	Product attributes comprise a name and a text or integer value

Purpose	SQL Statement	Remarks
Add images to listing	<pre>INSERT INTO `product_image` (`product`, `obs_image`, `order`) VALUES (?, ?, ?)</pre>	Images stored in HUAWEI Cloud Object Storage
Search for listings – ordered by most recent with number of bookmarks and product image	<pre>SELECT `id`, `name`, `price`, `type`, `quantity`, `owner`, `obs_image`, COUNT(`user_product_bookmark`.`user`) AS `bookmarks` FROM `product` INNER JOIN `product_image` ON `product_image`.`product` = `product`.`id` LEFT OUTER JOIN `user_product_bookmark` ON `user_product_bookmark`.`product` = `product`.`id` WHERE MATCH(`name`, `tags`) AGAINST(? IN BOOLEAN MODE) AND `product_image`.`order` = 1 GROUP BY `id` ORDER BY `id` DESC LIMIT 21</pre>	Full-text-search index on product table is used. The first ~5 products with videos attached will be shown at the top of the results page. Cursor pagination is used instead of offset pagination for better performance.
Search for listings with filters (on listing attributes) with number of bookmarks and product image	<pre>SELECT `id`, `name`, `price`, `type`, `quantity`, `owner`, `attr_name`, `attr_text`, `attr_int`, `obs_image`, COUNT(`user_product_bookmark`.`user`) AS `bookmarks` FROM `product` INNER JOIN `product_attr` ON `product`.`id` = `product_attr`.`product` INNER JOIN `product_image` ON `product_image`.`product` = `product`.`id` LEFT OUTER JOIN `user_product_bookmark` ON `user_product_bookmark`.`product` = `product`.`id` WHERE MATCH(`name`, `tags`) AGAINST(? IN BOOLEAN MODE) AND `product_attr`.`attr_name` = ? AND `product_attr`.`attr_int` > ? AND `product_image`.`order` = 1 GROUP BY `id` ORDER BY `id` DESC LIMIT 21</pre>	

Purpose	SQL Statement	Remarks
Update listing info	<pre>UPDATE `product` SET `name` = ?, `price` = ?, `tags` = ?, `type` = ?, `quantity` = ?, `owner` = ?, `description` = ?, `impressions` = ?, `points` = ? WHERE `id` = ?</pre>	
Get listing details	<pre>SELECT `id`, `name`, `price`, `tags`, `type`, `quantity`, `owner`, `description`, `points`, `created` FROM `product` WHERE `id` = 1</pre>	Listing images are stored in HUAWEI Cloud Object Storage
Get all listing images	<pre>SELECT `obs_image` FROM `product_image` WHERE `product` = ? ORDER BY `order`</pre>	
Fetch listings by user with number of bookmarks and product image	<pre>SELECT `id`, `name`, `price`, `type`, `quantity`, `owner`, `obs_image`, COUNT(`user_product_bookmark`.`user`) AS `bookmarks` FROM `product` INNER JOIN `product_image` ON `product_image`.`product` = `product`.`id` LEFT OUTER JOIN `user_product_bookmark` ON `user_product_bookmark`.`product` = `product`.`id` WHERE `owner` = ? AND `product_image`.`order` = 1 GROUP BY `id` ORDER BY `id` DESC LIMIT 21</pre>	
Get common attributes by listing name, sorted by usage frequency	<pre>SELECT `attr_name`, COUNT(`attr_name`) AS `frequency` FROM `product` INNER JOIN `product_attr` ON `product`.`id` = `product_attr`.`product` WHERE MATCH(`name`, `tags`) AGAINST(? IN BOOLEAN MODE) GROUP BY `attr_name` ORDER BY `frequency` DESC LIMIT 7</pre>	Used to suggest attributes which users can add to listings , or add as filters when searching listings
User bookmarks listing	<pre>INSERT INTO `user_product_bookmarks` (`user`, `product`) VALUES (?, ?)</pre>	

Purpose	SQL Statement	Remarks
Get user's bookmarked listings	<pre>SELECT `id`, `name`, `price`, `type`, `quantity`, `owner`, `obs_image`, COUNT(`user_product_bookmark`.`user`) AS `bookmarks` FROM `product` INNER JOIN `product_image` ON `product_image`.`product` = `product`.`id` LEFT OUTER JOIN `user_product_bookmark` ON `user_product_bookmark`.`product` = `product`.`id` WHERE `user_product_bookmark`.`user` = ? AND `product_image`.`order` = 1 GROUP BY `id` ORDER BY `id` DESC LIMIT 21</pre>	
<i>Video Platform-related Tasks</i>		
Search for videos (returns likes, dislikes, location, products)	<pre>SELECT `product`.`id`, `name`, `price`, `type`, `quantity`, `owner`, `obs_location`, `social_video`.`id`, COUNT(CASE WHEN `user_video_react`.`react` = 1 THEN 1 ELSE NULL END) as `likes`, COUNT(CASE WHEN `user_video_react`.`react` = 2 THEN 1 ELSE NULL END) as `dislikes` FROM `product` INNER JOIN `social_video_product` ON `social_video_product`.`product` = `product`.`id` INNER JOIN `social_video` ON `social_video_product`.`video` = `social_video`.`id` LEFT OUTER JOIN `user_video_react` ON `user_video_react`.`video` = `social_video`.`id` WHERE MATCH(`name`, `tags`) AGAINST("laptop" IN BOOLEAN MODE) GROUP BY `social_video`.`id`, `product`.`id` ORDER BY `socia_video`.`id` DESC LIMIT 5</pre>	<p>Video file stored in HUAWEI Cloud Object Storage</p> <p>The first ~5 products with videos attached will be shown at the top of the results page.</p>

Purpose	SQL Statement	Remarks
Create video	<code>INSERT INTO `social_video` (`product`, `obs_location`, `impressions`, `created`) VALUES (?, ?, ?, ?)</code>	Video file transcoded by HUAWEI Cloud Media Processing Centre
Link listings(s) to video	<code>INSERT INTO `social_video_product` (`video`, `product`) VALUES (?, ?)</code>	Listings featured in a video will be displayed below the video
Viewer likes video	<code>INSERT INTO `user_video_react` (`user`, `video`, `type`) VALUES (?, ?, 1)</code>	
Viewer dislikes video	<code>INSERT INTO `user_video_react` (`user`, `video`, `type`) VALUES (?, ?, 2)</code>	
<i>Chat-related Tasks</i>		
Start chat	<code>INSERT INTO `chat` (`buyer`, `seller`, `answer_bot`, `product`) VALUES (?, ?, ?, ?)</code>	
Get chat messages	<code>SELECT `sender`, `recipient`, `content`, `sent`, `obs_image`, `answer_bot` FROM `chat_message` WHERE `chat_id` = ? ORDER BY `id` DESC LIMIT 50</code>	
Buyer / seller or service provider sends message	<code>INSERT INTO `chat_message` (`chat_id`, `sender`, `recipient`, `content`, `answer_bot`, `answer_bot_feedback`, `obs_image`, `sent`) VALUES (?, ?, ?, ?, ?, ?, ?, ?)</code>	User provides message data
Chatbot answers buyer message using listing description		Chatbot provides message data based on listing description

Database Workload, Statistics and Monitoring

Our database schema has been designed specifically to enable high-performance querying, incorporating indices to avoid full table scans. **All queries shown above avoid full table scans due to our database indices.**

There are also several key statistics which we will use to monitor both **database performance** and our **users' usage patterns** within our app.

Performance Statistics

Database Statistic	Purpose / Significance	Remarks
Queries per second	Provides a rough estimate of overall database design performance	
Query run / execution time	Allows us to gauge overall query performance	
Query errors	Allows us to identify errors quickly before issues occur	
Slow queries	Allows us to identify and optimise slow queries, if any are detected Enables us to find code paths / user actions that may trigger slow queries	Due to our performant indices, we expect fewer slow queries
Cache	Allows us to identify and improve database caching patterns, if required	

User Statistics










App Statistic	Purpose / Significance	Remarks
Listings per second	Provides a rough estimate of platform activity	
Videos played per second	Allow us to estimate usage and popularity of the video-based social media platform.	
Total users and total listings	Rough estimate of platform size	
Average chatbot rating	Allows us to measure the effectiveness of our answering chatbot and tune it as necessary	Users may rate the chatbot's answers in our webapp
Video like to dislike ratio	Allows us to identify potentially undesirable content	HUAWEI Cloud Content Moderation may be used to identify bad content as well

Database Storage Estimates

We have done calculations based on our database schema to come up with a meaningful estimate for the amount of database storage space consumed by our platform.

Entity	Estimate	Remarks
Listing (with 2 images and 1 video linking to it)	product listing: ~1500 bytes image references (excludes HUAWEI Cloud Object Storage): ~270 bytes * 2 video reference (excludes HUAWEI Cloud Object Storage): ~150 bytes total: ~2190 bytes	Average description (4096 characters) and title (128 characters) lengths are assumed.
User	~160 bytes per user entry	Average username length of 12 is assumed.
User data (likes, dislikes, bookmarks)	~38 bytes per entry	
Chat data (25 chat messages)	chat entry: ~80 bytes 50 chat messages: ~478 bytes * 25 total: ~12030 bytes	Average message (128 characters) lengths are assumed.

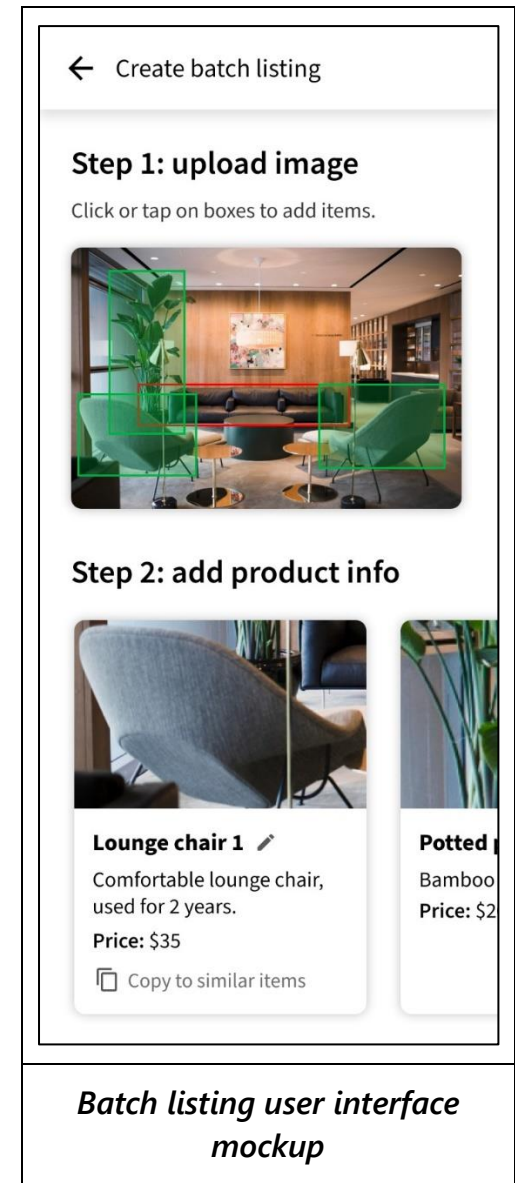
Usage of HUAWEI CLOUD Services

 HUAWEI CLOUD GaussDB for MySQL GaussDB for MySQL is the perfect database for our application due to its high reliability, flexibility and performance, and because our application prefers the consistent relational database model.	
 HUAWEI CLOUD Moderation Will help maintain our platform as a safe and peaceful community for our users.	 HUAWEI CLOUD CDN HUAWEI CLOUD's Content Delivery Network service will be used to accelerate our web application.
 HUAWEI CLOUD Object Storage Our apps' images, videos and other user-generated files will reside in HUAWEI CLOUD Object Storage.	 HUAWEI Media Processing Center User-uploaded videos will be transcoded to various resolutions to decrease loading times.
 HUAWEI CLOUD FunctionGraph We will use HUAWEI CLOUD FunctionGraph to power most of our API and backend.	 HUAWEI CLOUD API Gateway We will route API interactions to our FunctionGraph API using HUAWEI CLOUD API Gateway.
 HUAWEI CLOUD Image Tagging v2.0 User-uploaded images will be scanned for items for batch-listings using HUAWEI CLOUD Image Tagging v2.0.	 HUAWEI CLOUD Eye HUAWEI CLOUD Eye will be used to monitor our application's various aspects.
HUAWEI CLOUD ModelArts We will run various machine learning models on ModelArts for inference.	HUAWEI CLOUD Optical Character Recognition HUAWEI CLOUD Optical Character Recognition will be used to extract text from user-uploaded images, to save our users' time.

Features and Technical Implementation Details

- **Generic Text Summarization AI model - “CTRLSum”**
 - Given a ‘source’, takes two optional text inputs ‘query’ and ‘prompt’ and returns a text
 - ‘Source’ is the product listing description and OCR results from all listing pictures, unless both are the same (due to user using auto-fill function for description)
 - Deployed using **Huawei Cloud ModelArts** as a real-time service
 - Accessed whenever needed via RESTful APIs provided by ModelArts
 - Special Docker image will be created for this model containing the downloaded model weights
 - Model weights are approx. 1.6GB
- **Product Listing**
 - Information storage
 - **Huawei Cloud GaussDB**
 - **Huawei Cloud OBS**
 - Content moderation
 - **Huawei Cloud Content Moderation** on Text and Images (done after user finishes listing to reduce latency)
 - Listing title generation
 - Pororo’s ResNet+Transformer model generates a caption of the product image, which is then fed into CTRLSum for text summarization to improve precision
 - **Huawei Cloud Image Tagging v2.0**: identified object serves as ‘query’ for CTRLSum to aid in product title generation
 - Listing tagging
 - All levels of product category will automatically become tags e.g. Electronics, Smartphones, iPhone etc.
 - CTRLsum run with prompt “The keywords for this text:” as tags on the ‘source’
 - Tags are stored with other product information
 - Listing attributes
 - Predefined set of attributes for each category
 - CTRLSum with query “What is the <attribute> of it?” and prompt “The <attribute> of it is:” on the ‘source’

- User can choose to let CTRLSum auto-fill attribute fields or not
- Listing pictures information extraction
 - Using **Huawei Cloud Optical Character Recognition (OCR)** to extract all pieces of text from all listing images and store it in DB.
 - User can choose to auto-fill these extracted text as descriptions. Else, this information will still be used for CTRLSum related tasks as part of the 'source'.
- **Batch listing**
 - Premium feature unlocked only for paid users as it is computationally intensive
 - User takes ONE picture that contains multiple items to list
 - Uses **Huawei Image Tagging v2.0** to identify multiple objects in one picture and highlighting them on the front end by using the bounding box info returned by the API.
 - User selects items to be listed using checkboxes overlayed onto the picture, each checkbox will display the identified item class name.
 - Selected items will be cropped from the parent image and each cropped image (item) is sent for listing title generation, while being used as the cover picture of each listing.
 - While the titles are being generated, user can choose to type in description for the items to list. User can choose to copy the description for one item to all other items of same class e.g. chair A to all other chairs in the same picture.
 - Once user finishes inputting description for one item, he/she can trigger item attributes auto-fill on the UI, and similarly **synchronize them to all other items of same class**.
- **Product search**
 - Full and fuzzy text search on product title and description by GaussDB
 - Text search on listing tags generated by CTRLSum
 - Fuzzy image search by **Huawei Image Search** (user uploads image)
- **Product recommendation**



- Statistical tagging of users based on their recent browsing history and tags of the browsed items. Tags will be weighted by the frequency of opening listings that contains a certain tag. Browsing history includes both normal listings and short videos.
- Landing page recommended items initializes to be the items that received the most views in past 24 hours.
- Once user is logged in and is tagged, landing page recommendations will be the results of tag search using user's tags, ranked by weighted relevance (sum of user tag's weightage * tags present on listing).
- Each listing page will have a 'similar items' section where items are matched with relevance of their category and tags
- **Decided not to implement seller-to-buyer recommendation as in second-hand market context, sellers are not as important as the product itself, and chances of a seller being 'specialized' in selling a particular type of second-hand item is lower.**
- **Chatting**
 - Automated question-answering by CTRLSum: setting 'query' as question on the 'source'
 - Sellers can turn auto-answer on to reduce response time or off if they want to answer themselves
 - Using WebSockets for live communication between the seller and buyer
- **Videos**
 - **Huawei Media Processing Center** will be used for transcoding of videos of listings to various resolutions
- **Misc**
 - **Huawei Content Delivery Network (CDN)** will be used to accelerate delivery of all webpages
 - **React** with **Material UI** will be used for the front-end framework