Intelligent Reasoning Systems Project Report

Intelligent Image Recommendation System

Team Members:

Ma Xin A0261775W

Zhang Zhimu A0261649W

Sun Wenyuan A0261977M

Wang Zhiyuan A0261827Y

30-Oct-2022

Content

1. Abstract

A wallpaper is the first beautiful thing you will see upon booting up your computer, even before the annoying IMs of your manager. Thus a proper choice of wallpaper will definitely help you start your day.

In the information society, it is not an easy task to select a preferred and tasteful wallpaper, just as it is always hard to decide what for lunch (especially when your institute is on a hillside and there are no restaurants around). But there are so few websites focusing on recommending wallpapers or algorithms which can really learn from my preference of image composition and higher semantics. Thus, we are in demand of designing a wallpaper recommendation system that can meet the users' preference of images.

In this case, our group is dedicated to developing a whole wallpaper recommendation system with a frontend, a backend, a well performed algorithm and a database. The function of this system includes recommending wallpapers that the algorithm assumes that the user would like in the main page. Users can click into a certain wallpaper item to see the details like the tags of the image, liking, downloading or subscribing to this wallpaper. The subscribed wallpaper item would then be seen in the "my favorites" page afterwards. The system also provides a search bar where users can input the keywords to search for their interested tags or themes, clicking on the tags in the detail page would also do the job.

Each time a user clicks on "like" or "subscribe" at a certain wallpaper item, the system will rate for how the user likes this item, then records in the database. The more the system assumes that the user likes one item, the more it will recommend similar items to the user afterwards.

Our proposed image and semantic similarity based recommendation algorithm model has a wide range of application scenarios. Besides wallpaper recommendation, it can also be implemented on film or food recommendation fields, as long as the recommended item could be represented in a form of image combined with several tags or short descriptions.

2.Problem Introduction

2.1 Problem Introduction

Images are commonly used in life as a distinctive vehicle for the dissemination of information. With the rapid growth of social networks, a huge amount of online image data is being generated. Faced with a huge amount of image data, users want to quickly search for the image information they are interested in. An intelligent recommendation system capable of locating valid image information based on keywords provided by the user is a necessary tool for this purpose. An efficient recommendation system can greatly improve the efficiency and comfort of users and increase their adhesion and dependence on the product. This is why image recommendation systems are of great commercial value.

Image recommendation systems are already widely accepted and used. For example, on social media such as QQ and WeChat, users tend to communicate with others using image data called emojis that express their emotions in some way. The image recommendation system built into such social media helps users to quickly and accurately locate images of interest based on the search information they enter. However, due to subjective factors, it is difficult to fully characterize an image with the information entered by the user. This results in many recommendation systems not being able to locate the ideal image based on the user's search information. And the algorithmic logic of the recommendation system will also affect the accuracy of the recommendation function.

Based on this problem, we decided to develop an image recommendation system that can provide users with image feature labels to help them search effectively, improve the objectivity of the search information, and thus quickly locate the images they want. In addition, the recommendation system is capable of liking, favoriting and downloading. The system prioritizes images of the same type according to the type of image the user has liked and favorited, providing the user with more options to meet their needs.

2.2 Project Aim

Building an image recommendation system based on an image label similarity algorithm.

2.3 Project Objective

Apply Python to implement an image label similarity recommendation algorithm that enables a recommendation system to suggest images that users want based on their search information and like and favorite operations. Use React library of JavaScript to build front-end pages, including user login interface, registration interface, user search interface, image recommendation interface, image detail interface, favoriting interface, like, favorite and download functions. Use the Django library to build the back end of the system, interact with the front end for data, connect to the database and to the algorithmic model. Design the system's database to store image data.

3. Knowledge Modeling

Knowledge modeling includes the following aspects:

1. Requirement Analysis (Business Value)
2. Design
3. Implementation
4. Test

Requirement analysis includes the description of business value. The design part is divided into Data Acquisition, Data Cleaning and Knowledge Representation. The implementation indicates the workflow chart. The test part is providing a series of commands to test the operation result of the image recommendation system.

3.1 Requirement Analysis (Business Value)

The recommendation system has become the core function of social networking services. For example, in 2017 [1], Netflix has assessed that its recommentation system has a yearly business value of 1 billion dollars. There are 4 specific business values of applying recommendation engines:

1. User satisfaction: With years of research, it has indicated that users tend to look through recommended images from their last browsing and from like and favorite libraries. The recommendation system capable of using users' operation data will provide more suitable image selections, definitely helping and guiding their search activities. Therefore, high level user satisfaction enables valuable customer retention.
2. Personalization: The recommendation advice from family and friends is more acceptable because they are familiar with us and know what we like. Their advice is more believable. This is also the core concept of building recommendation models, which is also achieving personalization. The data accumulated from users can assist to improve the calculation capability of image recommendation systems for exact image presentation according to user's preferences. In return, users prefer to use the image recommendation systems and the corresponding main products with good mood.
3. User adhesion: Users normally prefer to accept recommendations which are similar to their preferences. When users find a recommendation website which satisfies their options extremely perfectly then they will definitely browse the website again.
4. Feedback: The recommendation systems capable of generating a particular image library for a specific user group will also generate a user group profile to reflect features of the user group. Then the creator of the recommendation system can produce more commercial functions and products to satisfy user's requirements, thus gaining revenue.

3.2 Design

3.2.1 Data Acquisition

The system needs a large number of images to build a reliable recommendation model, so we will use the scrapy framework in python to extract relevant data from the web page, such as images' name, tags and src.

Scrapy is an application framework written in Python for extracting website data and extracting structured data. Scrapy is often used in a range of applications including data mining, information processing or storing historical data.

The table below shows the web page where data is to be extracted.

Website	Data to crawl
https://wallhaven.cc	Images, images' name, tags and url. Each image has several tags

Table 3.1 Data resource website and description

Below figure shows the flow on how the extraction of data is done in our recommended system.

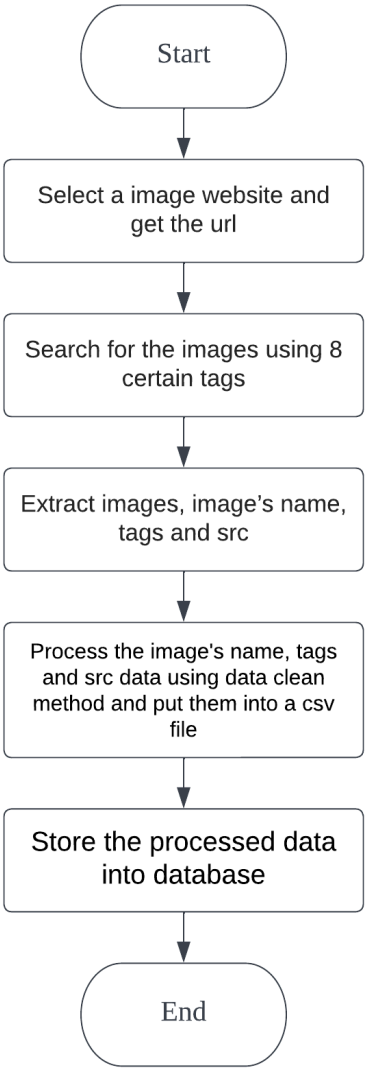


Figure 3.1 Data Extraction Flow

3.2.2 Data Cleaning

In order to determine the invalid and incomplete data and improve the quality of data extracted from the web page, data cleaning is needed in the preprocessing stage. Below figure shows the flow on how clean the data.

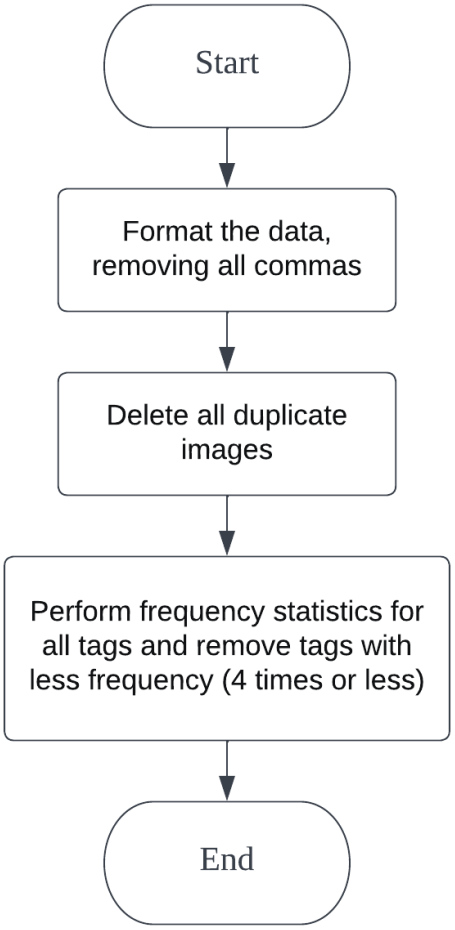


Figure 3.2 Data Cleaning Flow

3.2.3 Knowledge Representation - System Rules

The below table shows formal logic rules of the system.

LHS	RHS
If username and password is correct	Then jump to the main page
If username and password is not correct	Then show the message: Wrong username or password.
If search a tag in the search box	Then page will show all the images that contain that tag

If click one of the image in the page	Then jump to the detail page of the imagel
If click one of the tag in the detail page of the image	Then jump to the page will show all the images that contain that tag
If click star icon in the detail page of a image	Then show the message: Add to Favorite Successful or Remove from Favorite Successful
If click Favorite label in the menu	Then jump to the favorite page
If click heart icon or star icon in the detail page of a image	Then similar images will be recommended in the main page

Table 3.2 System logical rules

3.3 Implementation (Knowledge Model - Process flow and description)

The following flowchart defines the specific processes of using our image recommendation system.

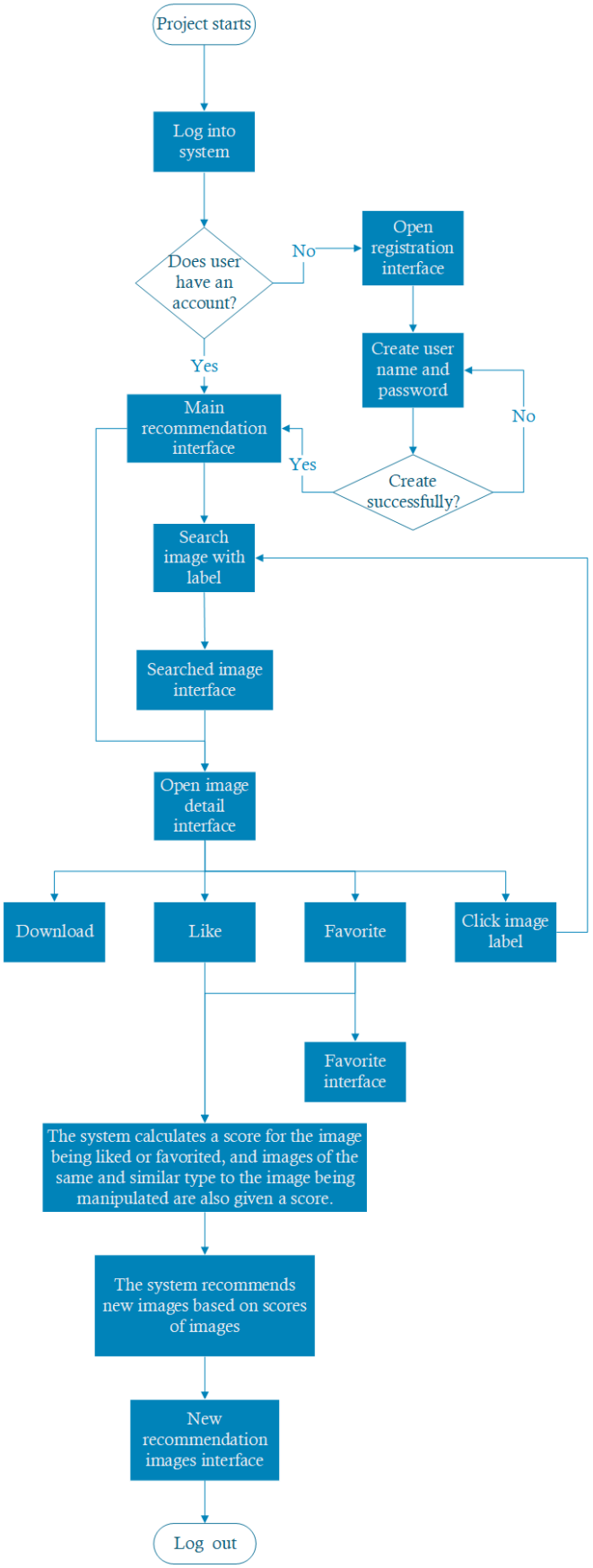


Figure 3.3 Implementation flow chart

3.4 Testing

After finishing construction of the frontend, backend and algorithm model, we try to define a series of commands to test the capability of our image recommendation system.

Test commands	Test results
Input user name and password being not permitted	User account fails to be created
Input user name and password being permitted	User account is created successfully
Input image label in search box	The searched image interface displays images successfully
Click one image in main interface	The page jumps into image detail interface
Click like and favorite buttons on image detail interface of an image	The image is liked and put into favorite library successfully

Table 3.4 Testing of system

4. Theoretical overview

4.1 Idea

We originally planned to use item collaborative filtering to do the recommendation, but soon we realized that item collaborative filtering won't suit our project well. Here are two main problems:

1. ItemCF will need a lot of user data collected from the website or created by our own, but either way seems too hard to achieve and time consuming.
2. ItemCF is mainly deployed in scenarios like product or movie recommendation, where the recommended items could come in many

forms and vary widely, just like the classic case “bear and nappies”. But in the wallpaper recommendation task, the recommended items are all in the same form of single image but variance in colors, compositions and objects, it is more like “bear and coke” compared with the previous case, thus the preference of certains users may not guide the others.

In this case, we decided to implement a content similarity based recommendation algorithm, which is recommending only based on the certain user’s preference rather than being guided by other users. Thus, though with little user data, the algorithm could still perform well.

4.2 Algorithm Details

Our content similarity based recommendation algorithm mainly takes into consideration the wallpaper image features and its label features, which enables it to learn from the user’s preference of image compositions of colors, objects and higher level semantics like whether it is artistic painting style or realistic style.

To achieve this, our recommendation algorithm is mainly designed in two parts, the image level similarity calculation and the semantic level similarity calculation.

4.2.1 Image level similarity

In the realization of this part, we decided to use the deep learning method to achieve the goal. After some considerations of the accuracy and the time cost for training, we choose the famous resnet50 to perform the task.

To make the sampling effect of this resnet50 model more consistent with the characteristics of our dataset, we train the net with pre-trained parameters on a multi-label classification task with our own wallpaper dataset. After the loss starts jittering in a small range, we can regard the model to have learned the characteristics of our own dataset and to be ready for inference.

We remove the fc layer of the resnet50, replacing it with a fully connected layer with an output dimension of 2048. Then we feed all the images to the modified

resnet50 to get the subsampled tensors. To calculate the image level similarity, we used cosine similarity as the measurements of how two images are similar to each other, and used a co-occurrence matrix to store the calculation results.

4.2.2 Semantic level similarity

The learning of semantic level of user preference is mainly based on the tags of the given wallpaper and it is carried out in a two phase procedure.

For the first step, we need to figure out the similarity between each tag. We assumed that the more frequent two tags appear together the more similar they are. Based on this hypothesis, we count the frequency of co-occurrence of each tag pair and use the co-occurrence matrix to record. Then we use the following formula to calculate the similarity between each pair of tags.

$$tag\_sim(i,\ j) = \frac{c(i,j)}{\sqrt{f(i)\ *f(j)}}$$

Here in this formula, *i* and *j* represent the two tags, *c(i, j)* represents the co-occurrence frequency of the two tags, *f(i)* represents the total frequency of tag *i*. The calculated result for each pair will be in the interval of 0 to 1.

After obtaining the tag similarities, it's time to solve the item similarity based on the given tags. Since every item has a list of tags of uncertain length, we decided to use the mean value of the highest similarity values between each tag of an item and all tags of the other item respectively to represent the similarity between two items. The result is also stored in the form of a co-occurrence matrix.

$$item\_sim(a,\ b) = \frac{sum_{i \in t_a}[max_{j \in t_b} tag\_sim(i,j)]}{length(t_a)}$$

Here in this formula, *a* and *b* represent the two items, *t* represents the tag list of an item. The calculated result for each pair will be in the interval of 0 to 1.

4.2.3 Collaborative similarity

The collaborative similarity is also represented using a co-occurrence matrix, the value is the average of the previous two matrices. After this we normalize the collaborative similarity co-occurrence matrix then we can use this to do the inference.

4.2.4 Recommend procedure

Upon inferencing, the algorithm will search through the database to get some rated items. Based on their rates, the algorithm will search the collaborative similarity matrix by the selected item to get some of the most similar ones and return. The rating of the item will affect the number of recommended items which are similar to that certain item, in other words, the more the user likes a wallpaper, the more the algorithm will recommend that kind of wallpaper to the user. If the user is newly registered, the algorithm would randomly assign some items and items that are similar to them and recommend them to the user.

4.3 Algorithm Limitations

As said before, the algorithm recommends only based on the preference of a single user and it disregards the similarity between user groups, so the recommendation is customized but isolated.

Besides, the algorithm is too simple that it will only consider the similarity and recommend only similar items to the user. So the user is hard to see something new and surprising from the mainpage, unless he starts to search for something he hasn't ever tried.

4.4 Algorithm Refinement

To overcome the first limitation, we could collect or create more user records and implement the item collaborative filtering method as a complement to the recommendation system, since the itemCF method is also using co-occurrence matrix as its main data structure (actually this algorithm is modified out of a framework of itemCF). And in this way, for the new sign ups, we can provide them with some of the most preferred wallpapers as initial recommendations rather than random ones.

As for the second limitation, we could add a few random recommendations every time the system recommends, so the user may not always be trapped in his recommendation cocoon.

5. Solution Outline

5.1 System Architecture

The whole system is built by python, using React in frontend, Django in backend, and build database with SQLite.

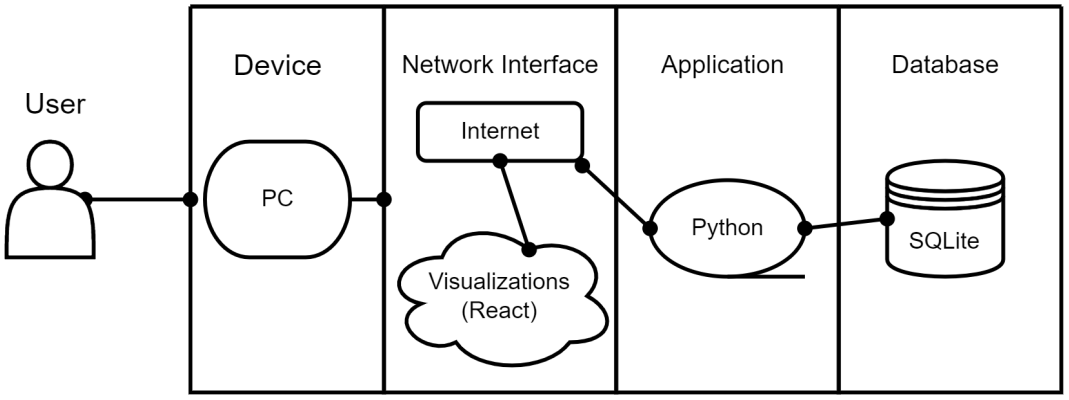


Figure 5.1 System Architecture

5.1.1 Database Construction

Database contains four tables, React_favorite, React_users, React_rating, React_wallpaper. React_favorite contains username and imagename. React_users contains password and username. React_rating contains username, imagename, favorite, like and rating. React_wallpaper contains imagname, image-source and image-tags.

At the initial login interface, the system will first randomly score 15 pictures and read the image names from the React_rating table by using the rec_init function. Second, according to the image names, the corresponding imagSRC of the pictures in the React_wallpaper table is read and output by using getItemInfo function, and then the front end will display the pictures in the order of score from high to low.

When doing the register, React_users table will record the username and following password by using the signin and signup functions. In searching, the system will use the search function we defined to search the tag which user input in React_rating table. Then the system will output the wallpaper with the following tag.

If the user likes the wallpaper, the system will add one point to 'like' in the React_rating table. If the user collects the wallpaper, the system will add one point to 'favorite' in the React_rating table. Next time the user returns to the homepage, it will recommend the user-like wallpapers by using the content based similarity recommendation algorithm.

5.1.2 Frontend Construction

We use React to build our front-end interface. React is a JavaScript library for building user interfaces. And there are two obvious advantages.

Declarative: React makes it painless to create interactive UIs. Design simple views for each state in our application, and React will efficiently update and render just the right components when our data changes. Declarative views make our code more predictable, simpler to understand, and easier to debug.

Component-Based: Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, we can easily pass rich data through your app and keep the state out of the DOM.

The figure 5.1 shows the whole pipeline of one use case.

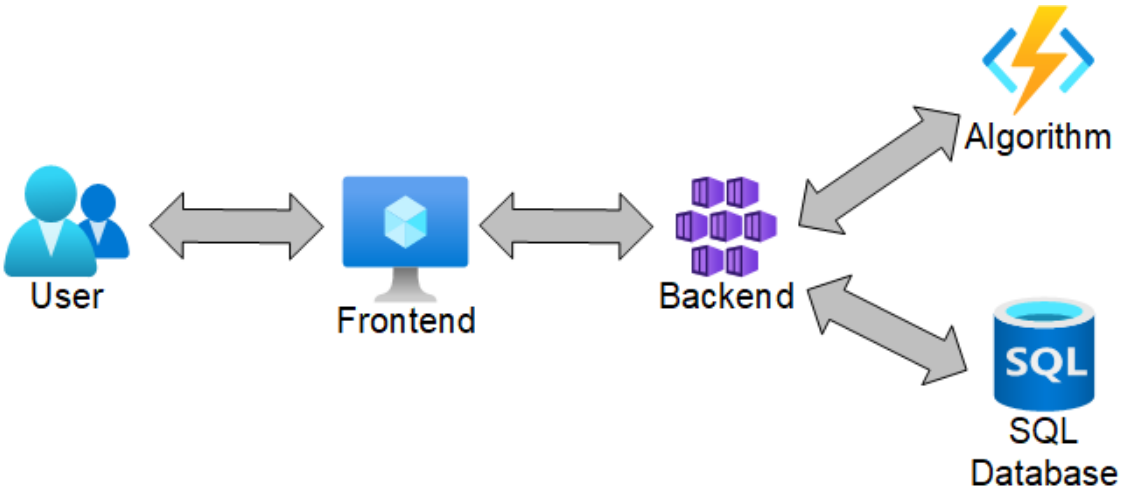


Figure 5.1 Pipeline Of Use Case

Front-end functionality consists of the following aspects:

Users need to register and login to access the main page. Users can also logout by clicking the logout label in the menu.

When the user searches for an image tag that interests them, the system will return the user images with the corresponding tag.

User can select an image and click it, then the system will enter the detail page about the image, which contains all the tags of the image, and the user can like or favorite the image by clicking the heart icon and star icon.

The user can also click one of a tag on the detail page, and the system will be redirected to a new page that contains all the images with that tag.

What's more, when the user clicks Favorite label in the menu, the page will enter the favorite interface and display all the images that the user has clicked the star icon in the detail page.

Most importantly, when the user returns to the home page, the page will display 50 images similar to those the user previously liked or favorited.

5.1.3 Backend Construction

The backend service is built with the Django framework which is a single Python process. First, create a django project. Second, create an app directory, which is used to process the user search and user information interface. Third, by

creating an API, it could get user information such as username, image, searching tags, image status(like, favorite).

API URL	DESCRIPTION
/log_in/	Enter the username and password to log in
/register/	Register username and password
/recommend/	Fifteen images were randomly recommended by the algorithm model
/search/	Index images by entering tag names
/getItemRatingDetail/	Get the username, show status if the user like or favorite the image
/like/	The status shows the user rating of the image
/favorite/	The status shows the user rating of the image
getFavorite/	Shows the favorite list

Table 5.1 URL Description

The backend service has following functions:

- Ø Receiving a request and package the data into work component
- Ø Integration with recommendation services (Connect to the algorithm model)
- Ø Provide API for front-end services
- Ø Connect to the database

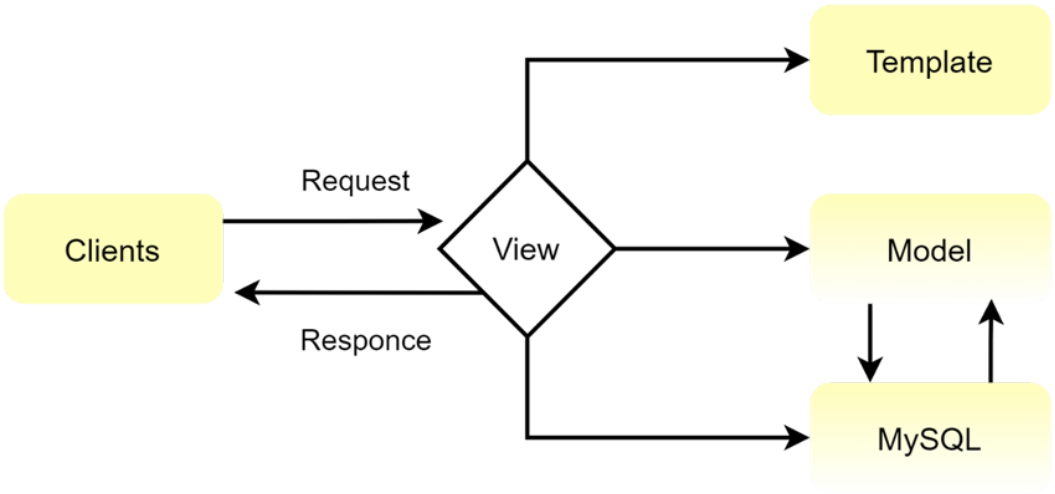


Figure 5.1 MVC Model

5.2 System Features

This personalized wallpaper recommendation system is web-based and you could simply use it by using any Internet-connected device that enters the URL. When you log in for the first time, the system will randomly recommend 15 images at the top. Then, the system will automatically recommend a list of wallpapers that the user mostly likes. Based on the images added to the Likes, Favorites list and the wallpapers rated by the content based similarity recommendation algorithm. By classifying the image types, users can browse a certain type of wallpaper more quickly. It also has fault tolerance Settings in the system, and when a username and password do not match, the user will be prompted to enter the correct user name and password or re-register. During the search, if you do not fully enter the name of the image by type, the wallpaper will be searched intelligently.

It is necessary to update the image iteratively. The wallpaper recommendation system stores the data in the database, which is convenient for information updating and has good scalability.

6. Conclusion

6.1 Overview

In this project, students put into practice what they have learned in class. In the process of practice, we also met many challenges in this process. However, we ended up using Python, HTML and SQL to build a complete Web-based recommendation system on the front end and back end.

6.2 Limitations

In general, the diversity and novelty of recommendations are very important to evaluate the effectiveness of recommendations, but there is a lack of in-depth analysis of the more complex relationship between the accuracy and diversity of recommendations.

In this system, the variety of wallpaper can be improved, insert more types of wallpaper and add adjective tags. The picture recommendation can be more accurate after adding more tags.

6.3 Future Development

1) Database expansion

The current database of the system is not rich enough,there is a lot of data that could be added. Not just static wallpapers, but also dynamic wallpapers could be included to provide users with more different choices. What's more, the current system only provides label information for images. If sharing and commenting functions can be added, the interaction between users will be increased and more people will be attracted.

2) Algorithm

In order to improve the accuracy rate, a variety of algorithms should be used to compare, and select a more appropriate algorithm. Even more, by combining various algorithms to achieve the best recommendation effect.

7. Reference

[1] 5 Advantages Recommendation Engines can Offer to Businesses. Maruti Techlabs[J/OL].5 Advantages Recommendation Engines can Offer to Businesses | by Maruti Techlabs | Towards Data Science.html, 2017-10-05

Appendix A: Project Proposal

PROJECT PROPOSAL

Date of proposal: **30/10/2022**
Project Title: **Intelligent Image Recommendation System**
Group ID (As Enrolled in LumiNUS Class Groups): **Group 3** **Group Members (name, Student ID):** **Ma Xin A0261775W** **Zhang Zhimu A0261649W** **Sun Wenyuan A0261977M** **Wang Zhiyuan A0261827Y**

Sponsor/Client: *(Company Name, Address and Contact Name, Email, if any)*

Background/Aims/Objectives:

Background

Images are commonly used in life as a distinctive vehicle for the dissemination of information. With the rapid growth of social networks, a huge amount of online image data is being generated. Faced with a huge amount of image data, users want to quickly search for the image information they are interested in. An intelligent recommendation system capable of locating valid image information based on keywords provided by the user is a necessary tool for this purpose. An efficient recommendation system can greatly improve the efficiency and comfort of users and increase their adhesion and dependence on the product. This is why image recommendation systems are of great commercial value. Based on this problem, we decided to develop an image recommendation system that can provide users with image feature labels to help them search effectively, improve the objectivity of the search information, and thus quickly locate the images they want. In addition, the recommendation system is capable of liking, favoriting and downloading.

Aims

Building an image recommendation system based on an image label similarity algorithm.

Objectives

Apply Python to implement an image label similarity recommendation algorithm that enables a recommendation system to suggest images that users want based on their search information and like and favorite operations. Use React library of JavaScript to build front-end pages, including user login interface, registration interface, user search interface, image recommendation interface, image detail interface, favoriting interface, like, favorite and download functions. Use the Django library to build the back end of the system, interact with the front end for data, connect to the database and to the algorithmic model. Design the system's database to store image data.

Project Descriptions

The project will consist of a frontend and a backend and content based similarity recommendation algorithm.

We use React to build our front-end interface. React is a JavaScript library for building user interfaces. And there are two obvious advantages.

Declarative: React makes it painless to create interactive UIs. Design simple views for each state in our application, and React will efficiently update and render just the right components when our data changes. Declarative views make our code more predictable, simpler to understand, and easier to debug.

Component-Based: Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, we can easily pass rich data through your app and keep the state out of the DOM.

The backend service is built with the Django framework which is a single Python process. First create a django project. Second, create an app directory, which is used to process the user search and user information interface. Third, by creating an API, it could get user information such as username, image, searching tags, image status(like, favorite).

Our content similarity based recommendation algorithm mainly takes into consideration the wallpaper image features and its label features, which enables it to learn from the user's preference of image compositions of colors, objects and higher level semantics like whether it is artistic painting style or realistic style. To achieve this, our recommendation algorithm is mainly designed in two parts, the image level similarity calculation and the semantic level similarity calculation.

Appendix B: Mapped System Functionalities against knowledge, techniques and skills of modular courses:

Modular Coursers	System Functionalities / Technique Applied
Machine Reasoning	**Knowledge Elicitation and Extraction:** Web crawling from websites, data cleanse. **Knowledge Representation:** Image downsampling to a 4096 dimension tensor via resnet50, label similarity, image similarity, collaborative similarity. **Rule based System:** Recommendation rules about how to select items to recommend based on the co-occurrence matrix.
Reasoning System	**Similarity-Based Reasoning:** Use cosine similarity to calculate image similarity. Use a content based similarity method to perform the recommendation task.
Cognitive System	**Cognitive System:** Use resnet50 to downsample the images.

Appendix C: Database table list

	id	username	imagname
	Search column...	Search column...	Search column...
1	1	admin99	wallhaven-4xrgyl.jpg
2	2	admin99	wallhaven-kww6z7.jpg
3	6	admin99	wallhaven-4l3x6y.jpg
4	7	mxxxx	wallhaven-9md3rx.jpg
5	8	mxxxx	wallhaven-nemgd8.jpg
6	9	mxxxx	wallhaven-0wvj8r.jpg
7	10	mxxxx	wallhaven-vmqjgp.jpg

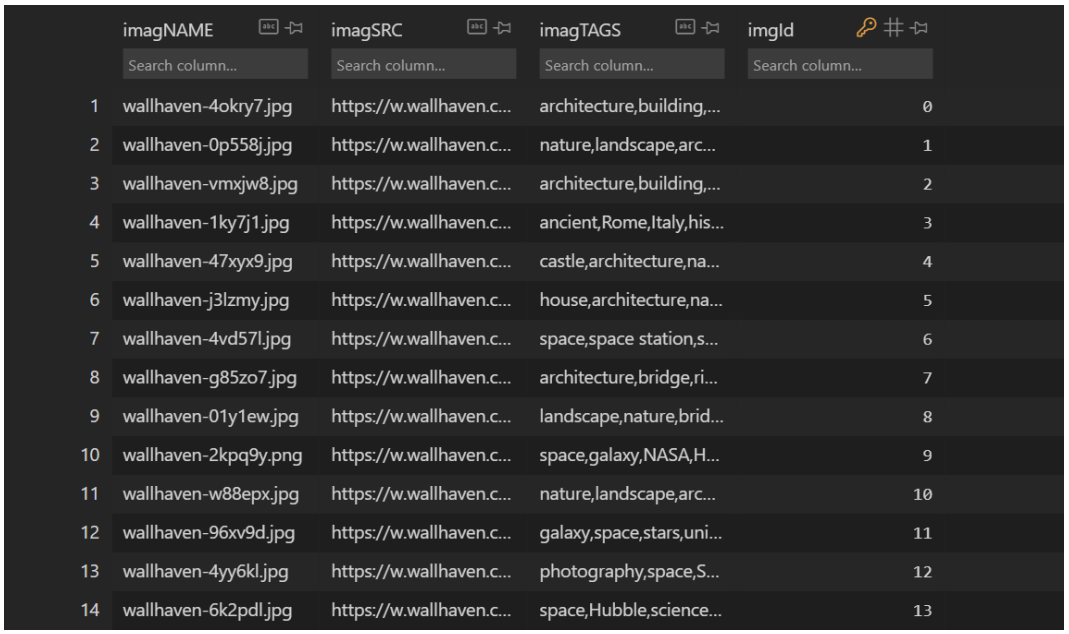
Figure a React_favorite

	id	username	imagNAME	fav	like	rating
	Search column...	Search column...	Search column...	Search column...	Search column...	Search column...
1	1	admin99	wallhaven-01mrq9.jpg	0	1	1
2	2	admin99	wallhaven-4oloyp.jpg	1	1	2
3	3	admin99	wallhaven-4xrgyl.jpg	0	0	0
4	4	admin99	wallhaven-kww6z7.jpg	1	0	1
5	5	admin99	wallhaven-0j2lxq.jpg	1	0	1
6	6	admin99	wallhaven-4l3x6y.jpg	1	0	1
7	7	mxxxx	wallhaven-9md3rx.jpg	1	1	2
8	8	mxxxx	wallhaven-xle17o.jpg	0	1	1
9	9	mxxxx	wallhaven-k99pr7.jpg	0	1	1
10	10	mxxxx	wallhaven-2kkz3m.jpg	0	0	0
11	11	mxxxx	wallhaven-1kp7x9.png	0	1	1
12	12	mxxxx	wallhaven-nemgd8.jpg	1	1	2

Figure b React_rating

	id	password	username
	Search column...	Search column...	Search column...
1	1	123123	admin99
2	2	1111	mmmx
3	3	1111	mxxxx
4	4	123123	admin88

Figure c React_users



	imagNAME	imagSRC	imagTAGS	imgId
	Search column...	Search column...	Search column...	Search column...
1	wallhaven-4okry7.jpg	https://w.wallhaven.c...	architecture,building,...	0
2	wallhaven-0p558j.jpg	https://w.wallhaven.c...	nature,landscape,arc...	1
3	wallhaven-vmxjw8.jpg	https://w.wallhaven.c...	architecture,building,...	2
4	wallhaven-1ky7j1.jpg	https://w.wallhaven.c...	ancient,Rome,Italy,his...	3
5	wallhaven-47xyx9.jpg	https://w.wallhaven.c...	castle,architecture,na...	4
6	wallhaven-j3lzmy.jpg	https://w.wallhaven.c...	house,architecture,na...	5
7	wallhaven-4vd57l.jpg	https://w.wallhaven.c...	space,space station,s...	6
8	wallhaven-g85zo7.jpg	https://w.wallhaven.c...	architecture,bridge,ri...	7
9	wallhaven-01y1ew.jpg	https://w.wallhaven.c...	landscape,nature,brid...	8
10	wallhaven-2kpq9y.png	https://w.wallhaven.c...	space,galaxy,NASA,H...	9
11	wallhaven-w88epx.jpg	https://w.wallhaven.c...	nature,landscape,arc...	10
12	wallhaven-96xv9d.jpg	https://w.wallhaven.c...	galaxy,space,stars,uni...	11
13	wallhaven-4yy6kl.jpg	https://w.wallhaven.c...	photography,space,S...	12
14	wallhaven-6k2pdl.jpg	https://w.wallhaven.c...	space,Hubble,science...	13

Figure d React_wallpaper

Appendix D: Installation & User Guide

Installation

For the front-end, we need to run the follow command:

Install node.js from the website https://nodejs.org/en/

npm install webpack webpack-cli

npm i -g create-react-app

npm axios

npm install antd --save

npm i react-router-dom

npm install sass-loader node-sass --save-dev

For the back-end, we need to run the follow command:

pip install numpy

pip install pandas

pip install django

pip install djangorestframework

pip install django-core

pip install django-cors-headers

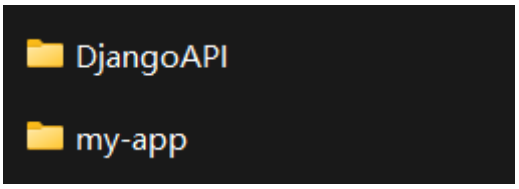
pip install matplotlib

pip install image

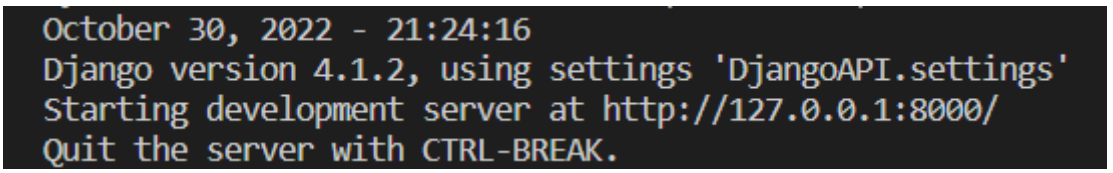
pip install pillow

All the libraries are the latest, no need to add a version for each one.

User guide



Use vscode or other compilers, open the DjangoAPI folder, then run command: **python manage.py runserver** to run back-end server.

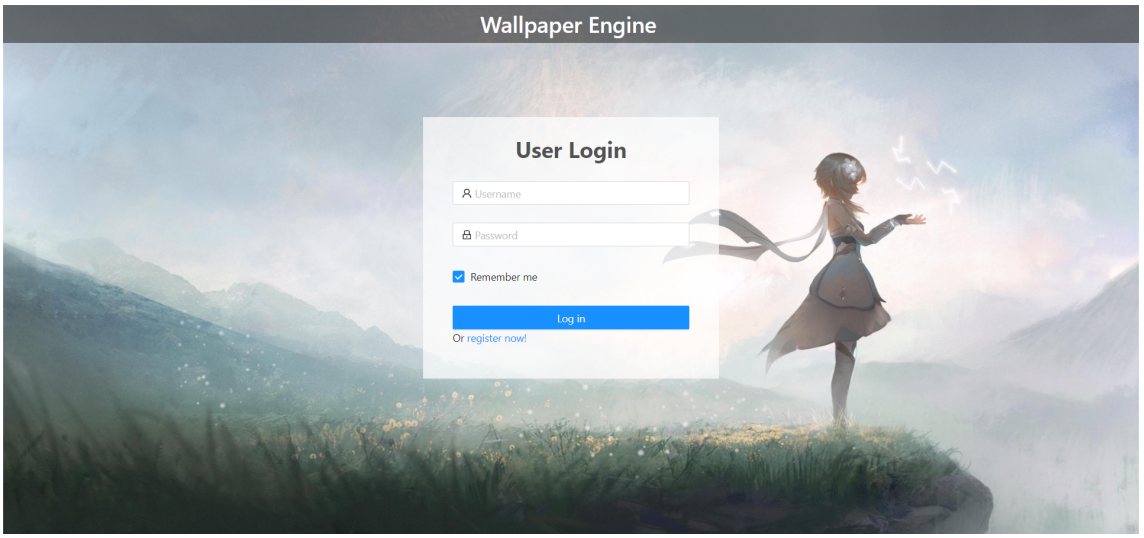


Use vscode or other compilers, open the my-app folder, then run command: **npm start** to run front-end server.

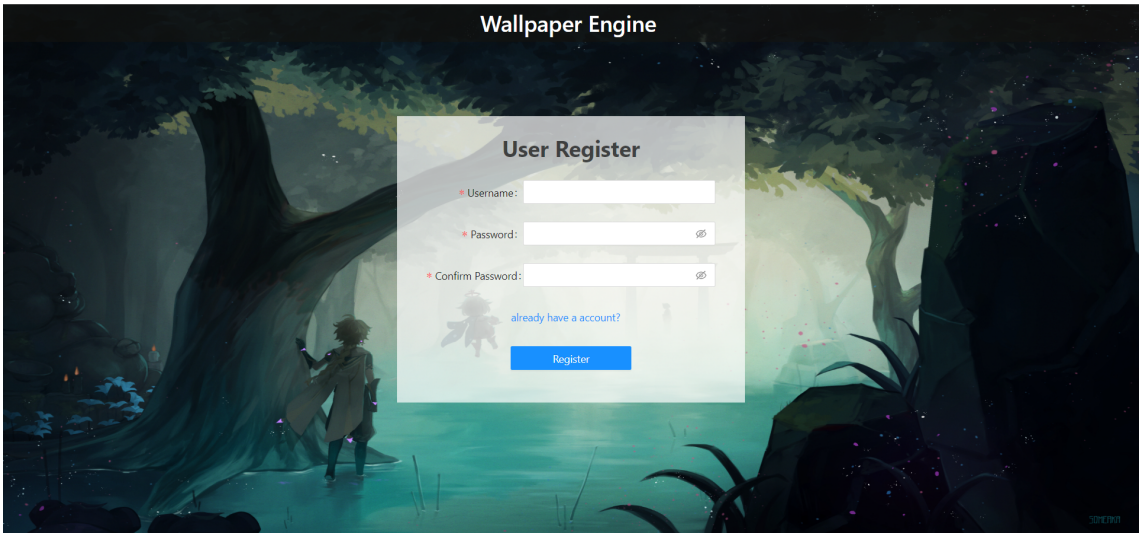
```
You can now view my-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use yarn build.

webpack compiled successfully
```

Enter the URL: http://localhost:3000

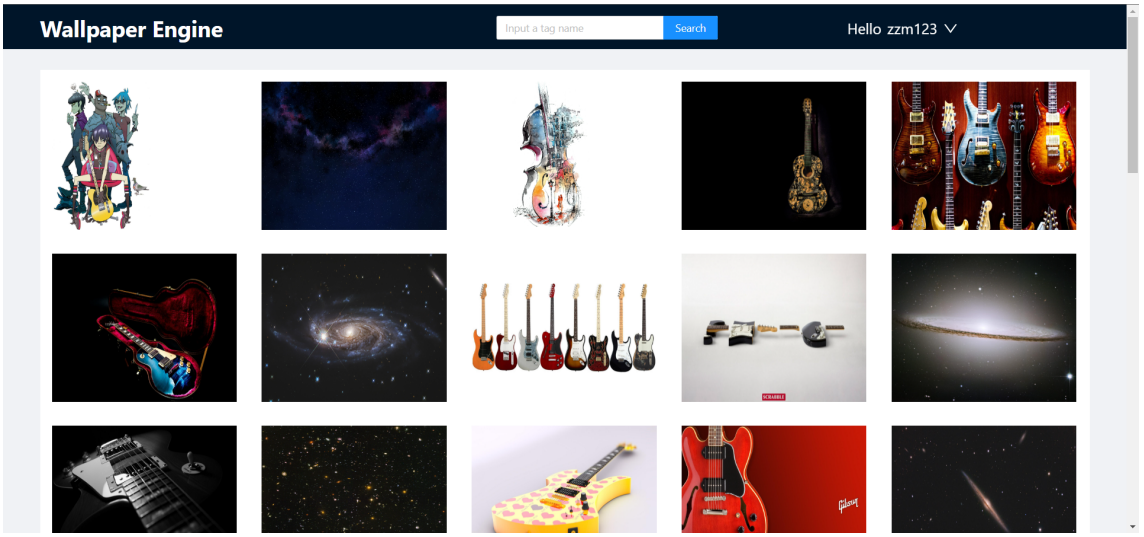
1. Click “register now”



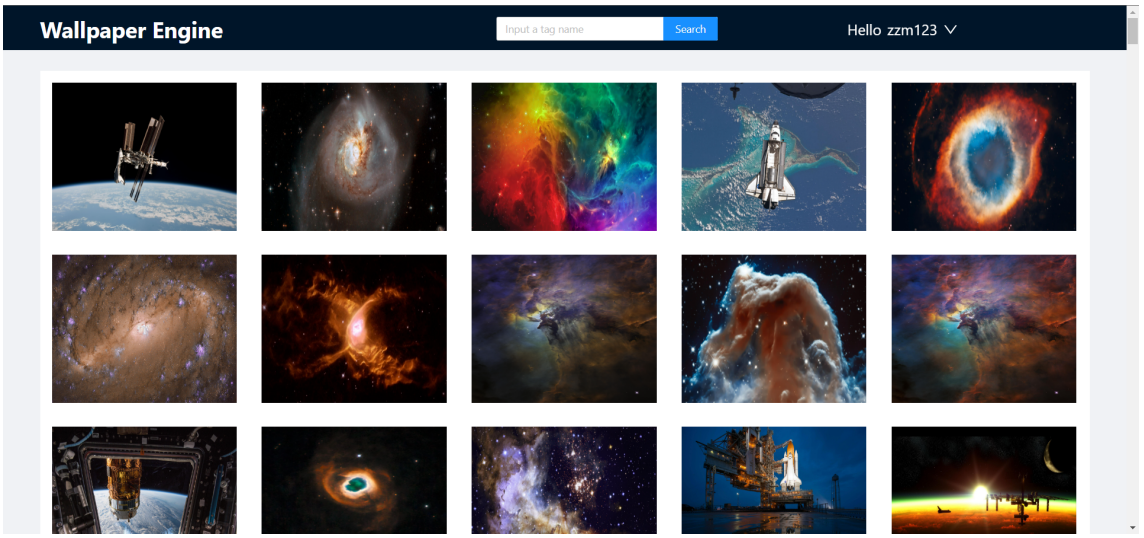
2. Register your account



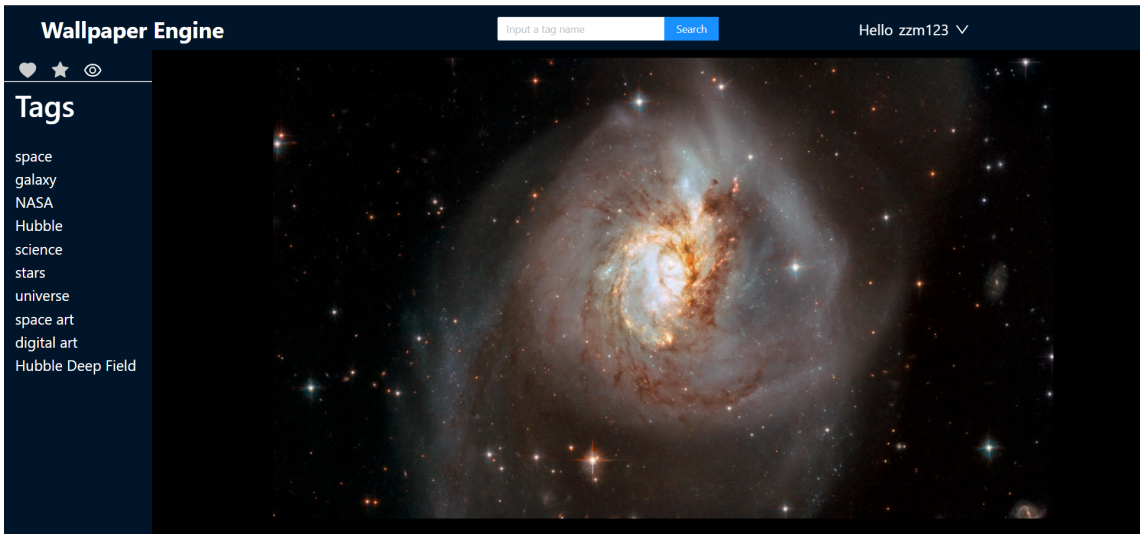
3. After registering as required, click the Register button to enter the homepage.



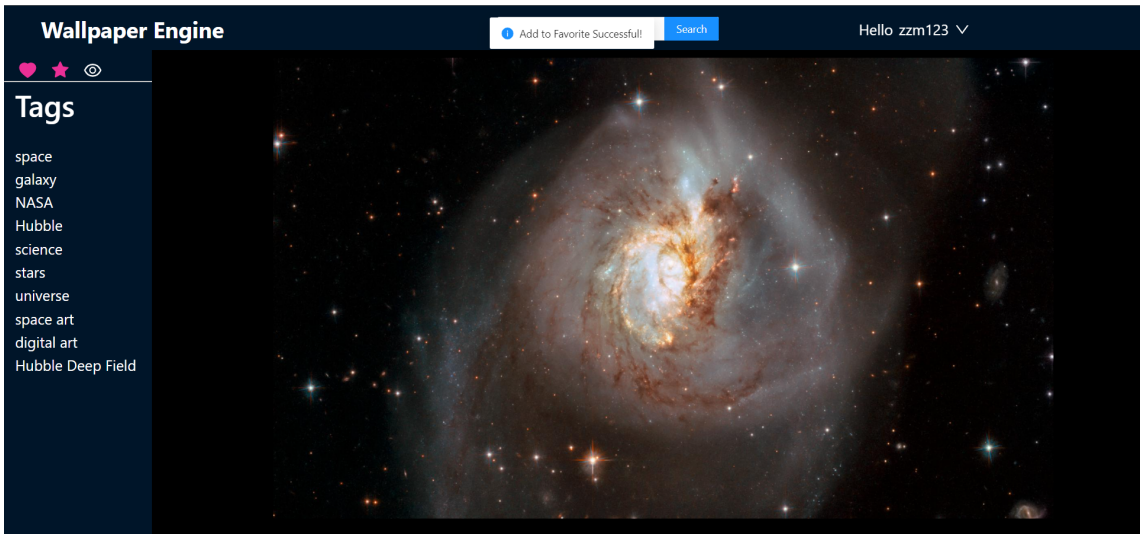
4. Search for an image tag that interests you (the below example tag is space), the system will return the user images with the corresponding tag. (The tags for space, food, art, tree, car, history, music are definitely exist in our system)



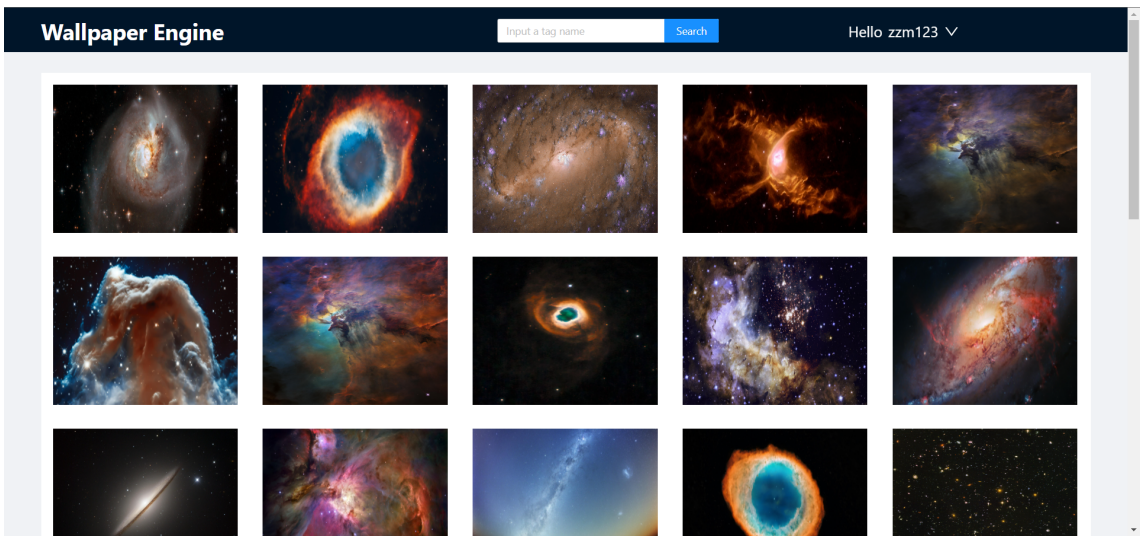
5. Select an image and click it, then the system will enter the detail page about the image, which contains all the tags of the image.



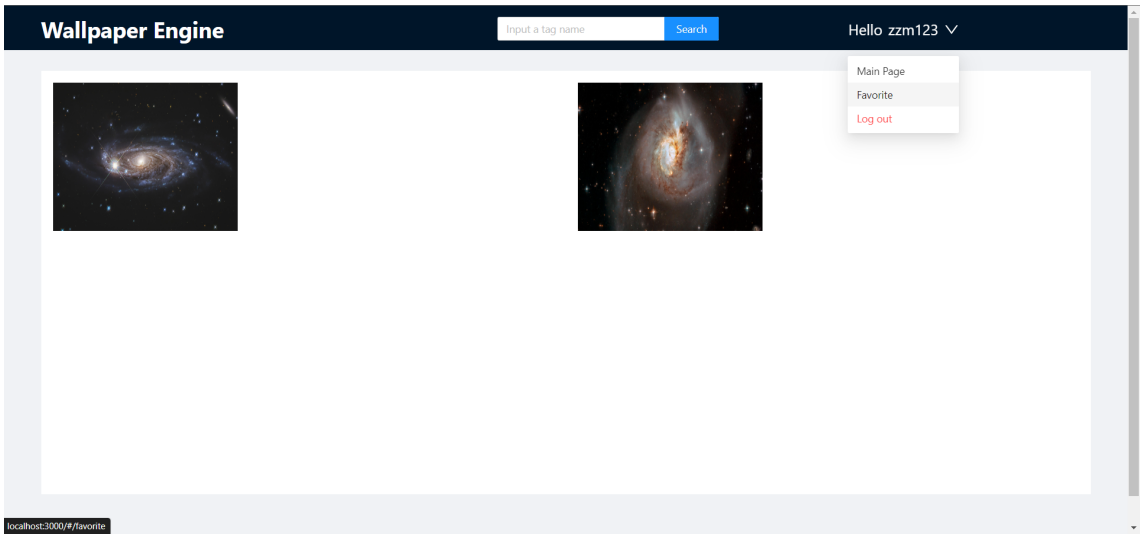
6. Like or favorite the image by clicking the heart icon and star icon.



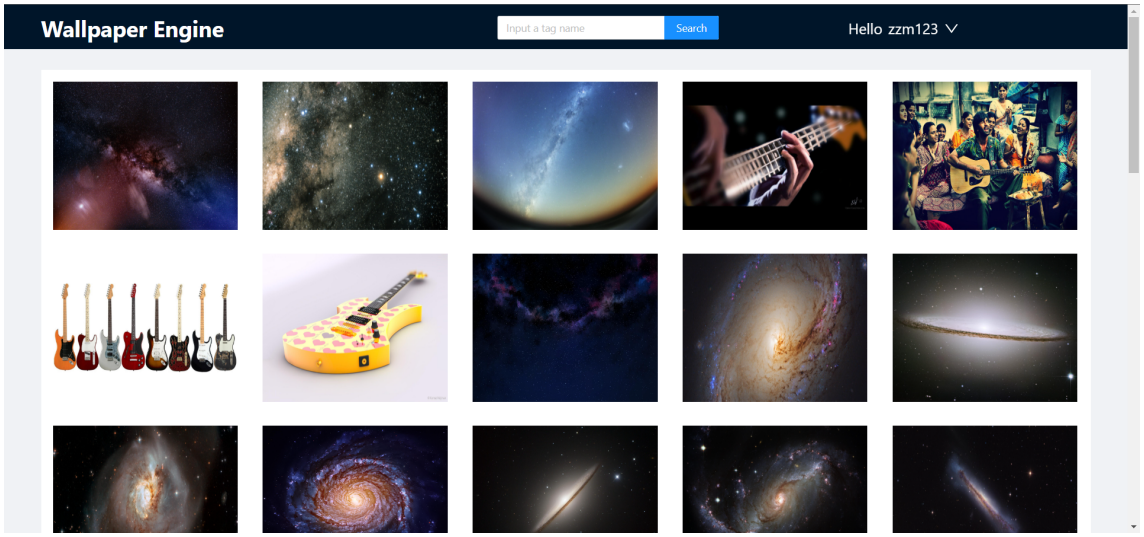
7. Click one of the tags from the tags list in the detail page can jump to the new page that shows all images that contain the tag. (The below example is click the Hubble tag in the tag list)



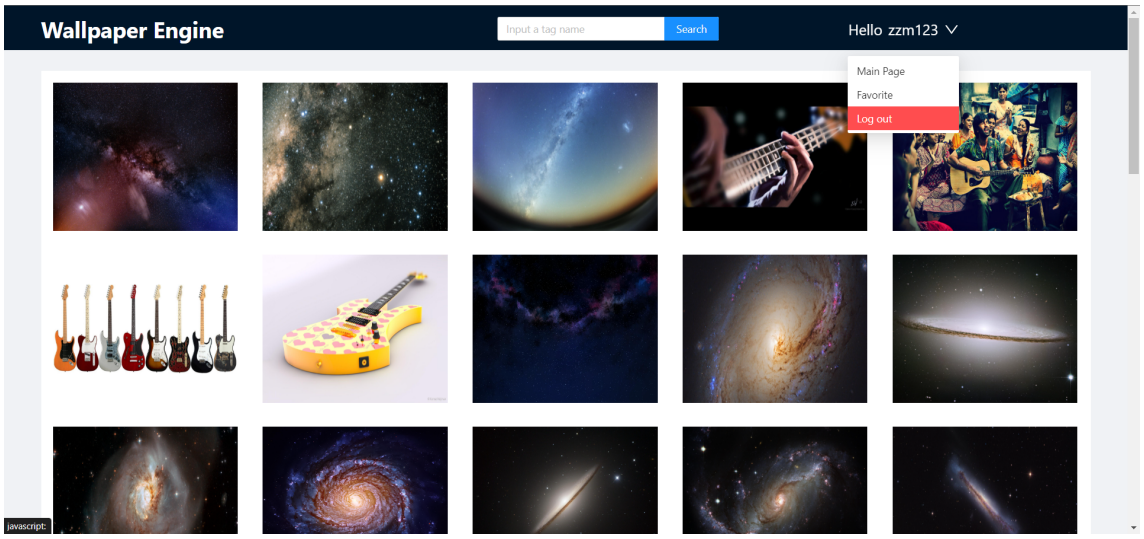
8. Click the Favorite label in the menu, the page will enter the favorite interface and display all the images that the user has clicked the star icon in the detail page.



9. Return to the main page by clicking the Main page label in the menu, the page will display 50 images similar to those the user previously liked or favorited.



10. Click the Logout label in the menu to log out.



Appendix E: Individual Report-Ma Xin

1. Your personal contribution to the project.

    1) Proposed the topic and idea, also responsible for task assigning and procedure following up
    2) Responsible for the algorithm model design and coding, providing algorithm APIs to the backend
    3) Participated in the backend coding and database design coding, providing few APIs to the backend guy

2. What did you learn from the project?

    1) Common recommendation algorithm methods and procedure
    2) Django development
    3) Database design

3. How to apply this to future work-related projects.

    1) Write this project experience into my resume
    2) Prepare for the interview
    3) Introduce this project and Q&A with the interviewer

Appendix F: Individual Report-Zhang Zhimu

1. Your personal contribution to the project.

In this project, I was responsible for building the entire front-end framework. I built the front end using React, and my job is to design web pages using React.js. First of all, I need to create the React scaffolding, which can make it easier and faster for me to write code. After that, I completed the login and registration page, where the user cannot access the main page without signing in. Many other features include APIs for postcalling, clicking buttons or labels to link to other pages, and other functions like searching, liking, and favorites. In the search function, all the pictures with the tag are displayed through the tag input by the user. After clicking the picture, the user can go to the detail page, which can display the detailed information of the picture, such as all the tags of the picture, and the user can also like and favorite the picture on the page. Users can also click the Favorites label on the menu bar to go to the Favorites page, which displays all the labels they have already favorites.

2. What did you learn from the project?

Cultivated practical abilities for front-end development; Become more proficient with React; Improved self-study ability; Strengthened communication skills and cooperation ability.

3. How to apply this to future work-related projects.

In the process of working for the company in the future, I can continue to design and implement more personalized functional websites. When working with colleagues on projects, being able to communicate with them more effectively can improve work efficiency.

Appendix G: Individual Report-Sun Wenyuan

1. Your personal contribution to the project.

In this project, I was responsible for building the back-end framework. The back-end framework we use is Django. Django is an open source MTV style web application framework powered by the python programming language. I created four classes with the model: wallpaper, users, rating and favorite. wallpaper stands for picture name, picture resource index, and picture tag recommended on your website. For the user class, it has two fields: username

and password, and for the rating class, it has five fields: User, Image, Like, Favorite, and rating. For the favorite class, it also has two classes: user and image. The most important part of the framework is the view class, which handles the requests sent by the front end. The back end processes these requests and returns the corresponding values to the front end. First, I used an index function to display all the default recommended images on the home page, showing the title and type. Next I use the search function to display the details and status of the image. In this feature, I processed two post requests. The first is to write a function to add images to a tag-list. The other is to return the user's rating (likes, favorites) for each image.

2. What did you learn from the project?

Firstly, I gained some experience in web development. The back-end framework is an important part. Secondly, I learned how to use Python's Django framework to build the back end and how to interact with the front end to receive requests from the front end and define functions to process requests and successfully return data to the front end. Through the practice of this project, I became more familiar with the whole framework and HTTP protocol, and consolidated and improved my basic computer ability. By writing code, my programming ability has also been improved, which is of great help to my future work.

3. How to apply this to future work-related projects.

First, Django, our back-end framework, is a very popular framework and an open-source model powered by the high-level python programming language. With this architecture, programmers can easily and quickly create high-quality, easy-to-maintain database-driven applications. Therefore, this project strengthened my understanding and mastery of this framework, which will be of great help to my future work. Secondly, in the process of completing this project, my programming ability has been improved to a certain extent.

Appendix H: Individual Report-Wang Zhiyuan

1. Your personal contribution to the project.

In the project, I was responsible for frontend, database design and coding APIs that interact with the database to the backend side. Besides, I still help to construct frontend. Frontend is the important part for the project because it needs to communicate with users and tries to satisfy users' requirements. React is used to build the frontend because it is very convenient and effective. I have tried to help Zhang Zhimu build the search page to allow users to input text. And the designed front end will transfer requests to the back end. Apart from that, the database is also my job. The designed database contains React_favorite, React_users, React_rating, React_wallpaper. React_favorite contains username and imagename. React_users contains password and username. React_rating contains username, imagename, favorite, like and rating. React_wallpaper contains imagname, image-source and image-tags.

Besides, I am also responsible for the report construction. I design the framework of our report and assign writing tasks to members. In addition, two videos are also my assignments. I designed the videos to indicate our project.

2. What did you learn from the project?

Learned about database design and django development. I met many new challenges during the period of database design. But with the help of my team members and online materials, I have solved them successfully. Besides, I also learned how to use react. It is based on the javaScript language. It's a new language for me, I'm glad to know it.

Learned about group development of software.

Learned the report writing and video design.

3. How to apply this to future work-related projects.

This experience has strengthened my understanding of basic database designing rules and django development, which is valuable for future work. The JavaScript language with React is also very useful for me to design front end pages in the future.