**Database Group Project Document**

**Sneaker Transaction System**

Group 4

Members:

Wang Bohan 1830010026,

Li Yixiao 1930026073,

Zhou Yisong 1930026175,

Zhou Hongjin 1930006275

Contents

[1 Brief Description 1](#_Toc73559238)

[1.1 Background 1](#_Toc73559239)

[1.2 Purpose & Functionalities 1](#_Toc73559240)

[2 Data Description 2](#_Toc73559241)

[2.1 Data Collection 2](#_Toc73559242)

[2.2 Data preprocessing 2](#_Toc73559243)

[2.3 Data Analysis & Visualization 3](#_Toc73559244)

[3 Assumptions 4](#_Toc73559245)

[4 ER Diagram 4](#_Toc73559246)

[4.1 Diagram 4](#_Toc73559247)

[4.2 Difficulties and Solutions 5](#_Toc73559248)

[5 Functional Dependencies 6](#_Toc73559249)

[6 SQL Codes and Explanations 6](#_Toc73559250)

[7 Website Design 9](#_Toc73559251)

[7.1 Main Menu 10](#_Toc73559252)

[7.2 Login & Register 10](#_Toc73559253)

[7.3 Change Information 11](#_Toc73559254)

[7.4 Buyers: Buying Goods 12](#_Toc73559255)

[7.5 History Order 12](#_Toc73559256)

[7.6 Seller: Sell the Products 13](#_Toc73559257)

[7.7 Administrator page 14](#_Toc73559258)

[8 Extra: speed of adding, deleting and searching data 14](#_Toc73559259)

[9 Extra: Manual to import the data 17](#_Toc73559260)

**Database Group Project Document**

Group 4 Members: Wang Bohan 1830010026, Li Yixiao 1930026073,

Zhou Yisong 1930026175, Zhou Hongjin 1930006275

1 Brief Description

1.1 Background

E-commerce is now a very popular new way of buying and selling. And sneakers are a very popular commodity in e-commerce transactions. On the other hand, the personalized demand of shoes is also a kind of service that the market is in urgent need of. Nowadays, however, customization services are mainly provided by third parties (service parties other than sellers and buyers). In this way, the service system of the buyer -- the seller -- the third party is not only cumbersome and complex, but also very inconvenient for the buyer. Our sneaker management system integrates the two functions of buying and selling and customization to realize the dream of directly "face to face" transaction and customization between buyers and sellers. Furthermore, our system allows buyers to choose between direct or custom trading, which allows both the custom shoe and the brand shoe prototype trading functions to exist side by side.

Because now the sneaker is a hot commodity on the hot e-commerce platform, individuation is also one of the things that people need most in the 21st century. People are eager to differentiate themselves, but they are stopped by tedious third-party customized services. Our system combines the trading function with the customization function, making it easier for customers who need to customize to get a different shoe through our platform. So this is interesting. Customers who have customized needs, sellers who can provide customized services and sellers of shoes are all users of our shoe management system.

1.2 Purpose & Functionalities

We plan to make a transaction system that can clearly show the information of each pair of sneakers when customers buy it, and provide searching ability for both buyers and sellers in order to find the inventory or the order of a specific kind of sneakers conveniently. To achieve that, the program should satisfy the following functionalities:

1. Send the orders of the buyer to the seller and the product custom manufacturer in time through the order data generation and trading system.

2. According to the customization requirements of the buyer, the customized service information shall be transmitted between the customization party and the buyer through the corresponding customized service page and procedures.

3. The buyer can search according to the commodity in the webpage, obtain the basic commodity information such as brand, category and price, and also search the other basic information of the commodity according to the information.

4. In the order management system, the seller has the functions of editing and creating the order data. For example, the store can take the initiative to modify the inventory state of some kind of sneakers.

5. The seller can inquire the basic information of the order and the status of the goods through the search function.

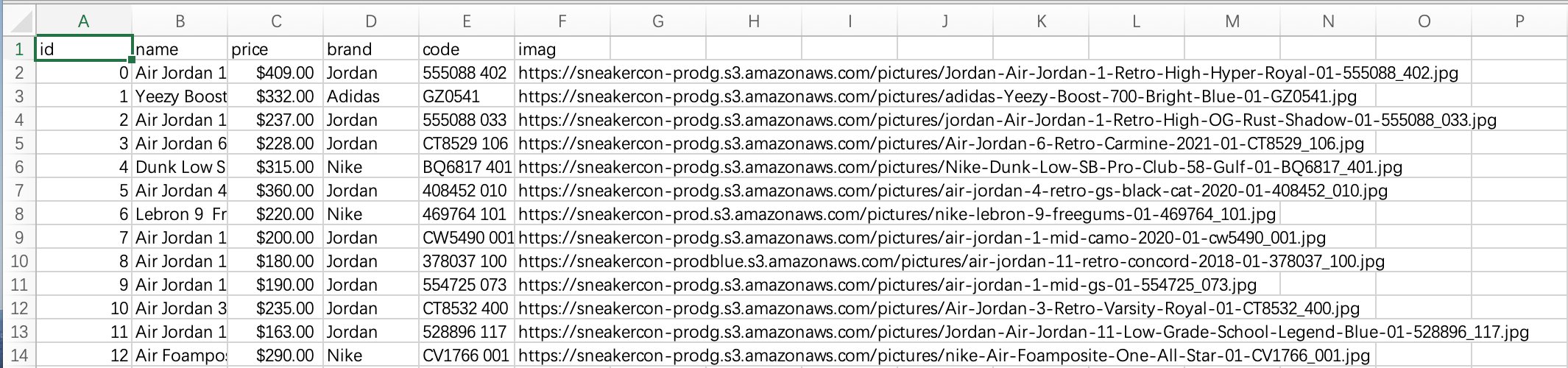
6. The manager has the right of data monitoring, and can directly access the detailed information of the buyer, seller and order through the order management system with the authority of the administrator. The error information in order can be compared and corrected timely.

7. The system shall be able to update product information in time and upload commodity data in time when order is generated, and update the data to indicate the change of trade status when the commodity trade is completed or there is a problem.

8. The web UI interface and account system should achieve the purposes above. The seller, the buyer and the program manager enter the corresponding operation interface when logging into the account.

2 Data Description

2.1 Data Collection

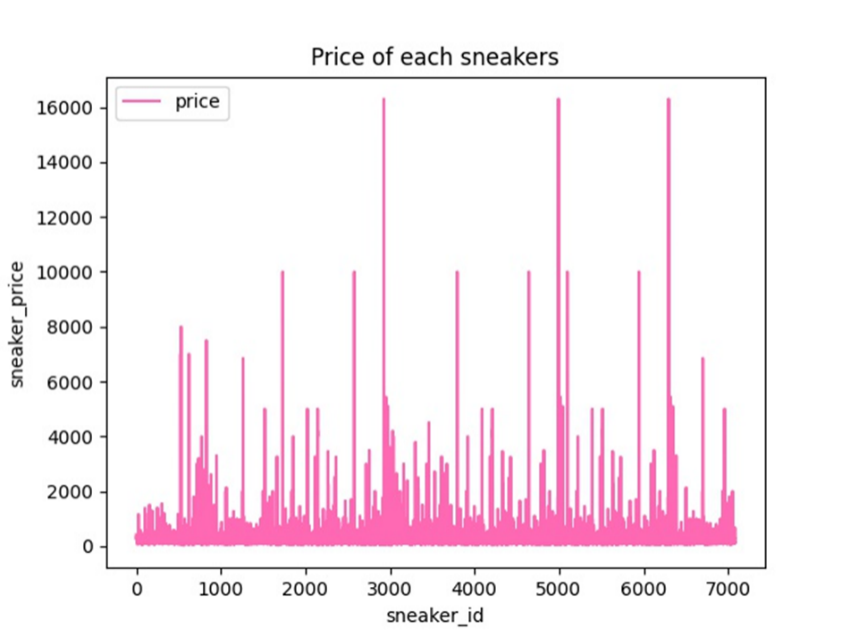
In the project, we have collected about 7200 data of the sneakers from a sneaker-selling website: https://sneakercon.com/shop , and all the data contains the information of the sneaker, such as name, price, brand, code, and image. A part of the data caught is as follows:

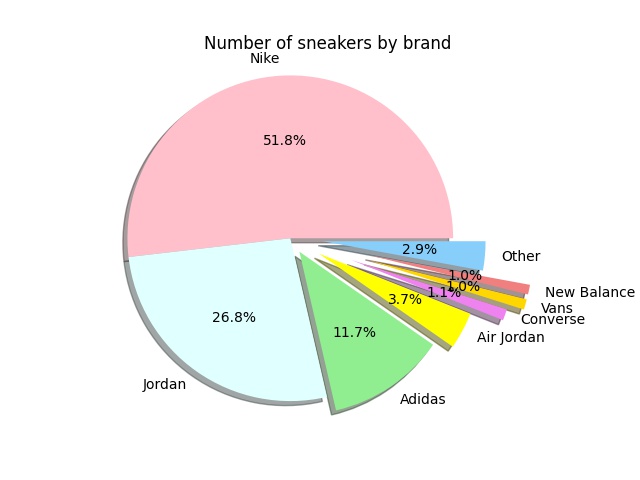
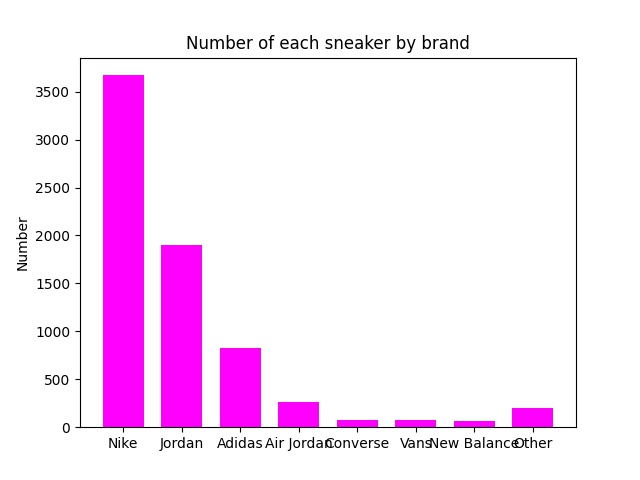
2.2 Data preprocessing

After catching data on the website, we fill the non-values with the average value of all the sneakers of its brand, and then transfer the image link we get in the column “imag” into real image file.

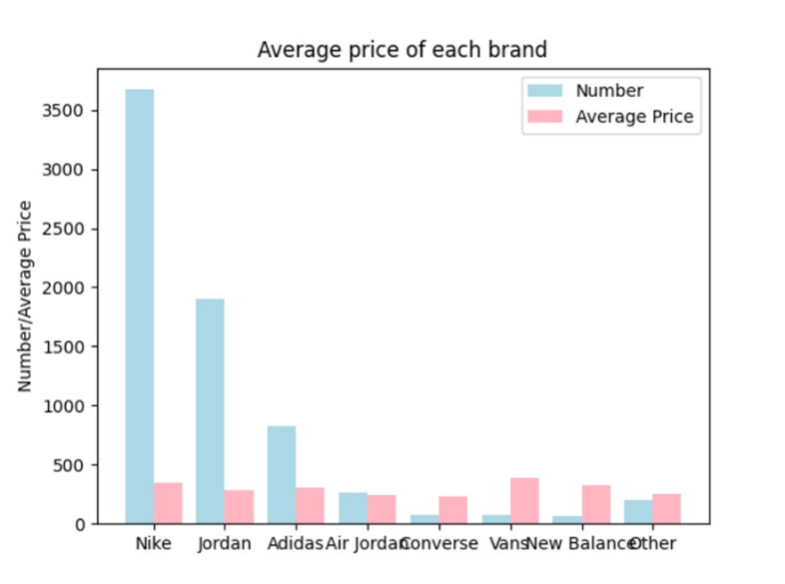
After collecting the data on the website, we generated more than 10 separate data bases on the information of each sneakers. Each sneaker has its own unique ID, so there’s totally 80,000 data of sneakers.

The detailed web crawler code is in the file “crowler”.

2.3 Data Analysis & Visualization

According to the figure above, we can get the price distribution of different kinds of sneakers. The most expensive sneaker is the red-circled one, which is $16,304. The cheapest price is $20. The average price of all sneakers is $315.84.

The pie chart above is the proportion of sneakers numbers of each brand. The most one is Nike, which takes 51.8% part of total. The least ones are Vans and New Balance, which only takes 1% part. The histogram on the right shows the number of sneaker types of each brand, and we can get the same result like the pie chart.

Additionally, we got the average price of each brand and then made a double histogram. The red part shows the average price of sneakers. Comparing those brands, we found that Nike has the most sneaker data with normal average price, while Vans have the least data with most expensive average price. Converse has the least date, but its average price is also the cheapest.

3 Assumptions

About the database design, we have the following assumptions about relationship and tables:

Attributes of Entities / Columns of Tables

1. Basically, we have the entity sets of orders, sellers, buyers, sneakers and customized sneakers’ information.

2. All the website users are divided by 2 groups: buyer and seller. Each buyer and seller data should contain username, password, address, date of registering.

3. Sneakers information should contain its name, price, brand, and image.

4. Customized sneakers need a table to contain the information of customization. They have another customized product ID.

5. Order table stores the information of each order, such as buyer’s name, seller’s name, sneaker’s ID, and an ID of customized orders to make relationships with other tables.

Relations:

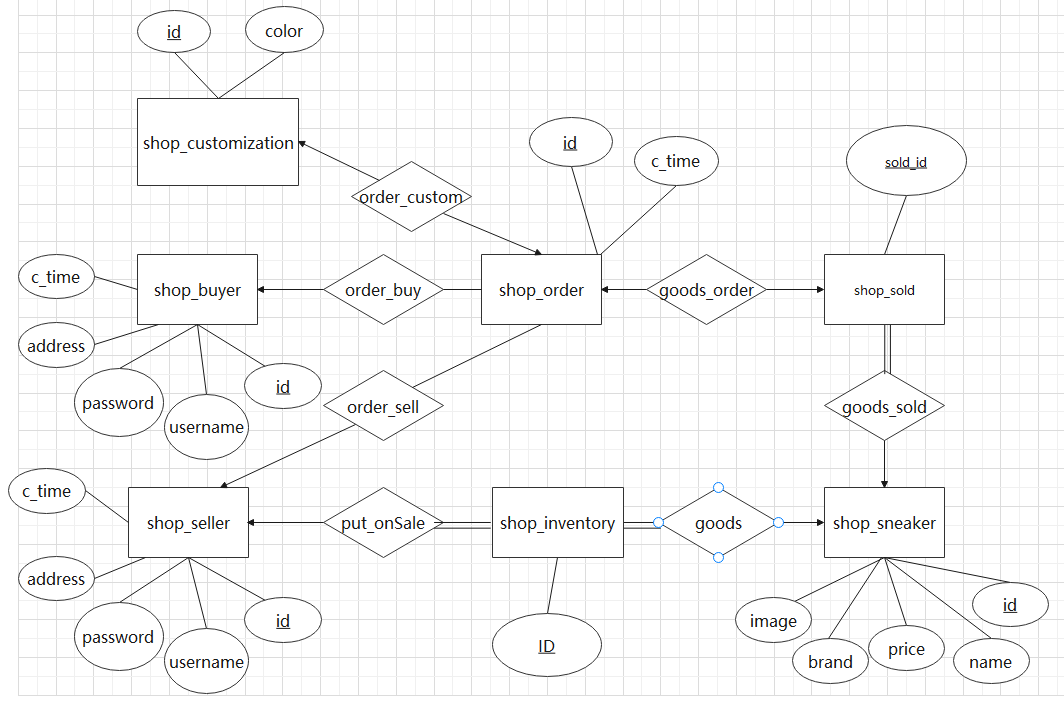
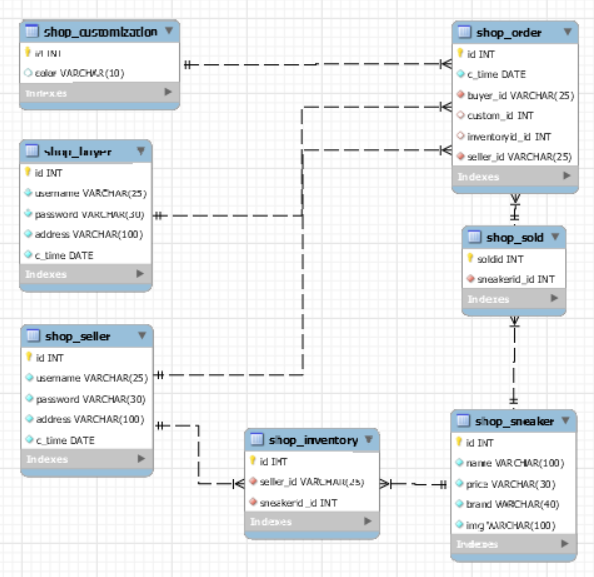
1. One order only points to 1 sneakers, 1 buyer, 1 seller and 1 customization ID.

2. 1 buyer or 1 seller can have many orders. 1 seller can have many sneakers to sell.

4 ER Diagram

4.1 Diagram

The ER diagram of our database is as follows:



4.2 Difficulties and Solutions

To conclude, the ER diagram perfectly implements our assumptions, but there are other entities in our ER diagram such as shop\_Inventory and shop\_sold. These entities are related to our solution of the problem.

The problem is that the direct deletion and insertion in the previous tables we assume (shop\_seller, shop\_sold, etc.) may lead to a searching problem. If we buy the goods, the data in shop\_sneaker will be directly put into shop\_order and deleted from the table, and this will cause a problem that the information of one sneaker could not be searched if it is sold out, since there are no more information about this sneaker in the table.

So in the ER diagram, we add 2 new tables: shop\_inventory and shop\_sold to take down all the information about the sneaker on sale and sold. Now shop\_sneaker only store the basic information of each kind of sneakers, and the 2 new tables store the ID of one pair of sneakers and we do the deletion and insertion in these 2 tables.

5 Functional Dependencies

shop\_Inventory= {id, seller\_id, sneakerid\_id}

F1={id → seller\_id, sneakerid\_id}

shop\_sneaker = {id, username, price, brand, img}

F2={id → username, price, brand, img}

shop\_sold = {id, sneakerid\_id}

F3={id → sneakerid\_id}

shop\_order = {id, c\_time, buyer\_id, custom\_id, inventory\_id, seller\_id}

F4={id → c\_time, buyer\_id, custom\_id, inventory\_id, seller\_id}

shop\_seller = {id, username, password, address, c\_time}

F5={id → username, password, address, c\_time}

shop\_buyer = {id, username, password, address, c\_time}

F6={id → username, password, address, c\_time}

shop\_customization = {id, color}

F7={id → color}

6 SQL Codes and Explanations

Here are all the codes of creating database tables:

**shop\_sold:**

CREATE TABLE `shop\_sold` (

`soldid` int NOT NULL,

`sneakerid\_id` int NOT NULL,

PRIMARY KEY (`soldid`),

KEY `shop\_sold\_sneakerid\_id\_53df1e5c\_fk\_shop\_sneaker\_id` (`sneakerid\_id`),

CONSTRAINT `shop\_sold\_sneakerid\_id\_53df1e5c\_fk\_shop\_sneaker\_id` FOREIGN KEY (`sneakerid\_id`) REFERENCES `shop\_sneaker` (`id`)

)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

Explanation:

This is the table to store goods sold. The primary key is “soldid”. About the foreign keys, “sneakerid\_id” references “id” in shop\_sneaker to get related to buyer table.

**shop\_order:**

CREATE TABLE `shop\_order` (

`id` int NOT NULL AUTO\_INCREMENT,

`c\_time` date NOT NULL,

`buyer\_id` varchar(25) NOT NULL,

`custom\_id` int DEFAULT NULL,

`inventoryid\_id` int DEFAULT NULL,

`seller\_id` varchar(25) NOT NULL,

PRIMARY KEY (`id`),

KEY `shop\_order\_buyer\_id\_cffd21d9\_fk\_shop\_buyer\_username` (`buyer\_id`),

KEY `shop\_order\_custom\_id\_5398df5f\_fk\_shop\_customization\_id` (`custom\_id`),

KEY `shop\_order\_inventoryid\_id\_3b5aecb0\_fk\_shop\_sold\_soldid` (`inventoryid\_id`),

KEY `shop\_order\_seller\_id\_550d3e2b\_fk\_shop\_seller\_username` (`seller\_id`),

CONSTRAINT `shop\_order\_buyer\_id\_cffd21d9\_fk\_shop\_buyer\_username` FOREIGN KEY (`buyer\_id`) REFERENCES `shop\_buyer` (`username`),

CONSTRAINT `shop\_order\_custom\_id\_5398df5f\_fk\_shop\_customization\_id` FOREIGN KEY (`custom\_id`) REFERENCES `shop\_customization` (`id`),

CONSTRAINT `shop\_order\_inventoryid\_id\_3b5aecb0\_fk\_shop\_sold\_soldid` FOREIGN KEY (`inventoryid\_id`) REFERENCES `shop\_sold` (`soldid`),

CONSTRAINT `shop\_order\_seller\_id\_550d3e2b\_fk\_shop\_seller\_username` FOREIGN KEY (`seller\_id`) REFERENCES `shop\_seller` (`username`)

) ENGINE=InnoDB AUTO\_INCREMENT=8001 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

Explanation:

This is the table to store orders. The primary key is “id”. All the keys except custom\_id are not null, because customization is an optional service. About 4 foreign keys, “buyer\_id” references “username” in shop\_buyer to get related to buyer table. “custom\_id” points to “id” in customization, “inventoryid\_id” points to “soldid” in shop\_sold, “seller\_id” points to “username” in shop\_seller.

**shop\_sneaker:**

CREATE TABLE `shop\_sneaker` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

`price` varchar(30) NOT NULL,

`brand` varchar(40) NOT NULL,

`img` varchar(100) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=7085 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

Explanation:

This is the table to store sneaker information. The primary key is “id”. Because each id points to a unique name, so name is also unique.

**shop\_seller:**

CREATE TABLE `shop\_seller` (

`id` int NOT NULL AUTO\_INCREMENT,

`username` varchar(25) NOT NULL,

`password` varchar(30) NOT NULL,

`address` varchar(100) NOT NULL,

`c\_time` date NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `username` (`username`)

) ENGINE=InnoDB AUTO\_INCREMENT=5599 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

Explanation:

This is the table to store sellers’ information. The primary key is “id”. Because each id points to a unique username, so username is also unique.

**shop\_buyer:**

CREATE TABLE `shop\_buyer` (

`id` int NOT NULL AUTO\_INCREMENT,

`username` varchar(25) NOT NULL,

`password` varchar(30) NOT NULL,

`address` varchar(100) NOT NULL,

`c\_time` date NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `username` (`username`)

) ENGINE=InnoDB AUTO\_INCREMENT=6001 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

**shop\_customization:**

CREATE TABLE `shop\_customization` (

`id` int NOT NULL AUTO\_INCREMENT,

`color` varchar(10) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5001 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

Explanation:

This is the table to store buyers’ information. The primary key is “id”. Because each id points to a unique name, so name is also unique.

**shop\_inventory:**

CREATE TABLE `shop\_inventory` (

`id` int NOT NULL AUTO\_INCREMENT,

`seller\_id` varchar(25) NOT NULL,

`sneakerid\_id` int NOT NULL,

PRIMARY KEY (`id`),

KEY `shop\_inventory\_seller\_id\_6912b8f3\_fk\_shop\_seller\_username` (`seller\_id`),

KEY `shop\_inventory\_sneakerid\_id\_d7071c66\_fk\_shop\_sneaker\_id` (`sneakerid\_id`),

CONSTRAINT `shop\_inventory\_seller\_id\_6912b8f3\_fk\_shop\_seller\_username` FOREIGN KEY (`seller\_id`) REFERENCES `shop\_seller` (`username`),

CONSTRAINT `shop\_inventory\_sneakerid\_id\_d7071c66\_fk\_shop\_sneaker\_id` FOREIGN KEY (`sneakerid\_id`) REFERENCES `shop\_sneaker` (`id`)

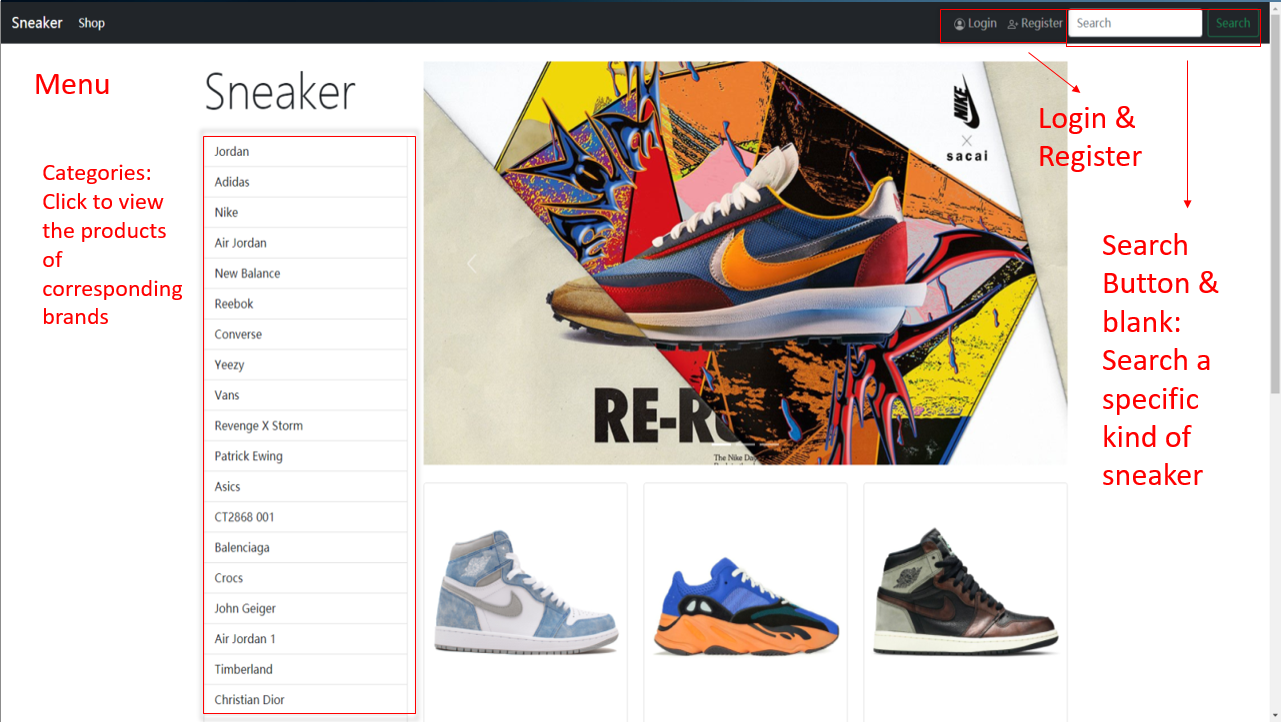
) ENGINE=InnoDB AUTO\_INCREMENT=80001 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

Explanation:

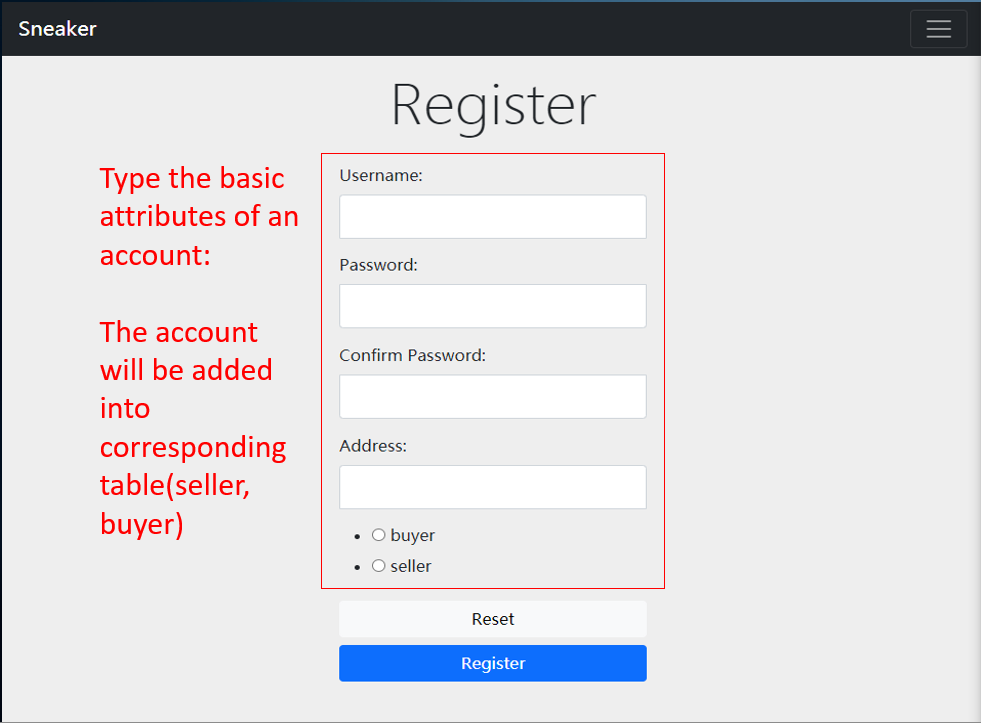
This is the table to store information of sneakers on sale but not sold. The primary key is “id”. About the foreign keys, “seller\_id” references “username” in shop\_seller to get related to seller table. “sneakerid\_id” points to “id” in shop\_sneaker.

7 Website Design

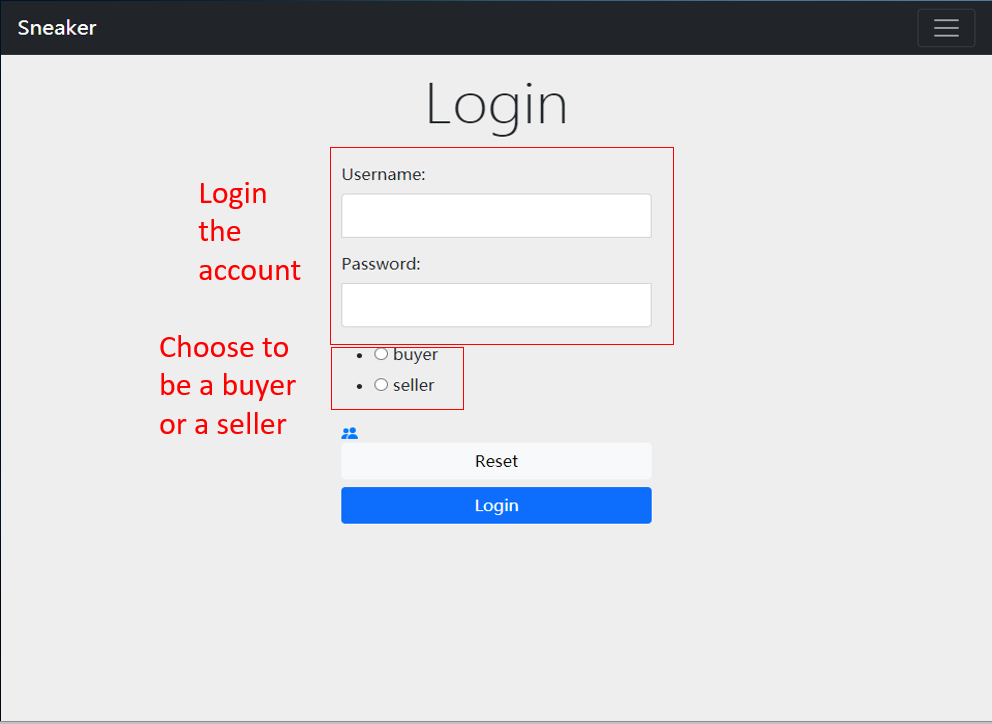
Our website consists of the following parts:

7.1 Main Menu

After entering <http://127.0.0.1:8000/sneaker/> , you get the main menu above. You can click categories button in the left to see the good of corresponding brands, and you can search the product by the blank on the top-right corner. If you want to buy the sneakers, you need to register and log in first.

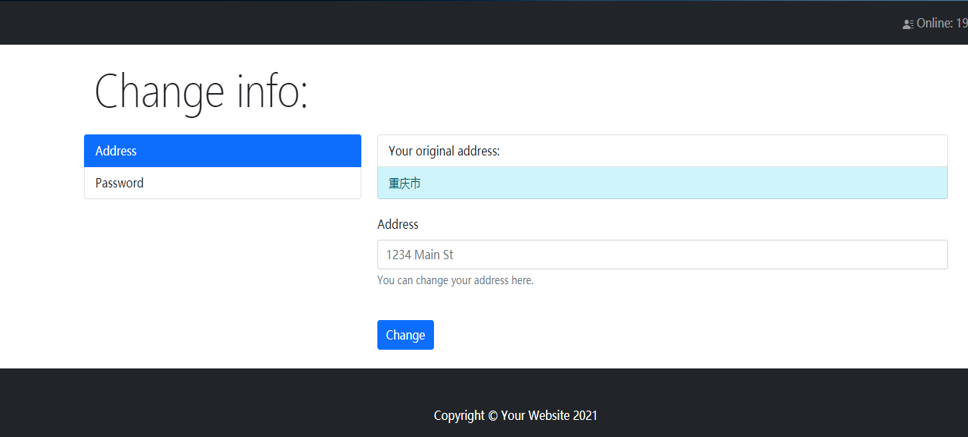
7.2 Login & Register

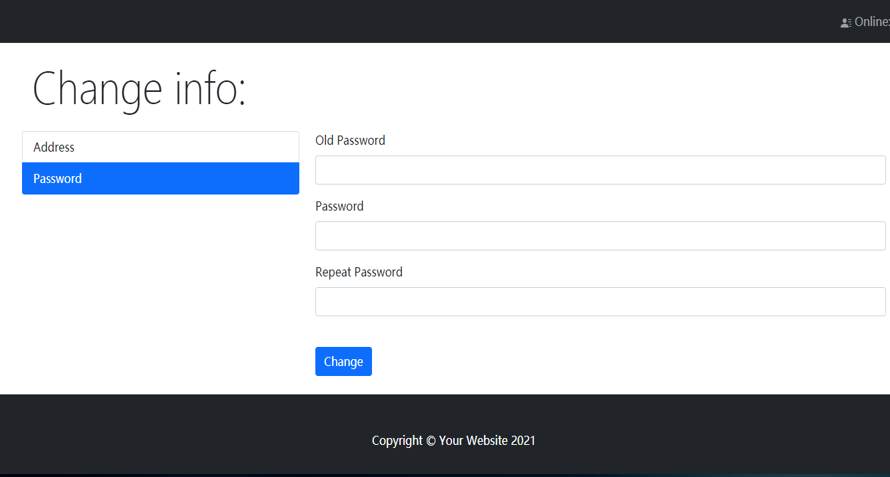
In the registration you need to insert the information needed in the graph above. Then your information will be added to the table shop\_user or shop\_seller depending on what you choose.



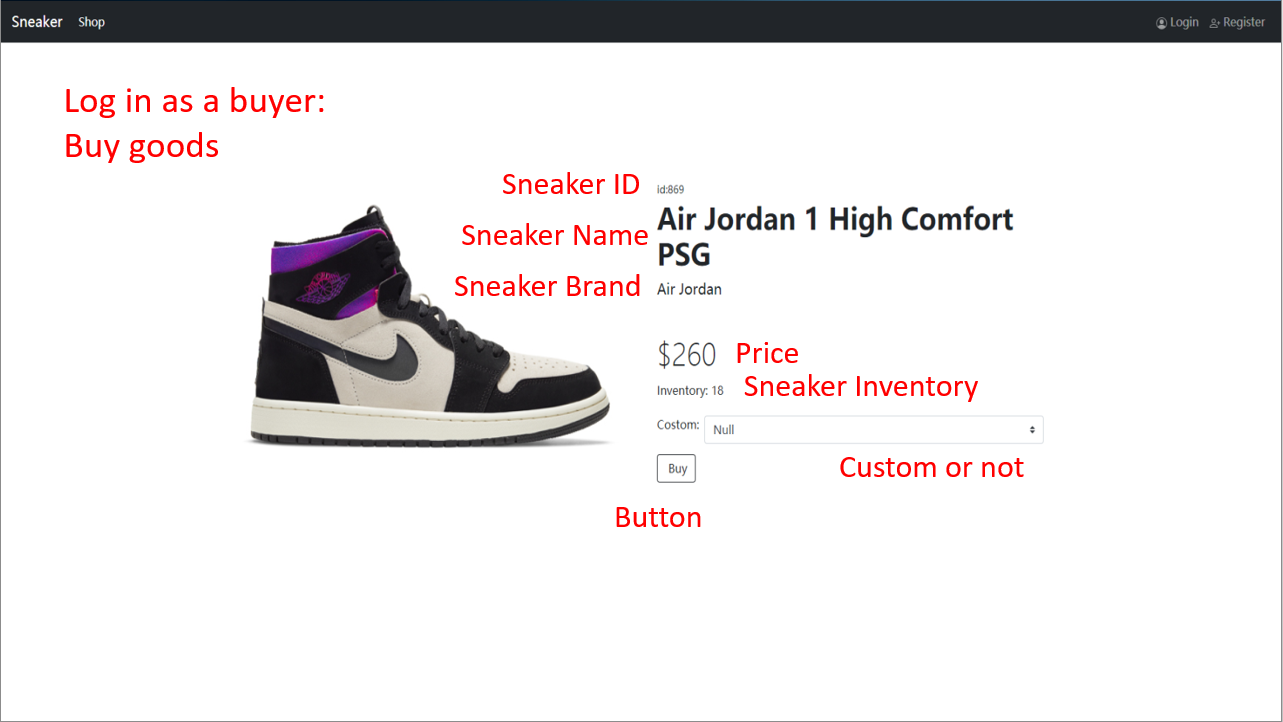
After inserting your username, password and identification, you can log in. If you choose the different identification (e.g. buyer user chooses seller when logging in), you will get a warning.

7.3 Change Information

Of Course, you can change the information in the following page. After inserting new information, the previous data will be updated by new one. It’s the same for both buyers and sellers.

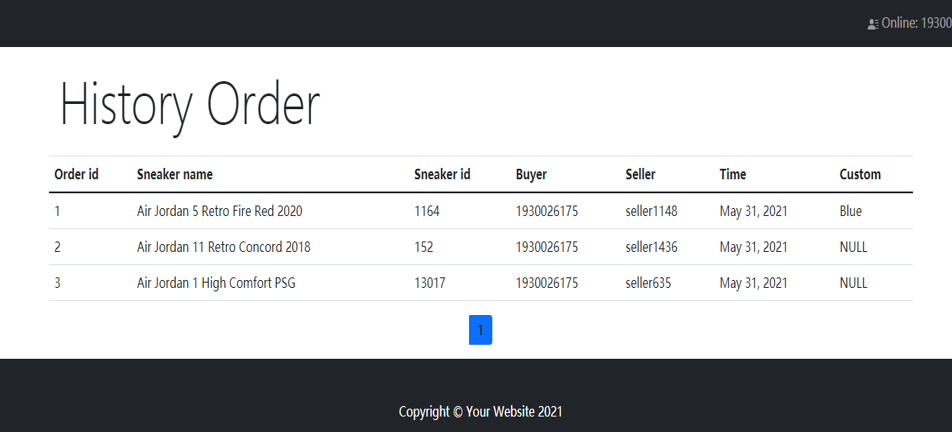


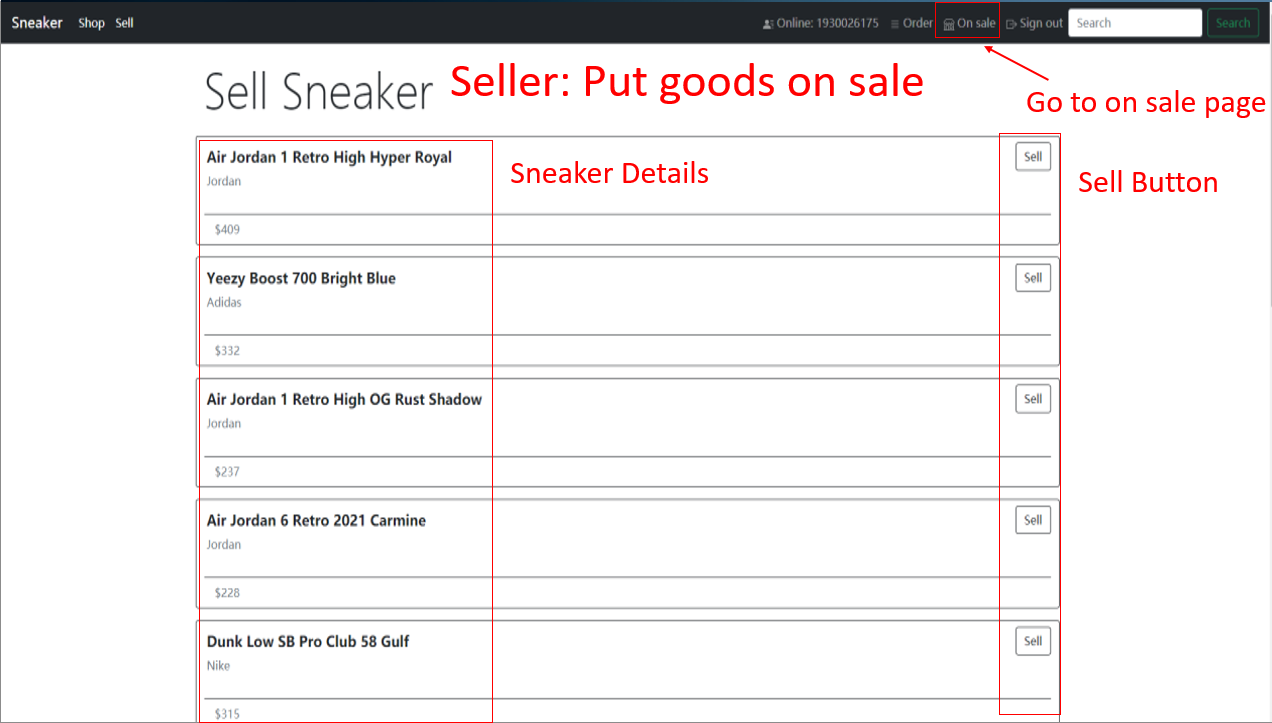
7.4 Buyers: Buying Goods

After logging in as a buyer, you can click into the page of one pair of sneaker to view the basic information and click the button to buy it. The data of the sneaker bought will be put into the history order of your account with an order ID.

7.5 History Order

Buyers can view the previous order in the following page and search for a specific order.

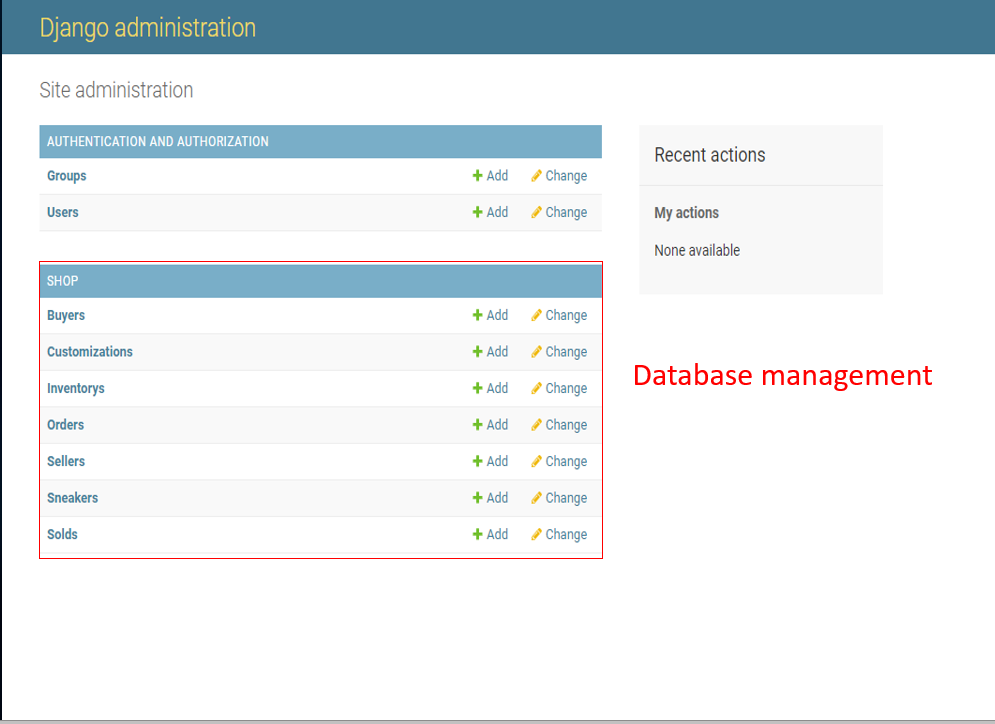
7.6 Seller: Sell the Products

If you log in as a seller, you can view your own inventory sneakers and sell them by pressing sell button.

After selling it, the goods will be put on sale and you can check them in the following page. All the sellers cannot be seen when the buyers buy sneakers only if they buy it. The earliest sneaker put on sale will be sold automatically to buyer.

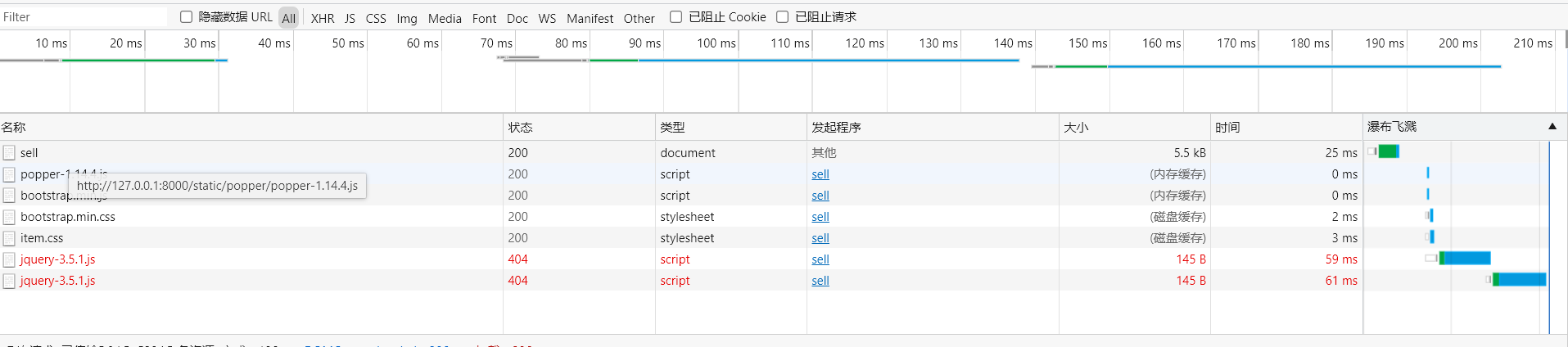
7.7 Administrator page

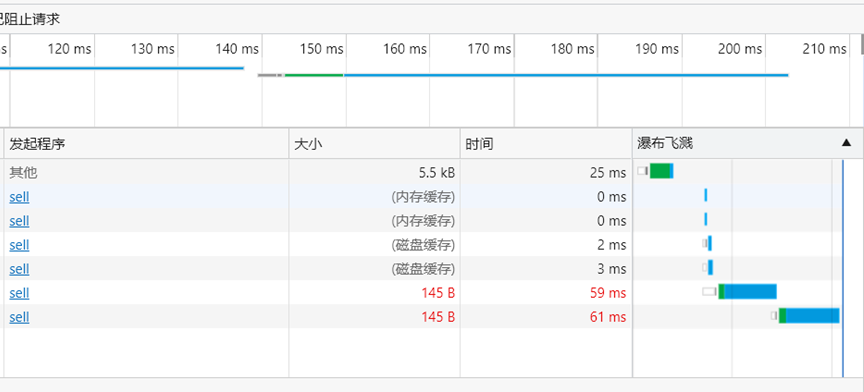
The administrator of the website can log in and manage database here.

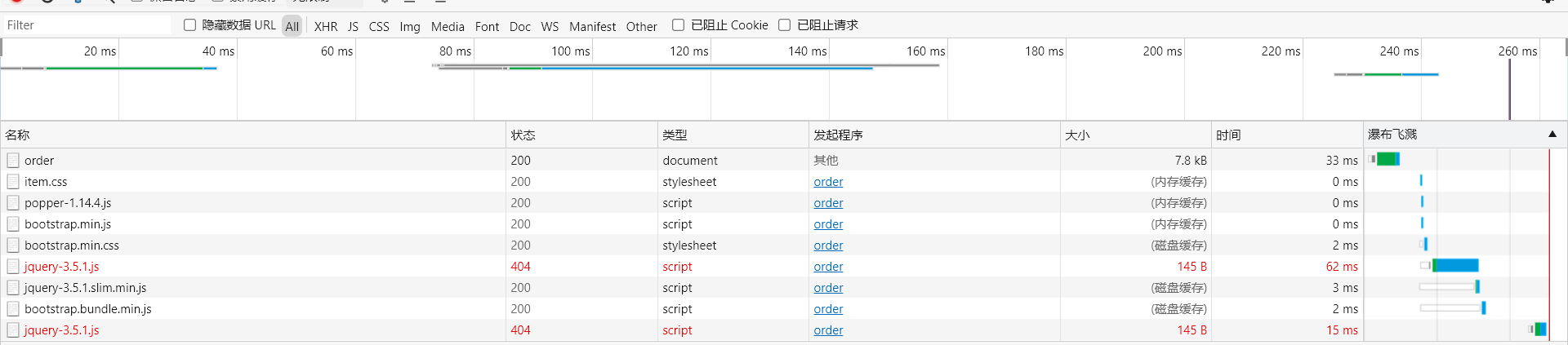


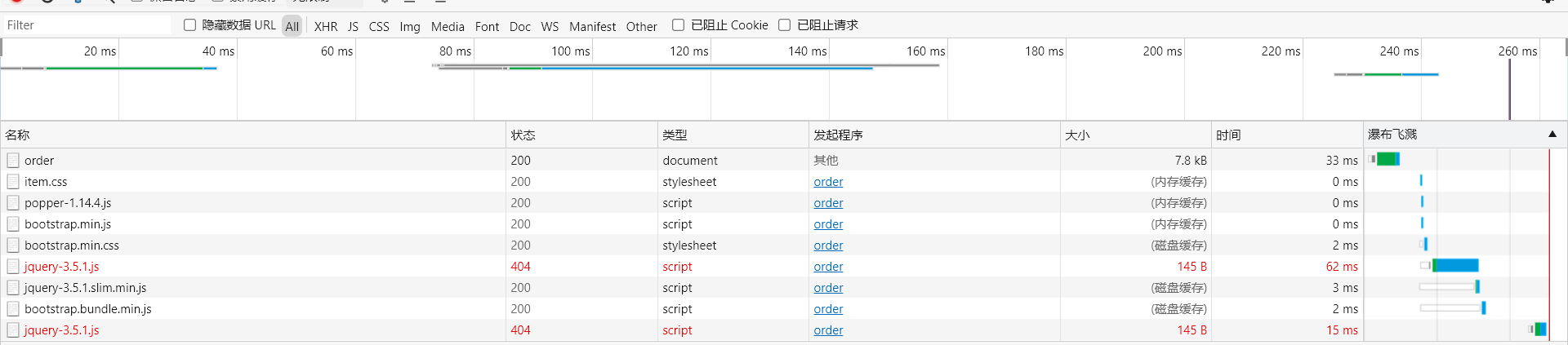
8 Extra: speed of adding, deleting and searching data

Here are the results of the speed of adding, deleting and searching data.

**Adding:**

****

**Deleting:**

**Searching:**

**As shown above, our speed of managing data satisfies the requirement.**

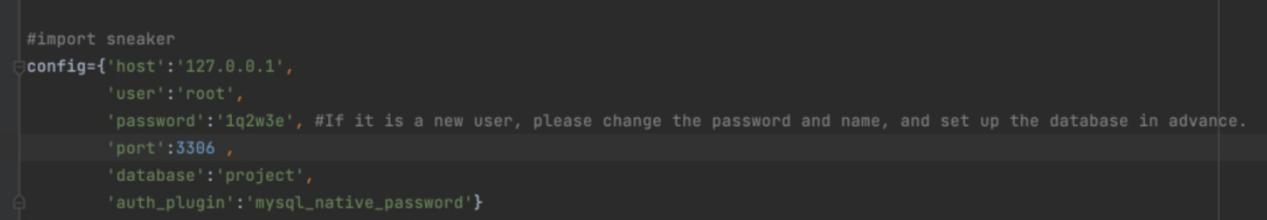
9 Extra: Manual to import the data

1.Modify the settings and set the port to connect to the database.

Documents you need to modify：

In Sneaker/sneaker/settings.py

In Sneaker/img/dataimport.py

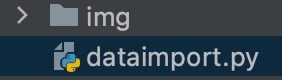


1. Creating models

Run python manage.py migrate to apply those changes to the database.  
 Run python manage.py makemigrations to create migrations for those changes.

1. Read data, import data.

Run dataimport.py



1. After successful import, you can open the website for operation.

Click <http://127.0.0.1:8000/sneaker/>