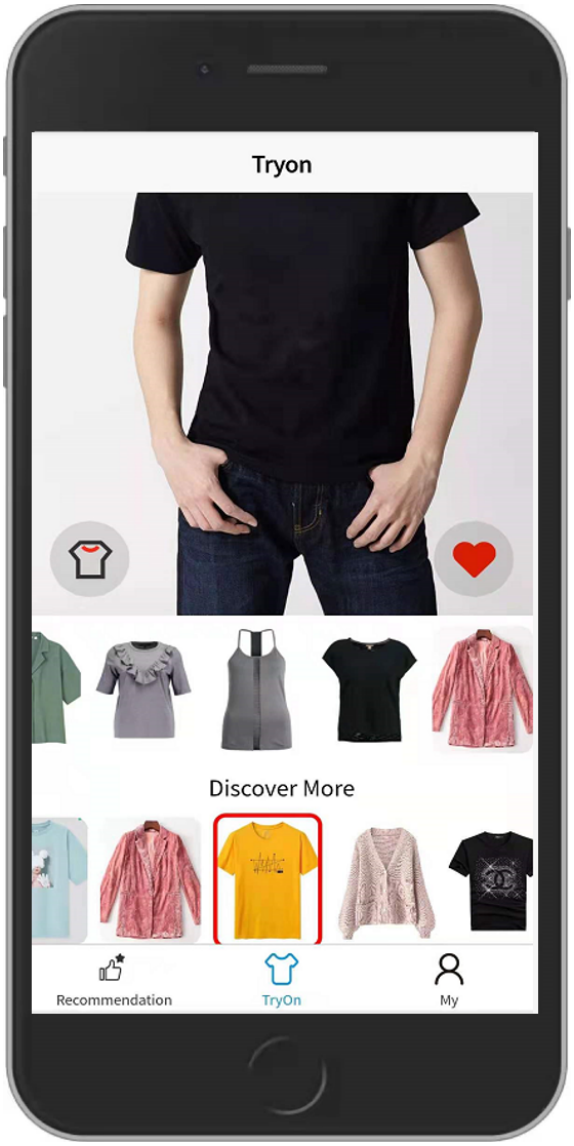
AnyFitting: A Fancy Virtual Try-on Application

- CSC4001 Final Project Report -



Chongjie Ye (118010378)
Ruiyu Gao (118010074)
Xinfei Li (118010151)
Longxiang Li (118010144)
Ke Li (118020231)

Contents

1. Abstract

Online apparel shopping has gained huge popularity on account of its huge advantages compared to traditional offline shopping, such as less time, more choices, and lower price. However, a large number of consumers regret the unfitness of clothes bought online while they cannot give the try-on feedbacks inconvenience. There are some existing commercial platforms to break the obstacles between online shopping and try-on experiences such as Cloudream, Gooda, and Meida. However, they have some fatal defects in modeling, customizing clothes, huge computational cost, and rare commercial opportunities. we propose AnyFitting, a virtual try-on app that provides a fancy try-on experience for customers and online warehousing for business users. AnyFitting aims to build a solid communication bridge between individual consumers and business companies. For individual users, AnyFitting provides users with virtual try-on function and outfit recommendations. For business users, AnyFitting is an intelligent data assistant on database management, data analysis, and data visualization. Overall, AnyFitting shows significant superiority in the state-of-the-art algorithm, customized try-on experience, low threshold, and economical computational cost. It has a relatively high possibility to be of commercial usage.

2. Background and Motivation

Recently, online shopping has become one of the most popular topic in all aspects of life with its huge traffic. According to a survey conducted in 2021, the number of digital buyers has reached 2.14 billion, which makes 27.6 percent of the 7.74 billion people in the world. Among all the online sales, the amount of fashion-related sales are distinguished. In 2021, sales on fashion constitute 30.4 percent of total online sales in the US. This verifies the popularity and great commercial potential in online fashion marketing.

As distinguished as the priority of online shopping are its weaknesses. Remote scanning and imagination cannot be on a par with physical fittings. An apple in the painting is never an apple in the hand. This causes a higher refund rate (which may account for 40 percent of all the refunds) and worse shopping experiences. In order to make up for the missing experience, virtual fitting systems are implemented.

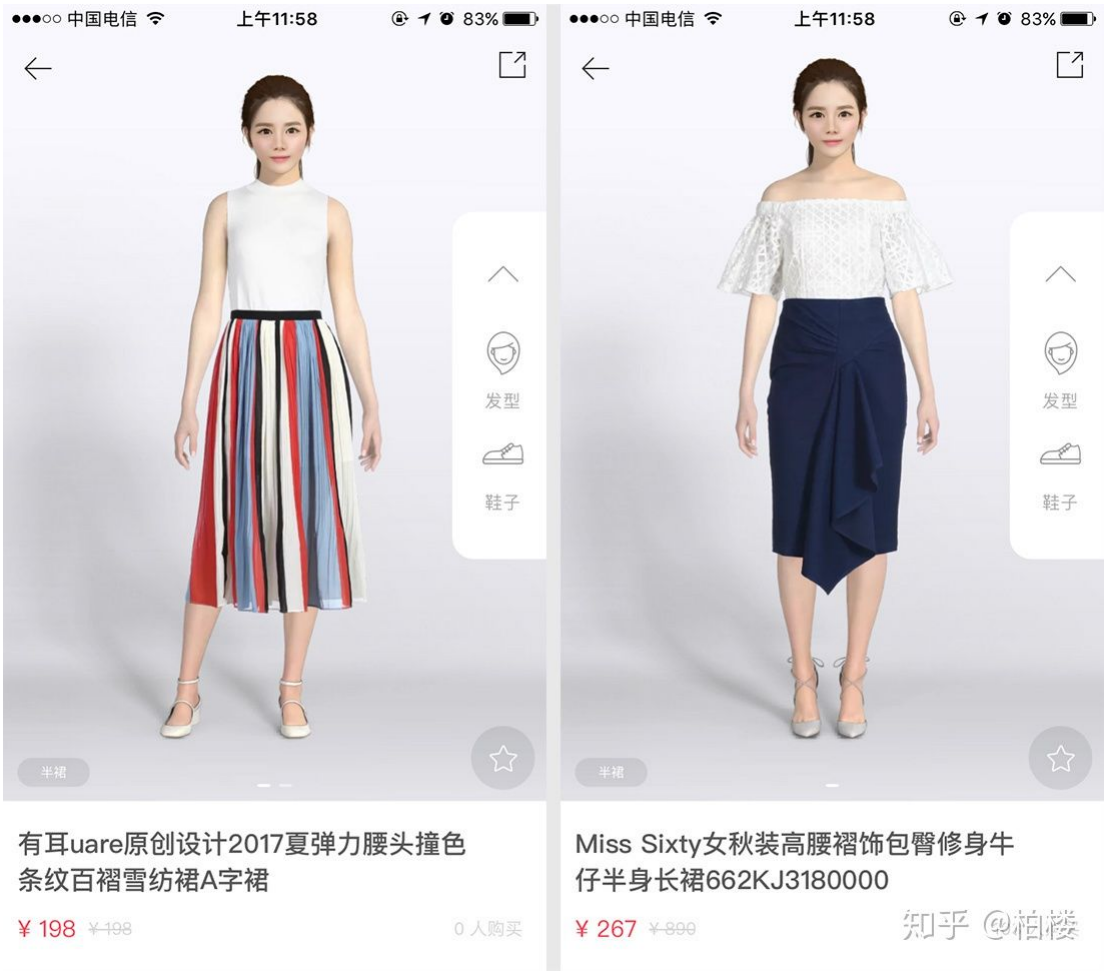


Figure 2.1: Application 1

Honestly speaking, this is not a brand new concept. The concept Virtual Fitting Room is firstly proposed on the International Forum on Science and Technology. It combines Augmented Reality, Virtual Reality and 3-D Calculation Picture to generate omnidirectional 3-D visual effect. In 2006, Face72 occurs as the first online shopping website with virtual fitting. However, due to its poor capacity in customization, it is more like a flash dressing up game rather than a powerful shopping guild. In the recent years, shopping platforms (such as Amazon, JD) and fashion brands (such as UNIQLO, Nike) are successively providing virtual fitting serves. Most of them enable

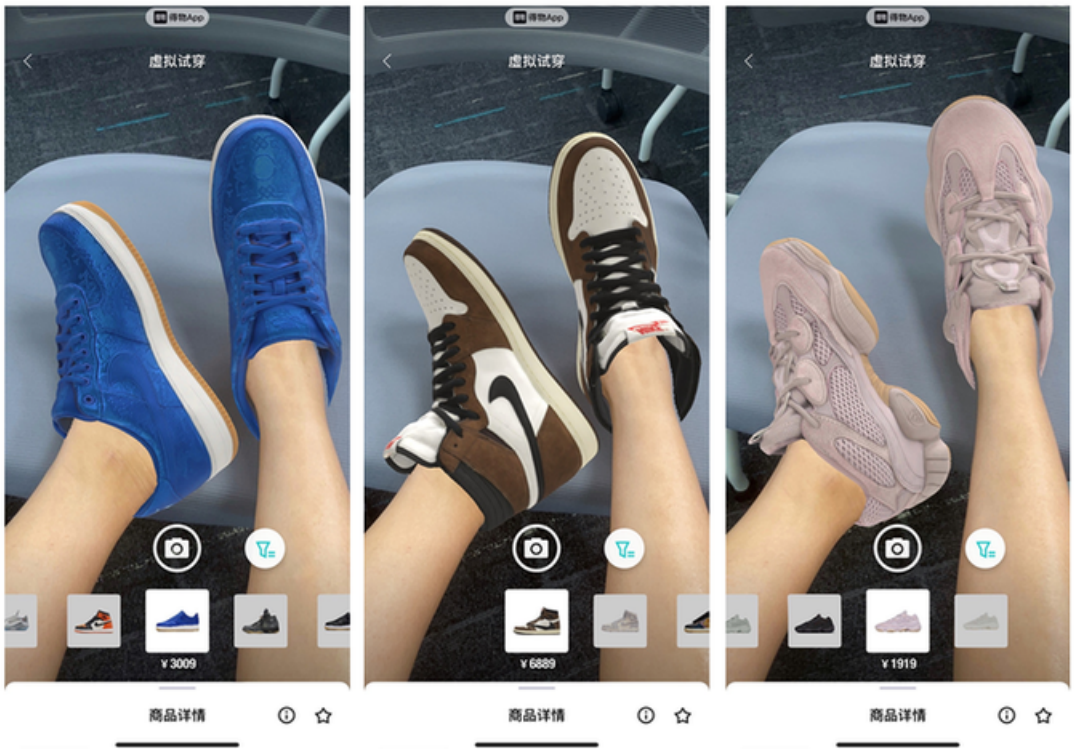


Figure 2.2: Application 2

the user to build the 3-D images by adjusting a set of body indexes. The virtual fitting system named Outfit-VITON implemented by Google is a little bit different. It utilizes the Generative Adversarial Network. Keeping the contour of user's body unaffected, the appearance features such as product texture, logo and embroidery are retained to the greatest extent.

Although the concept of virtual fitting seems perfect, lots of problems arise in practice. One is the figure customization. Most models on the virtual fitting system supports adjustment, but they are always cartoon images with fixed gestures. The effect of the cartoon models trying on cartoon clothing images is quite different from what is in the real life. The other is computational cost. Take Outfit-VITON as an example. a single try-on operation takes at least 5 seconds or more. In this case, commodity selection and comparison will be a extravagant hope. What's more, due to the computational cost, it takes amounts of money and several days for a merchant to model a single garment. It will be a great load for small businesses, and the user can only try on existing clothing rather than self-uploaded ones.

Considering all the issues above, we proposed our virtual fitting product, AnyFitting. For the practicability and versatility, our product prefers to applying the image processing algorithm, which is based on the algorithm of Outfit-VITON. The efforts are put into the optimization in both time and imaging effect.

3. User Document

In this chapter, we will describe our functions from two types of user view, which is personal users and business users respectively.

3.1 Personal User

- Register and login

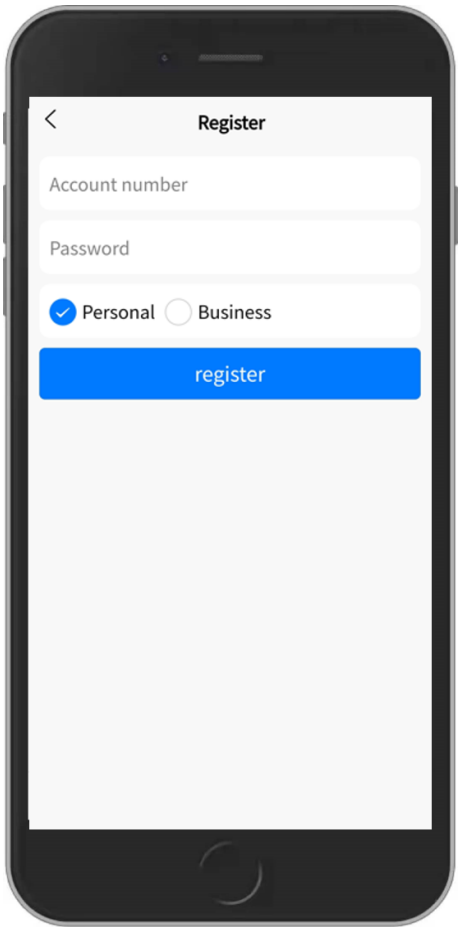


Figure 3.1: User Register

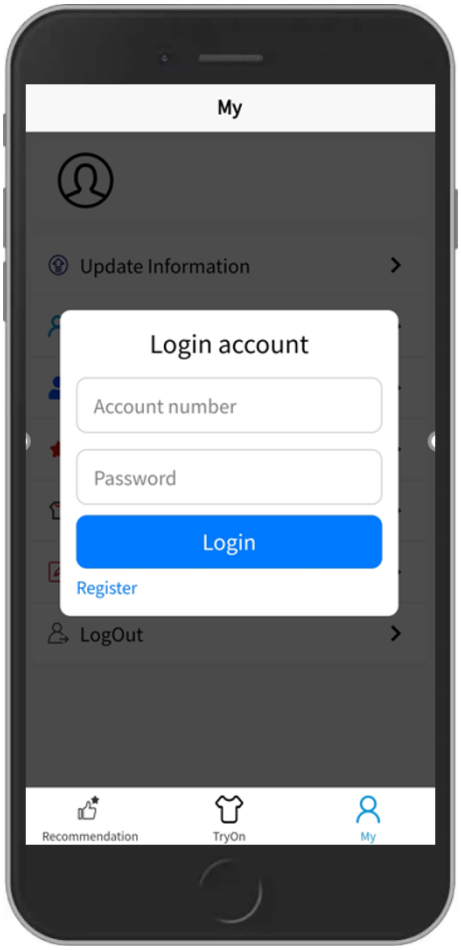


Figure 3.2: User Login

Users are required to register an account at the first time to use AnyFitting as shown in 3.16. Personal Users should choose "Personal" type when he/she registers an account. The account name support both english latters and latin numbers. The password is at least 6 bit for the sake of security.

After users register, users can login the account as shown in 3.17.

- Update Information

Users can modify their information after registering and login the account as shown in 3.3 and 3.4.

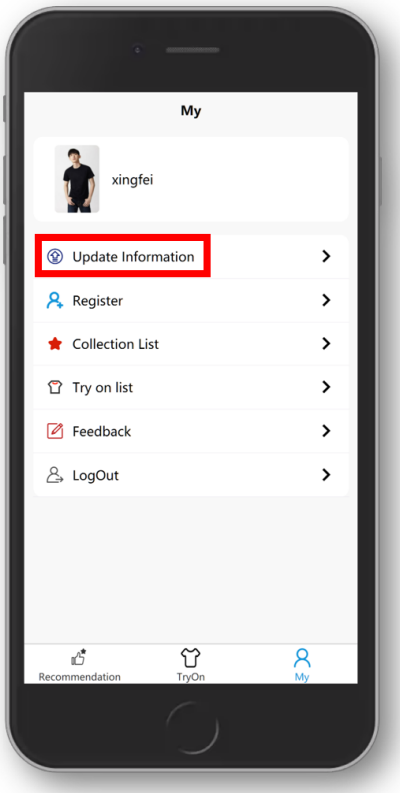


Figure 3.3: Update Information

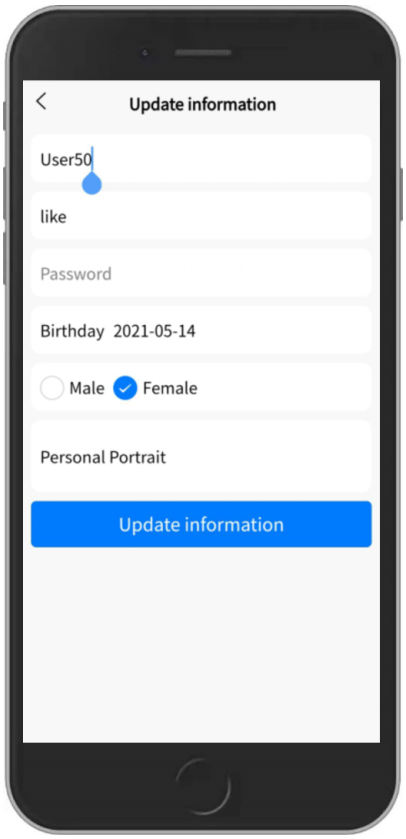


Figure 3.4: Update Information

- Upload Clothes

  After users login, users can upload their own clothes by clicking the uploading button in the left side as shown in 3.5 and 3.6. Users can choose either upload from file or upload from camera.

- Virtual Try-On

  After Users upload their clothes, users can click the try-on butthon in the left side to virtually try on the selected clothes as shown in 3.7. The try on result is shown in 3.8.

  Besides DIY uploading their own clothes, users can also try on the clothes uploaded from business users in the "Discover more" column.

- Like a Try-On

  After trying on some clothes, users may have preferences for certain clothes. Users can click the "Like" button in the right side as shown in 3.9. The users can click the "Collection List" as shown in 3.10 to see the preferred list. This list records all the clothes the user once liked. The example result is shown in 3.11

- View Try-On History List

  After trying on some clothes, users may want to view what clothes they once tried on. Users can click the "Try On List" button as shown in 3.12 to view the history of try on list. This list records all the clothes the user once tried. The example result is shown in 3.13

- Daily Personalized Recommendation to Users

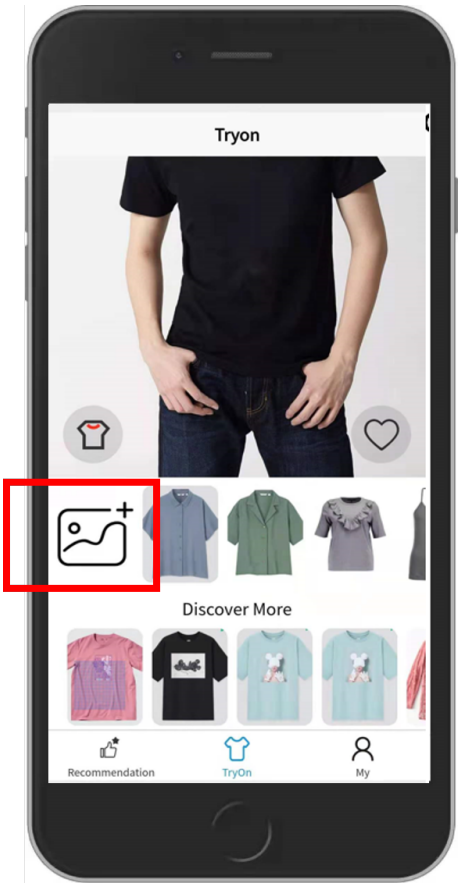


Figure 3.5: Upload Clothes Click

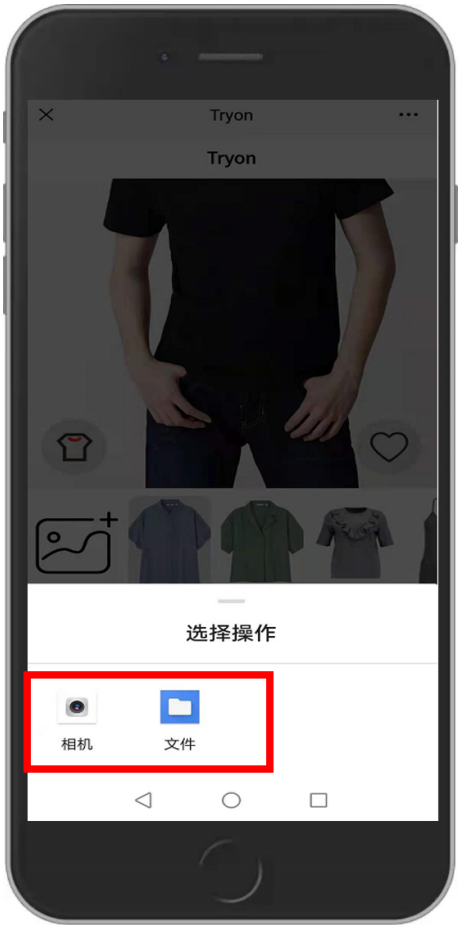


Figure 3.6: Upload Clothes Selection

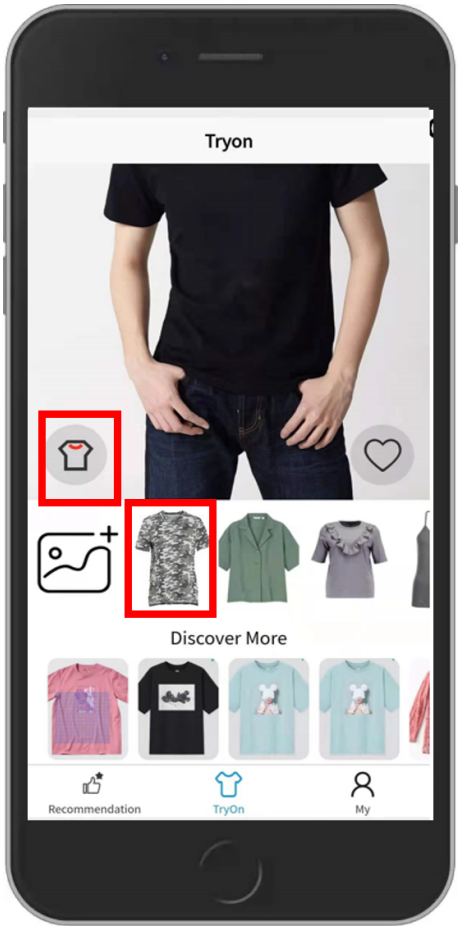


Figure 3.7: Virtual Try-On Selection

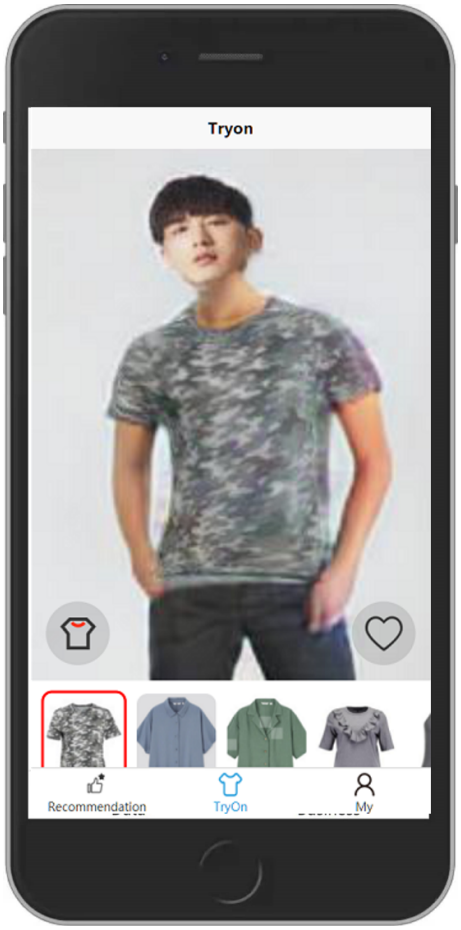


Figure 3.8: Virtual Try-On Result

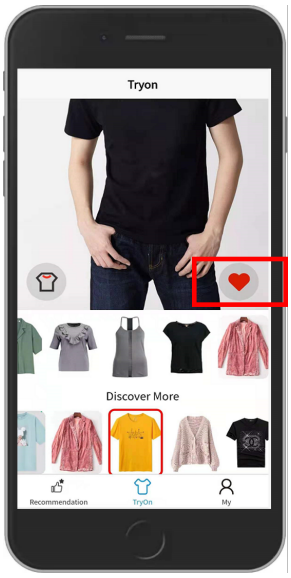


Figure 3.9: Like a Try-On

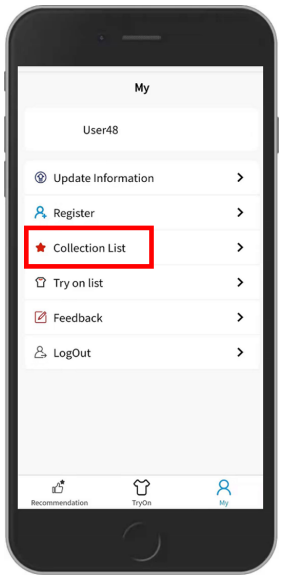


Figure 3.10: Collection List Click

Figure 3.11: Collection List

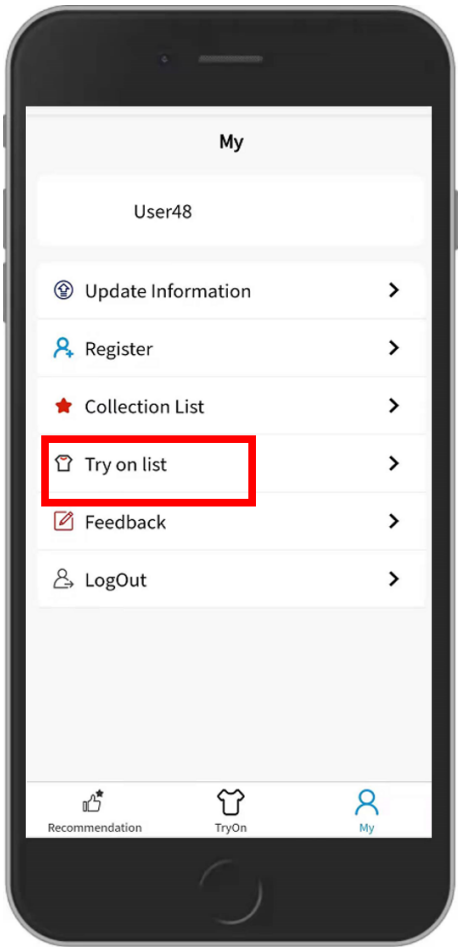


Figure 3.12: History Try-On Click

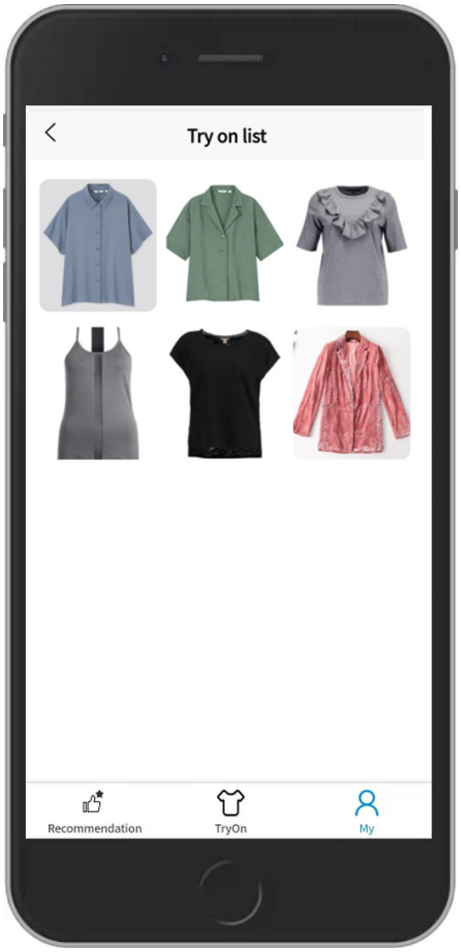


Figure 3.13: History Try-On List

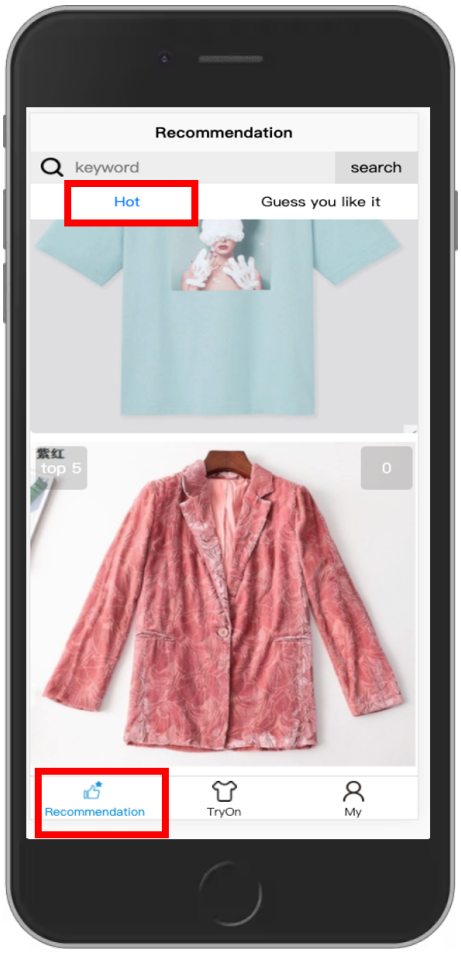


Figure 3.14: Popularity Recommendation

Figure 3.15: Personalized Recommendation

This is one of the most fancy functions of our application! Users can click the recommendation button in the left down side. We have two types of recommendation. One is called "hot" as shown in 3.14, which would recommended the most popular clothes daily. The popularity is measured by how many times this clothes are "liked" by users. Another type of recommendation is "Guess you like it" as shown in 3.15, which would recommend the clothes that particular user may like. This personalized recommendation is based on the users like list and history try on list.

3.2 Business User

- Register and login

Users are required to register an account at the first time to use AnyFitting as shown in 3.16. Business Users should choose "Business" type when he/she registers an account. The account name support both english latters and latin numbers. The password is at least 6 bit for the sake of security.

After business users register, users can login the account as shown in 3.17.

- Upload Clothes in Batch

After business users login, they can upload the clothes by clicking the uploading button in the left side. The operations are quite similar to personal users. The difference is that we support batchsize uploading for business users in view of convenience.

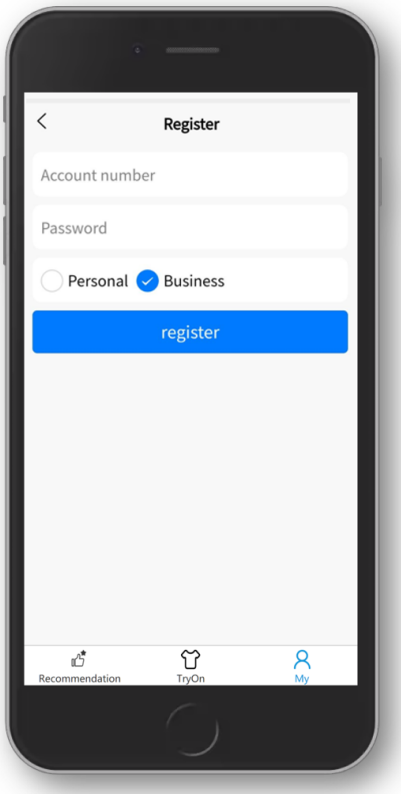


Figure 3.16: Business User Register

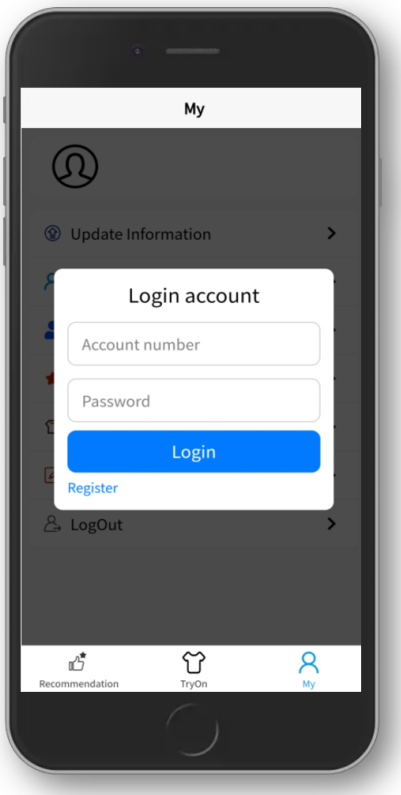


Figure 3.17: Business User Login

- Business Data Analysis and Visualization
  This is also one of the most fancy functions of our application. Business users can click "Data" button in the left down side to use the data analysis and visualization function as shown in 3.18 and 3.19.

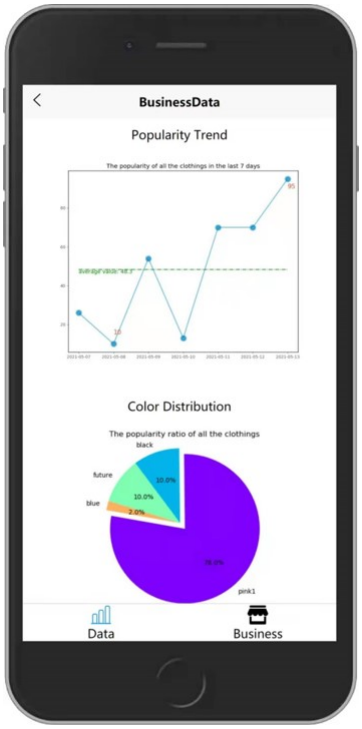


Figure 3.18: Data Analysis Visualization 1

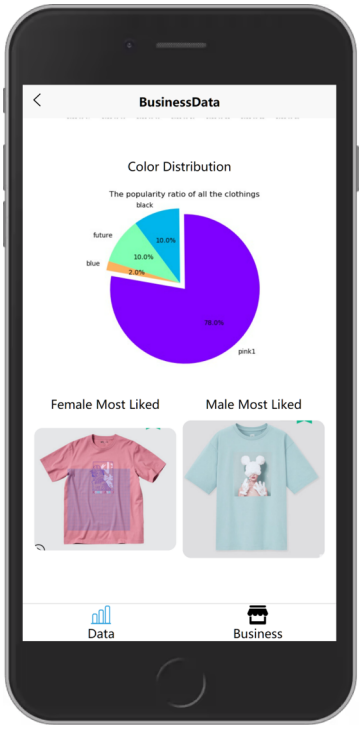


Figure 3.19: Data Analysis Visualization 2

4. Design Document

In this chapter, we will explain in details how we design and implement our application in terms of front-end, back-end, database and algorithms.

4.1 Front-end

For the front-end of AnyFitting – A Virtual Try-on cross-platform App, we used uni-app which uses Vue.js uses Vue.js to develop all front-end applications. We used a certain amount of useful features of Vue. It can quickly render data to the page, and it can also be automatically rendered after the data is updated to enable development Faster, plus some APIs provided by uniapp, certain network requests, and pop-up reminders are more convenient.

If we only use the traditional html+css+JavaScript without using the framework, the development cycle will be approximately long. Using the API provided by uni-app, we can call the interface to get the data faster, and it can also render the data to the page more conveniently. When the data is updated, the data can also be changed automatically. Uni-app helps to manipulate the DOM automatically in high-efficiency and accuracy. Using uni-app, we can also run the project on various platforms (Android, iOS, Web), and we can run it in the APP or applet with a little modification.

As for uni-app code visualization, we used HBuilderX, a universal front-end develop toolkits which is specially enhanced for uni-app. It is a powerful editor and a stable IDE as shown in 4.1.

Uni-app can be used in beautifying and unifying web page styles, implementing the logical part inside and interaction between front-end and back-end. In order to realize a beautiful and uniformed view as a fast development speed, we used lots of existing open-source components to support for API connection. Here is a sample component used for user login as shown in 4.2.

Via uni-app, we can publish it as a mini-program in WeChat / AliPay / Baidu / Bytedance / QQ / 360, or deliver it as a web page both adapted for mobile web browser and PC web browser.

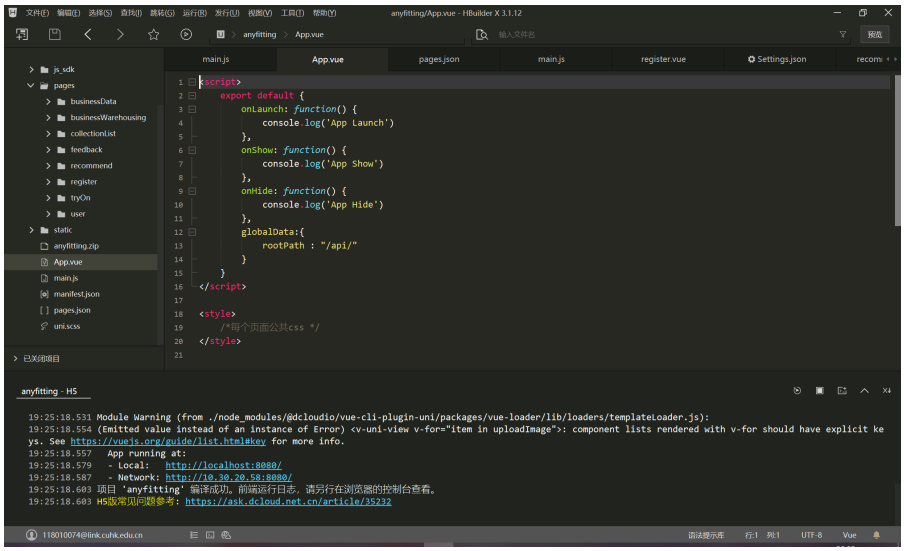


Figure 4.1: HBuilderX IDE

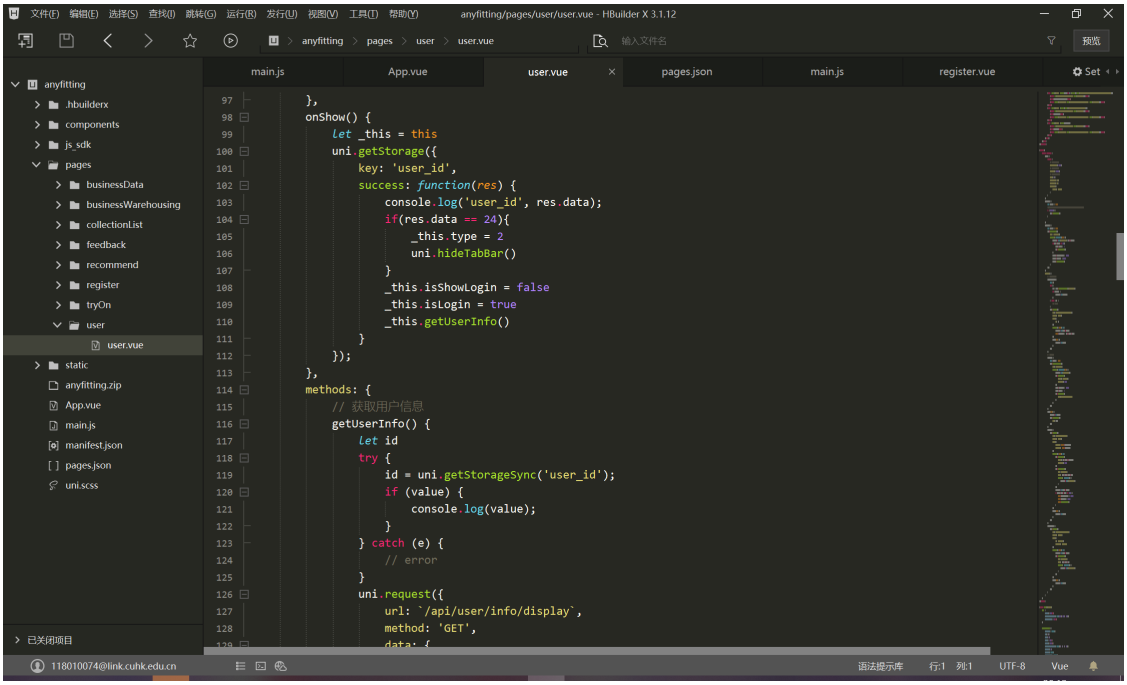


Figure 4.2: API component for get user information

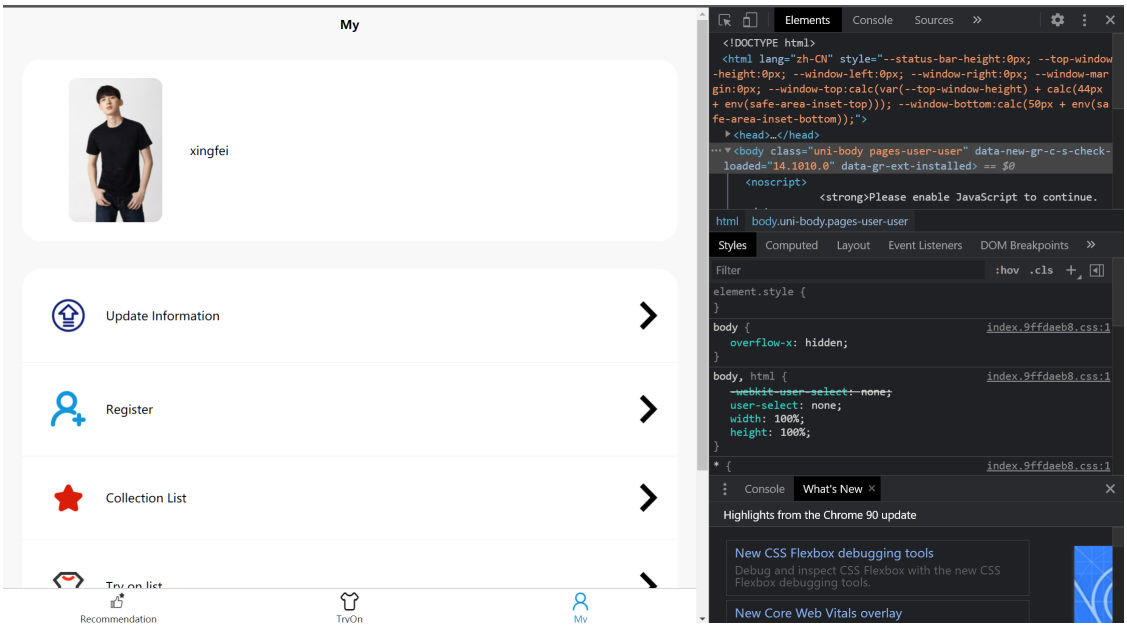


Figure 4.3: uni-app in multiple platforms

```
// 退出登陆
LogOut() {
    let _this = this
    uni.removeStorage({
        key: 'user_id',
        success: function(res) {
            console.log('success');
            uni.showToast({
                title: 'LogOut',
                duration: 2000
            });
            _this.isLogin = false
            _this.type = 1
            uni.switchTab({
                url:'/pages/user/user'
            })
        }
    });
},
jump(url) {
    console.log(url)
    uni.navigateTo({
        url: url
    });
},
```

Figure 4.4: LogOut button in my page

During the interface realization, various integrated components appear and are convenient to use:

- Tabbar
  Tabbar is used to divide the interface into four parts when users enter the app. Individual users can click each tab to enter the My Page, TryOn page, Recommendation page, and business users have business page and data page.

- Swiper
  There are two kinds of swipers used in our program. First one is image swiper, and we apply it to collection list block, discover more block, and recommendation block. And the other one is item swiper, which is widely used in pages where we need to recommendation page.

- Button
  The button is widely used in the program. Its function is to jump to the target page or update data

- Tag
  Tags not only are used in the forum page to differentiate top recommendation items but are used in choosing items to note whether they are finished or still in process. The top tag is used as a component, while the state (finished or not) tag is recognized by image, and we use “finish” variable to determine it.

4.2 Back-end

4.2.1 Back-End Architecture Design

The architecture of Back-End in AnyFitting can be illustrated by this schematic diagram as shown in 4.7.

```
114  methods: {
115      // 获取用户信息
116      getUserInfo() {
117          let id
118          try {
119              id = uni.getStorageSync('user_id');
120              if (value) {
121                  console.log(value);
122              }
123          } catch (e) {
124              // error
125          }
126          uni.request({
127              url: `/api/user/info/display`,
128              method: 'GET',
129              data: {
130                  id: id
131              },
132              success: (res) => {
133                  console.log(res.data);
134                  this.userInfo = res.data.data
135              },
136              fail(err) {
137                  console.log(err)
138              }
139          });
140      },
```

Figure 4.5: Tag Code

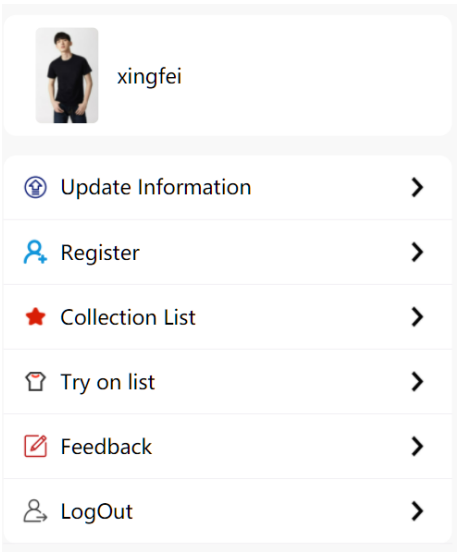


Figure 4.6: Tag UI

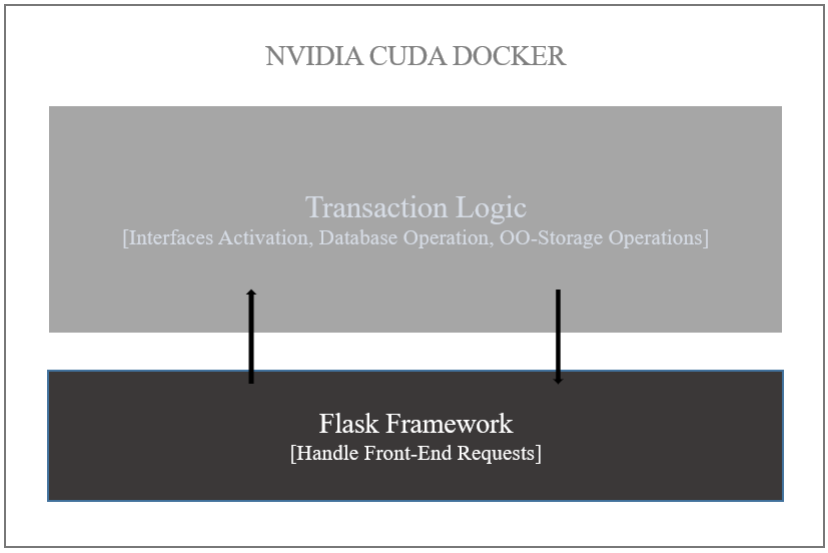


Figure 4.7: whole structure of backend

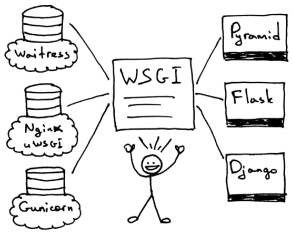


Figure 4.8: theory behind WSGI

As shown in 4.7, the whole Back-End is contained in a NVIDIA CUDA DOCKER and there are mainly two functional parts in Back-End, the Flask Framework part and Transaction Logic part. More details about their functions will be explain then.

4.2.2 Implementation

4.2.2.1 Flask Framework

Flask is a lightweight WSGI web application framework. The WSGI means this framework uses the basic Web Server Gateway Interface, which allows the web framework to ignore details in web communication at Operating System level. Of course, Flask has a build-in WSGI implementation called wsgiref, which is easy to use but has no outstanding performance. The nginx/gunicorn framework can be combined with Flask to replace the default wsgiref module. The relationship between these frameworks can also be illustrated by this figure.

Here we want to show how we construct the back-end with Flask by three features of Flask most commonly used in this project, the Routes, Requests and CORS.

- Routes
  As mentioned in above sections, we develop this project with separation of frontend and back-end. The communication form between frontend and backend is REST API, which means all the resources should be found by URL and use HTTP operations (GET,POST,DELETE,...) to request. So, this project use routes in flask to assign different sources to interfaces. For example, the route /user/login is assigned to the login transaction. Back-End application will monitor this port and return the login state when Front-End send a request with parameter username and password to this route.

- Requests
  The request from frontend takes information for backend, then backend will use Request object in Flask to get the whole information the request taken. What is more, the information can be taken in every parts of http request.

- CORS
  The frontend of this project is written in Vue.js, which runs in browser. When browser execute the JavaScript program, it will forbid the program to send Request to a strange website, which means the target website does not has same domain name, port and communication protocol. There are multiple methods to solve this problem, the method used in our project is CORS, the Cross-Origin Resource Sharing, which is used in backend to inform the browser that this server receives cross-domain request. This process is shown in below figure.

4.2.2.2 Transaction Logic

We have known how the backend communicates with frontend, here we want to show how backend determines the information return to frontend. In addition, the backend of our project returns the data in JSON format. Here gives an example return statement under a route. Obviously, here we get the parameters from the request and call an API function to get the return value as shown

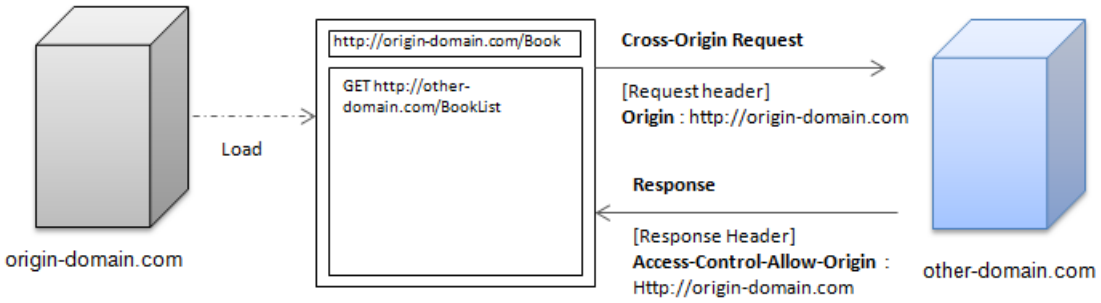


Figure 4.9: CORS

```
@app.route('/user/info/update',methods=['PUT','POST','GET'])
def update_backend():
    id = request.args.get('id')
    mailbox = request.args.get('mailbox')
    passwd = request.args.get('password')
    nickname = request.args.get('nickname')
    picture = request.args.get('picture')
    birthday = request.args.get('birthday')
    sex = request.args.get('sex')
    flag = update(id,mailbox,nickname,picture,birthday,sex)
    if (flag):
        return json.loads('{"code":200,"message":"success","data":null}')
    else:
        return json.loads('{"code":400,"message":"fail","data":null}')
```

Figure 4.10: Example of api function

in 4.10. In following of this part, we want to show how backend determines the data by introduce different kinds of API functions in backend.

As Figure 4.11 shows, the transaction functions in Back-End can be divided into three kinds: Database Functions, Computer Vision Functions and Object-Oriented Functions. The Database Functions are responsible for querying data from cloud database by different conditions. The Computer Vision Functions are responsible for executing pictures pre-processing and synthesis. The Object-Oriented Functions are responsible for upload/download pictures from Tencent Cloud. These functions are combined with each other to get correct return value under different route functions.

4.3 Database

In order to implement a dynamic application, we construct a remote database on Alibaba cloud server. Alibaba RDS SQL Server is distinguishable in security maintaining, data and resource monitoring, high concurrency of hot data, fault handling and etc.. These properties fit perfectly on a potential business trading platform.

Considering the business attributes of the merchant clothing, we separate the information storage of the users and merchants to avoid unnecessary null values and filtering operations, so as to achieve the third norm form. On the other hand, it assures the privacy among users. Additionally, we also implement a well-designed interface to reduce the data redundancy by always checking in before inserting. Indexing on frequently used attributes also accelerates the operation speed of our application.

The relational schema as well as the type of each attributes are shown below. A brief description of each table is as followed.

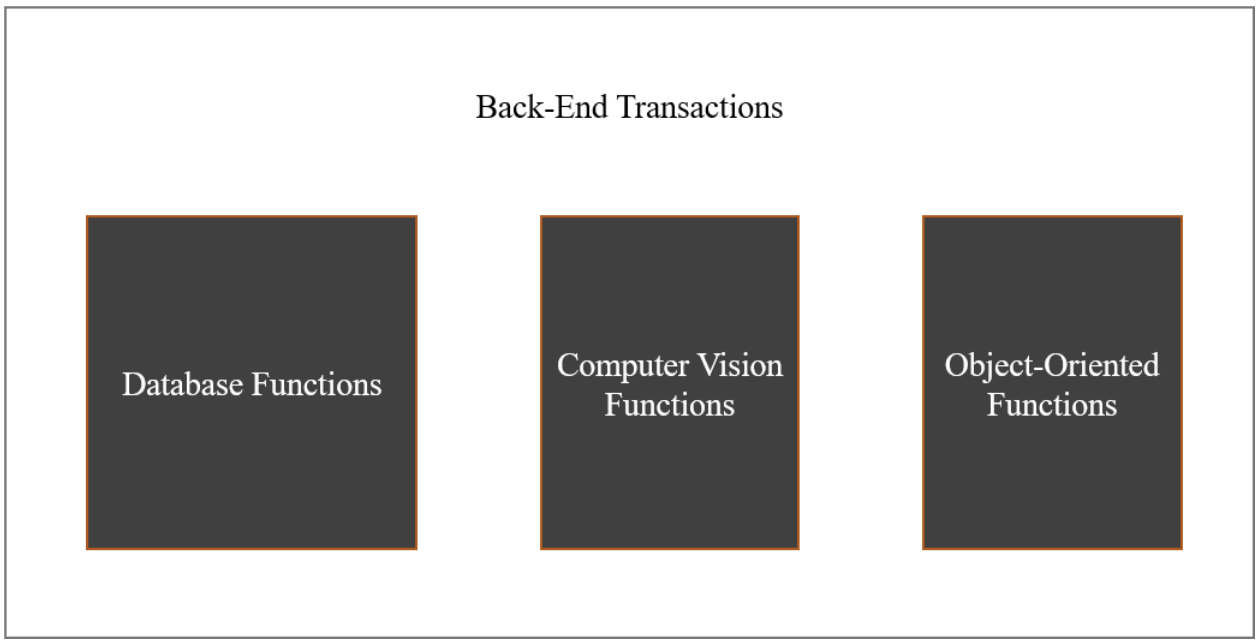


Figure 4.11: Transaction Process

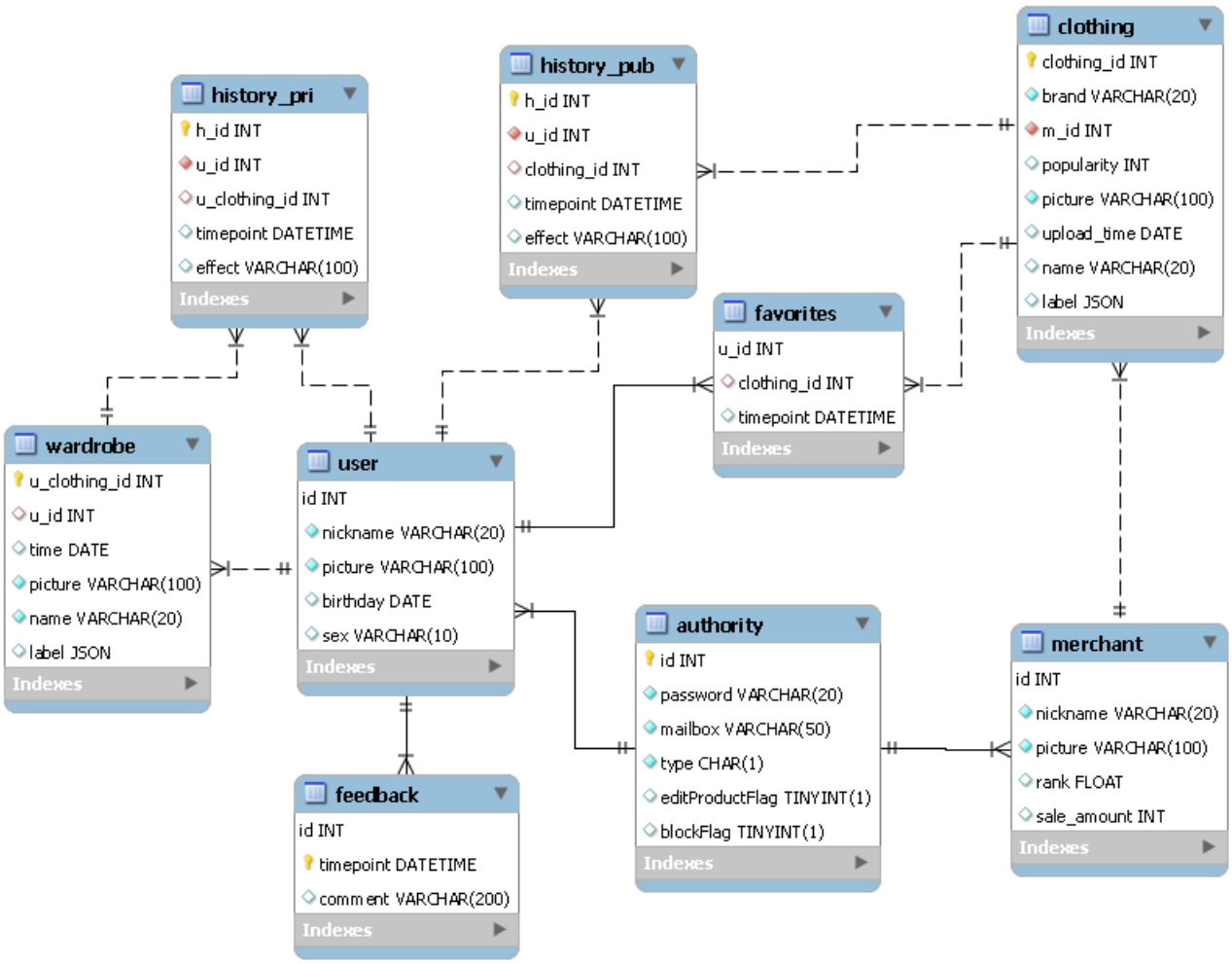


Figure 4.12: The ER diagram of database

- Authority
  It stores the basic and private information of each account. It is used to for registering, account authority checking, and potential account banning. In order to maintain the security, we apply the hash algorithm to encrypt the password.

- User
  It stores the displayable personal user information. The figure picture of each user is store here as well. The user information (such as gender and age) assists to implement the data analysis as well.

- Merchant
  It stores the displayable merchant information. It also contains the information about the shop rank and sale amounts, which extend the potential of our application to develop as a trading platform.

- Clothing and wardrobe
  The table clothing stores all the information of user-uploaded clothing. The table wardrobe stores the information of merchant-uploaded clothing. The information separation is due to the extra attributes (such as brand, popularity and label) of the business-purposed clothing. They will be utilized to achieve the recommendation, customer analysis and other business data mining.

- History_pub and history_pri
  Similarly to above, the try-on records of the user-uploaded clothing and merchant-uploaded clothing are respectively stored in two tables. Together with the ids of the user and the clothing, the final effect pictures will be also kept for displaying history queries. These records are essential for personalized recommendation and customer analysis.

- Favorites
  It acts as favorites so as for the convenient access to the preferred clothing. It assists the user taste analysis as well.

- Feedback
  It is an additional table so as to offer a feedback channel of each particular user. No user but the administrator is able to access.

4.4 Algorithms

The algorithm part of our project was based on https://github.com/sergeywong/cp-vton, which is the official implementation of eccv2018 paper 'Toward Characteristic-Preserving Image-based Virtual Try-On Network'. Our contribution are: 1.We fuse two deep learning models (human parsing model and pose estimation model) into one model sharing one Resnet101 backbone but three different head for three different task; 2. We refact the repo code of cp-vton such that it can accept the result of our new model; 3. Since the original cp-vton is pre-trained on American women's image, which perform badly on Asian cloth, we collect 100 clothes from the internet and finetune the model.

4.5 human parsing and human pose algorithm

As we can see from the repo of cp-vton, the project require a complex pre-process including using JPPNet to segment the human segmentation and cloth segmentation, and Openpose to estimate the pose of human. After above operations, it can finnally run the tryon algorithm. Due to the limited GPU memory of RTX2060, we have to run two models sequentially. These two models take more than 9 GB GPU memory in total, thus we failed to hold the GPU model in memory for quickly inference. Therefore, the pre-process process, as far as we know, cost more than 6 seconds (inluding two model loading into memory and inference). Inspired by CE2P [**?**] which do the segmentation task and edge detection task at the same time (Figure 4.13). We design an double-head network based on DeeplabV3Plus[**?**], where two head was used for segmentaion and pose estimation correspondly. Sepcifically, the segmentation branch output an mask with the same

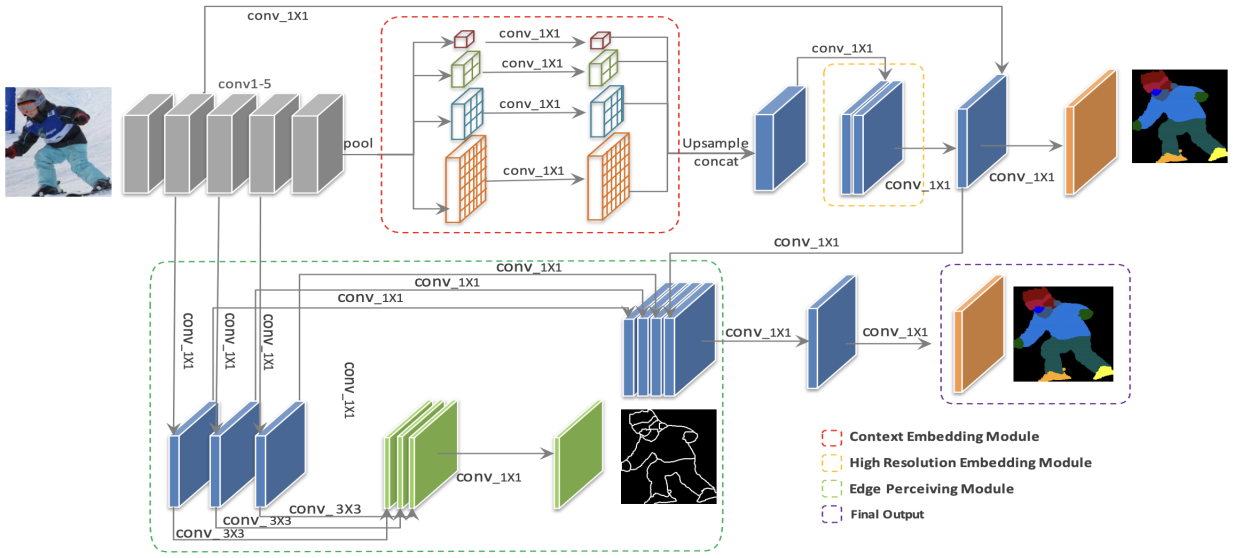


Figure 4.13: Framework of CE2P network

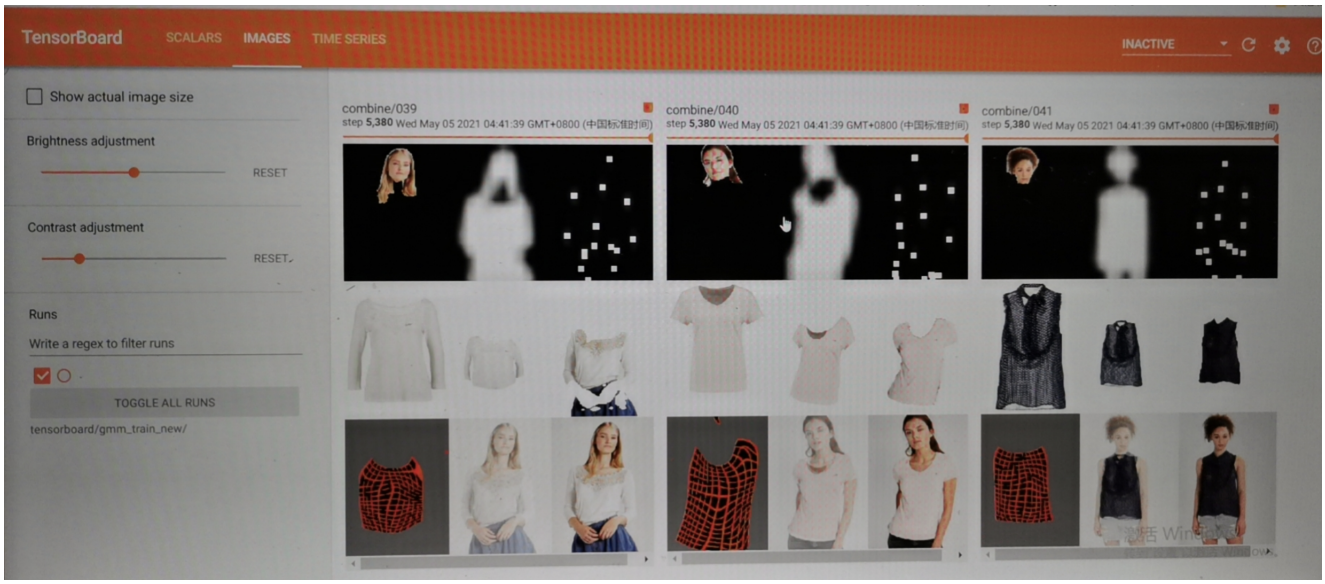


Figure 4.14: Re-Training process and sample output of our network

size of the original image, where the value represent the class of corrsponeding pixel. Besides, the pose branch output an heatmap with the size of (20, 1/4 * h, 1/4 * w). Then, we upsample the heatmap to the orinal size and grap the local maximam point as the keypoint. Finally, we save the location of estimated pose keypoints in the form of json and segmenation mask as a one-channel png file. These two file will be passed to tryon algorithm. Figure 4.14 show the sample output and training process of our network.

4.6 Try-on algorithm

You can find the detailed algorithms in this paper "VITON: An Image-based Virtual Try-on Network". We referred to an image-based VIirtual Try-On Network (VITON) without using 3D information in any form, which seamlessly transfers a desired clothing item onto the corresponding region of a person using a coarse-to-fine strategy. Conditioned upon a new clothing-agnostic yet descriptive person representation, our framework first generates a coarse synthesized image with the target clothing item overlaid on that same person in the same pose. We further enhance the initial blurry clothing area with a refinement network. The network is trained to learn how much detail to utilize from the target clothing item, and where to apply to the person in order to synthesize a photo-realistic image in which the target item deforms naturally with clear visual patterns. Experiments on our newly collected Zalando dataset demonstrate its promise in the image-based virtual try-on task over state-of-the-art generative models.

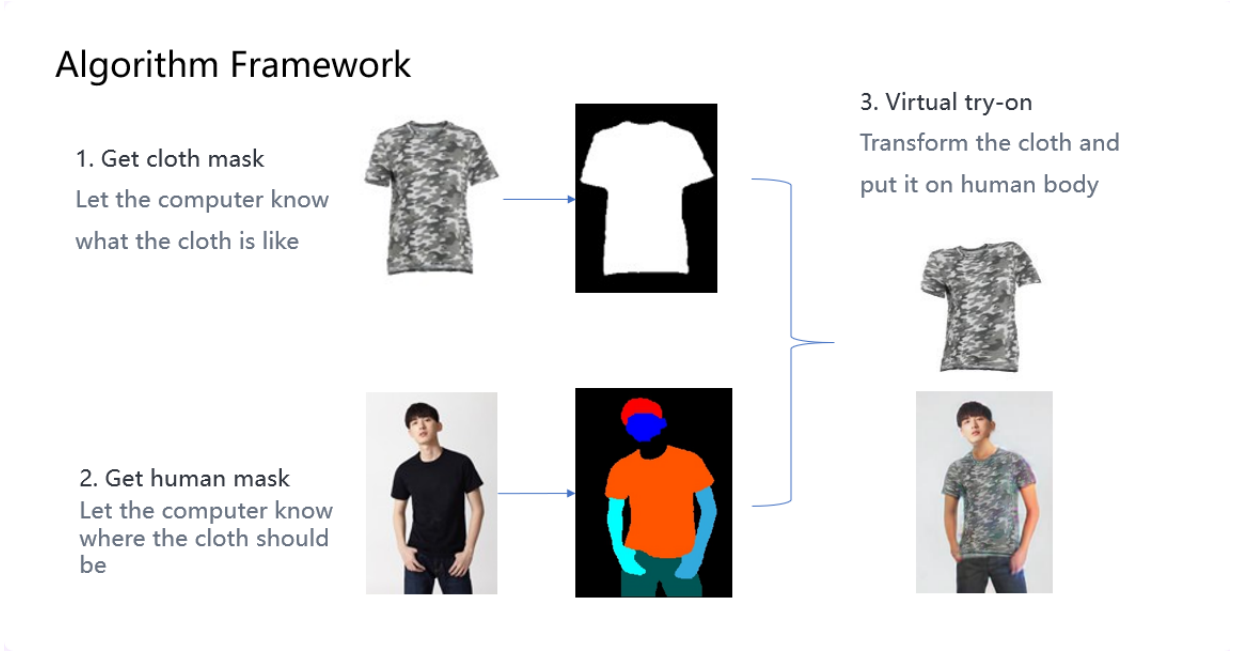


Figure 4.15: The structure of our model

The goal is, given a reference image I with a clothed person and a target clothing item c, to synthesize a new image, where c is transferred naturally onto the corresponding region of the same person whose body parts and pose information are preserved. Key to a high-quality synthesis is to learn a proper transformation from product images to clothes on the body. A straightforward approach is to leverage training data of a person with fixed pose wearing different clothes and the corresponding product images, which, however, is usually difficult to acquire. In a practical virtual try-on scenario, only a reference image and a desired product image are available at test time. Therefore, we adopt the same setting for training, where a reference image I with a person wearing c and the product image of c are given as inputs. Now the problem becomes given the product image c and the person's information, how to learn a generator that not only produces I during training, but more importantly is able to generalize at test time – synthesizing a perceptually.

4.7 Deployment Server

As for deployment, we adopt an architecture to separate web server and application server (Fig 4.16). We use the web server to prevent SQL injection and Challenge Collapse, as well as expose the local IP. Besides, a personal computer with GPU was used as application server to provide model inference service and store MySQL data.

4.7.1 web server

4.7.1.1 Baota Firewall

We use an Commercial off-the-shelf (COTS) software called Baota Firewall to protect our port and server.

4.7.1.2 Fast Reverse Proxy(FRP)

Due to the lack of public IP, we have to use an FRP service to export the IP of our application server for public service. We adopt an open source software - FRP**??** to perform reverse proxy. It supports TCP and UDP, as well as HTTP and HTTPS protocols, where requests can be forwarded to internal services by domain name. Figure 4.17 show the funciton and architecutre of the FRP service.

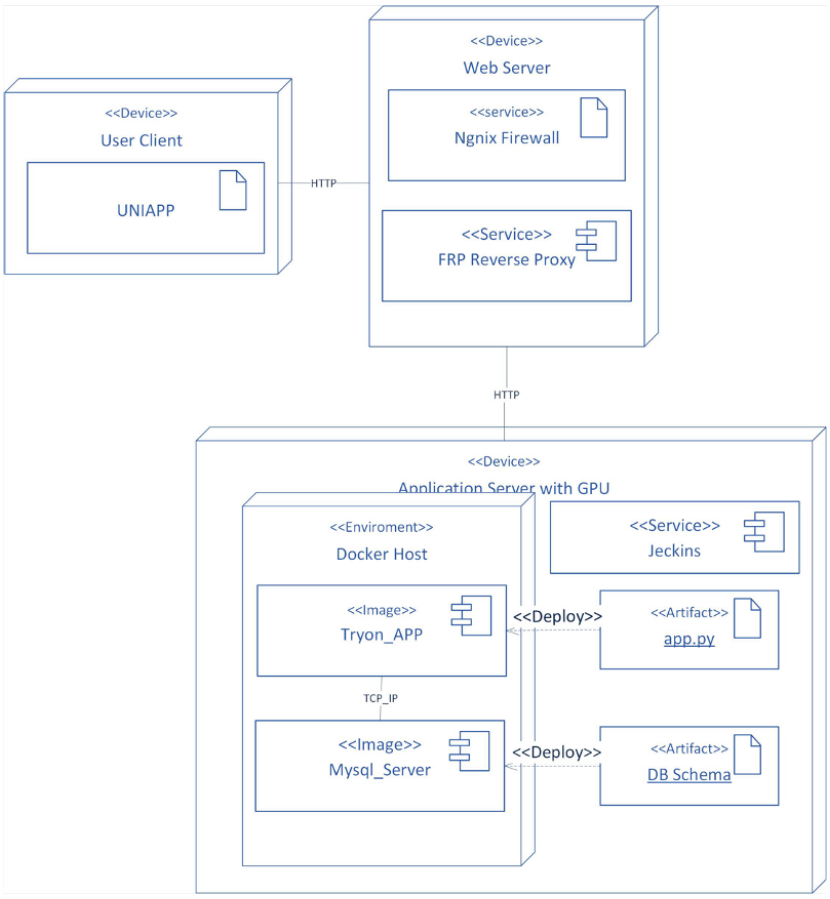


Figure 4.16: Overall deployment framework

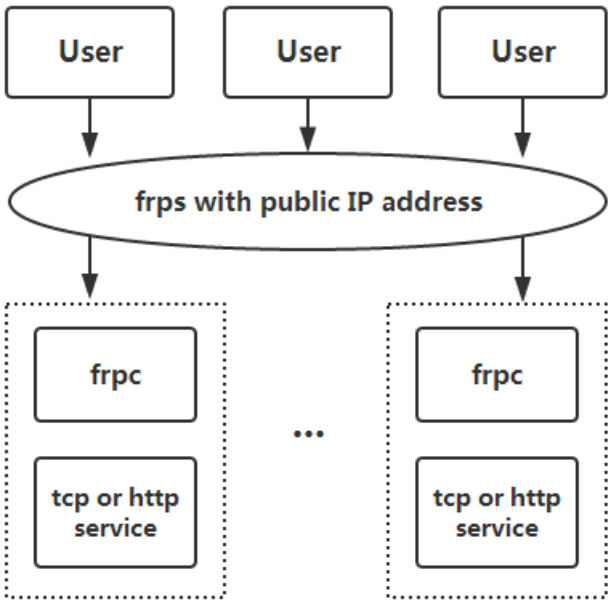


Figure 4.17: FRP Architecture

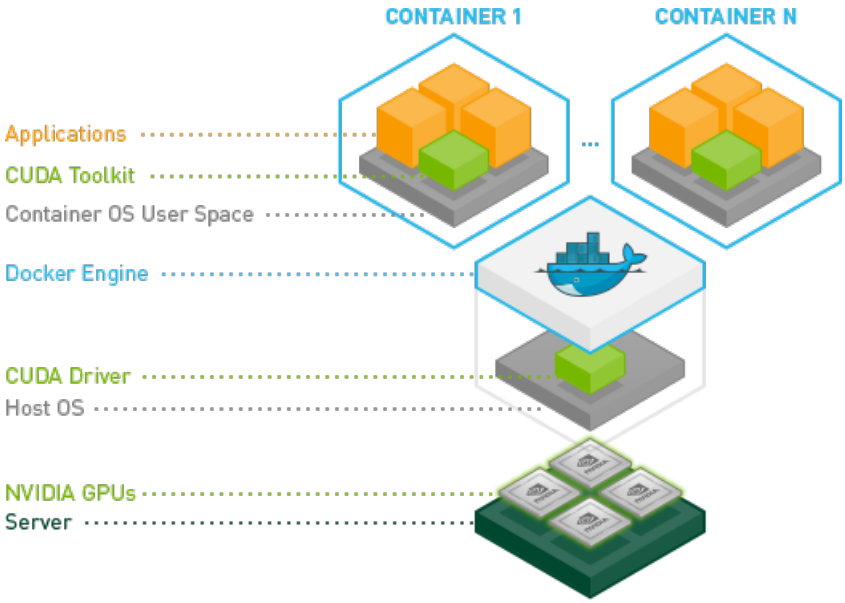


Figure 4.18: NVIDIA Docker Architecture

4.7.2 application server

4.7.2.1 NVIDIA Docker

Docker is a mechanism for bundling a Linux application with all of its libraries, data files, and environment variables so that the execution environment is always the same, on whatever Linux system it runs and between instances on the same host. Docker containers are user-mode only, so all kernel calls from the container are handled by the host system kernel [**?**]. Since docker containers are hardware-agnostic, it failed to use specialized hardware such as NVIDIA GPUs which require kernel modules and user-level libraries to operate. As a result, Docker does not natively support NVIDIA GPUs within containers. To use Docker images with NVIDIA GPUs, we adopted nvidia-docker, an open-source project hosted on Github that provides the two critical components needed for portable GPU-based containers, as Figure 4.18 shows.

4.7.2.2 Jenkins

Jenkins is an open source automation server which enables developers to reliably build, test, and deploy their software[**?**]. Jenkins could manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage, deployment, static code analysis and much more. In this project, we use Jenkins to devleop an CI/CD pipeline[**?**]. CI/CD here means Continuous Integration and Continuous Delivery. It is one of the best practices for devops teams to implement. It's known as an agile methodology best practice, as it enables software development teams to focus on meeting business requirements, code quality, and security because deployment steps are automated. Continuous integration is to drive development teams to implement small changes and check in code to version control repositories frequently, while continuous delivery automates the delivery of applications to selected infrastructure environments. Usually, CD picks up where continuous integration ends of CI. Figure 4.19 show our designed CI/CD pipeline.

As graph 4.20 shows, our CI/CD pipeline include three steps: 1.Automatic unit test; 2.Automatic deployment; 3.Automatic reminder. In the first step, once there is code push or merge to target branch, Jenkins service will firstly pull the code to local as 4.21 shows. Then a series of regression tests were performed to ensure the code compile and built correctly. In the second step, Jenkins server will automatically deploy the code to target directory and run the docker worker. Once there is docker worker running, it will kill it firstly and start it again with new code to restart the back-end and algorithm service. The process takes around 3 5 seconds averagely. In the third steps, an informed email will be send to adminstartor and related developers email to inform the results of the build. Graph 4.22 is several subjects of informing email and 4.23 is the content of one email.

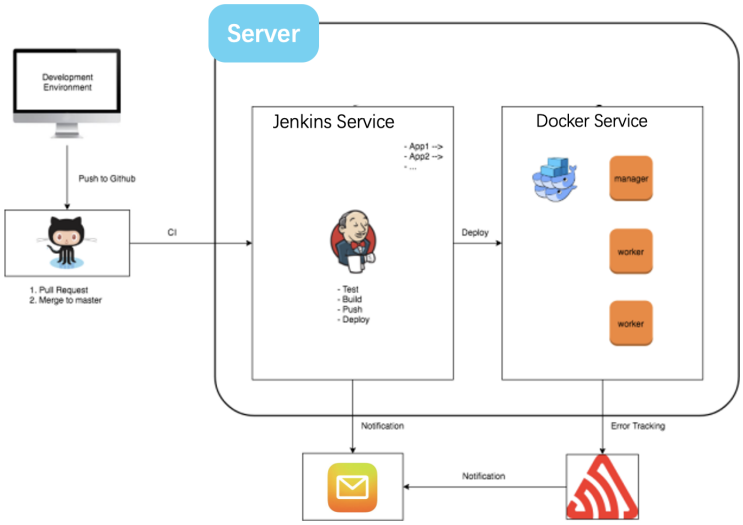


Figure 4.19: CI/CD Diagram based on Git, Jenkins, Docker

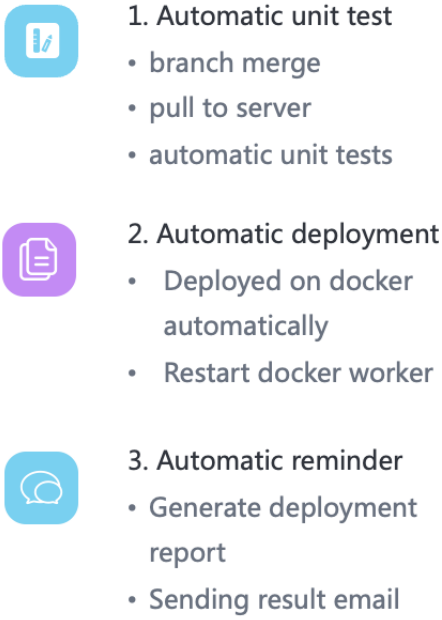


Figure 4.20: CI/CD Pipeline

Figure 4.21: Code Fetch Example

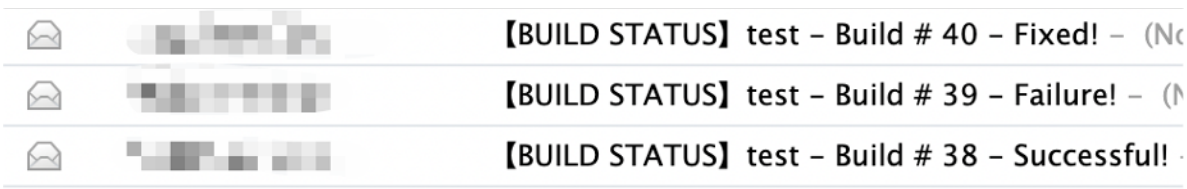


Figure 4.22: Subject of informing email

Figure 4.23: Build status email example

5. Test Document

The testing tools we used in development are Apifox and postman, two interfaces management testing software. In the different part of development, we take different testings.

5.1 Unit Testing

When the backend is constructing, the API functions will be used is also developing. Just as the database functions and computer vision functions. During this period, we have done the unit testing, which means we manually call the functions and check if the result correct.

5.2 Component testing

When the backend is combined with API functions, we begin to use Apifox to test if the interface works well. What is more, in this period, the frontend also uses Apifox to test if the graphic user interface works well. The Apifox will provides a Mock Environment for frontend, which simulates the data return from backend by a port of local address as shown in Figure 5.1.

5.3 System Testing

When frontend and backend have done the component testing, we will link the frontend and backend and directly test if every function of application works well.

5.3.1 Performance Testing

When complete the system testing, we begin our performance testing using postman. The main conditions we focus are speed test and pressure test. Here figure 5.2 and figure 5.3 gives the results.

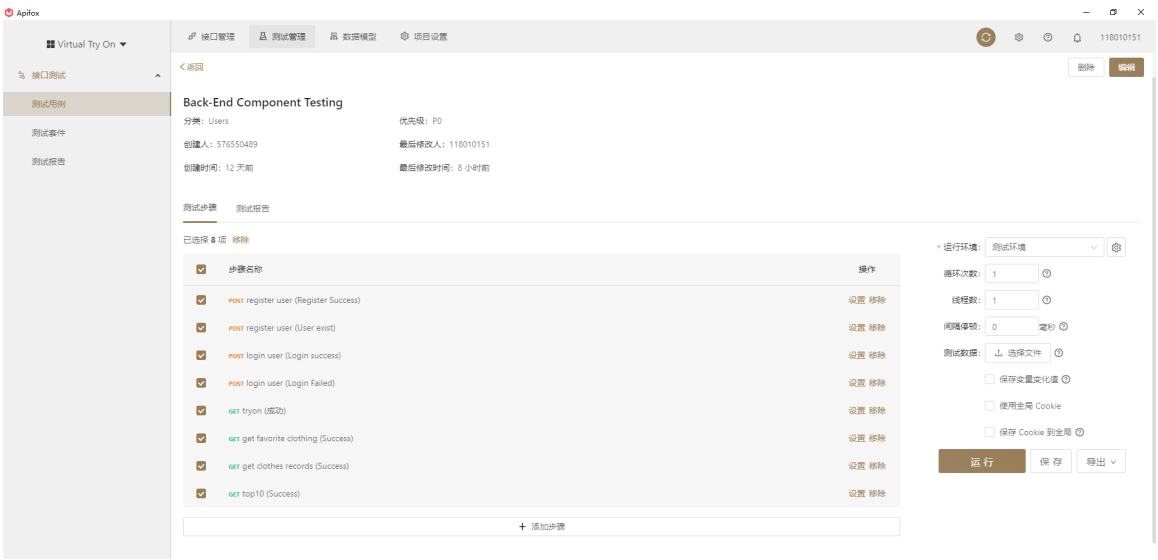


Figure 5.1: Component Testing

Figure 5.2: Test result 1

Figure 5.3: Test result 2

6. Discussions and Conclusions

6.1 Future Work and further improvement

So far, we implemented a commercially available virtual try-on cross-platform app which can work on Android /iOS / Web. AnyFitting integrated almost all the possible try-on operations: clothes uploading, virtual try-on, building a online wardrobe (collection list and try-on list). Moreover, AnyFitting is not only designed for individual users but also for business users to manage online warehousing, view personalized data graph with users, and do business data analysis and visualization. By applying uni-app for front-end, Flask and REST for back-end, Docker, Alibaba Cloud, BT.CN, Tencent Cloud, interface management system (Apifox) and two version control systems (Gitee, Github) for data storage, algorithm implementation, and data analytics, our AnyFitting shows significant superiority in the efficiency of model inference and concurrency of application server.

Overall, limited by time and commercial requirements, AnyFitting is a well-developed cross-platform app consisting of considerable design, thought-provoking ideas, and theoretical support for virtual try-on experience and commercial opportunities. While this app has supplied multiple basics for a system to tracking try-on experience and personalized recommendation for individual users, online warehousing and data visualization for business users, some further enhancements are desired for greater using experience of the information.

The first major enhancement would be for individual users to be able to have communications with each other and view others' comments on recommended clothes. They are supposed to be able to see why the specific top ranking formed and how about other's view on the specific clothes. With more sophisticated recommendation page, both individuals and business men can interact with no obstacles.

Another useful addition to AnyFitting would be an extended system by which individual users have a mailbox which contains public information and commercial promotion message. From making operations more efficient to improving customer service, a shared mailbox has several advantages over using individual accounts alone:

- Monitor and send email from a centralized account.
- Handle customer email inquiries faster.
- Share personalized recommendation information in a humanized way.

Finally, data visualization is better to be customized for business users. This feature gives the authorization for business users to adjust viewing components. It might be complex on the administrative end, but it can be worked out and will ultimately differentiate the user as an app owner.

6.2 Summary of our work

In conclusion, our application is a literally a huge project utilizing many platforms. Our application has the following superiority. First, the algorithm is a state-of -the-art research project, and its performance is distinguished. Second, the figure image is user-generated, which makes our final effect much more customized in terms of both gestures and figure sizes. Third, it is prior to

current virtual fitting applications.

Compared with the applications using 3-D models, our image processing model is much more economically friendly and low-threshold, and users are free to upload and fit any clothing. In comparison to the current 2-D virtual fitting, our application outperforms in both speed and robustness. Fourth, we have made efforts in maintaining the security, concurrency and fault tolerance. This offers a great potential for our application to be a trading platform. With mentioned above, our application is promising and commercially valuable.

7. References

Oberol (2021, Feb 16). *How many people shop online in 2021?.* OBERLO. Retrieved May 14, 2021 from https://www.oberlo.com/statistics/how-many-people-shop-online

Titanium. (2021, Jan 23). *Virtual Fitting, perfect in concept but meaningless in practical?.* Netease. Retrieved May 14, 2021 from https://www.163.com/dy/article/G10V1SN605118O92.html

K. Gong, X. Liang, X. Shen, and L. Lin. Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In CVPR, 2017. 3

S. Liu, Z. Song, G. Liu, C. Xu, H. Lu, and S. Yan. Street-toshop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In CVPR, 2012. 2

M. Sekine, K. Sugita, F. Perbet, B. Stenger, and M. Nishiyama. Virtual fitting by single-shot body shape estimation. In 3D Body Scanning Technologies, 2014. 1, 2

Han, X., Wu, Z., Wu, Z., Yu, R., Davis, L. S. (2018). Viton: An image-based virtual try-on network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7543-7552).