[redacted]

Group_10_Very

System Documentation

[redacted]

[redacted]

Abstract

Background

Due to the breakout of COVID-19, many customers prefer to buy musical instruments online. A family-owned musical instrument shop, which has been operating offline for years, wants to explore the online market with a new website.

Deliverables

This project completed an online instrument shop website considering both functional requirements and non-functional requirements.

- Functional requirements
  - The website has two portals – customer and staff:
    - Customers can view products, manage shopping carts, place orders and learn about the company.
    - Staff can organize instruments categories, products, orders and the status of the shop.
  - Customers and staff can communicate with each other easily.
  - The website supports both English and Chinese.
- Non-functional requirements
  - Server: The server can support the website stably.
  - Platform: The website can be accessed from both computers and mobile phones.
  - Security: authentication, verification of role, password encryption
  - Reliability: low probability of system failure
  - Maintainability: The code is modular and reusable.
  - Usability: The website is easy to use. UI is beautifully designed.

Implementation

This website is implemented by Bootstrap and Jquery for the front-end, Flask framework in python for the back-end, and Sqlite for the database.

Group work

With the help of teachers and teaching assistants, six members of our group divided the work and made progress in solving problems. Problems we met can be divided into technique aspects and collaboration aspects.

- Technique problems: Technique problems were almost solved by learning from the Internet. If it was too difficult we would replace them with other methods.
- Collaboration problems: In terms of cooperation, we have encountered problems such as conflicting ideas and inconsistent progress. This requires adequate communication. We tried to hold offline meetings once a week. Even when the pandemic was serious again, we communicated through online meetings. Wechat groups have also been made full use of to share ideas and progress. We also took a prototype approach to align the group's view of the task.

Keyword: Online instrument shopping, Website development, System documentation, Group work, COVID-19

Contents

1 Introduction

1.1 What this system achieve

The purpose of this project is to establish an instrument shopping website which contains two portals, including a customer portal and an employee portal. In this website, costumers can make orders and get their orders directly delivered to their doors or pick up the orders in a particular store, so that they are not required to be exposed to the risk of the pandemic.

- Customers can
  - Search, view and purchase all products for sale.
  - Manage shopping carts, place orders and choose the way they get their goods (deliver/pickup).
  - See order status and modify those are not completed yet.
  - Send messages or products to staff.
- Employees can,
  - Manage instrument categories, products and the status of the shop.
  - Organize, modify, prioritize and keep track of orders in the system.
  - See and reply to communications from customers.
- The system should
  - Support both English and Chinese, both computers and mobile phones.
  - Protect the security of users' account.
  - Provide an attractive interface to users.
  - Be easy for users to learn how to operate it.

Besides of the functions mentioned above, it has some features which may gain more customers' interests.

- Comment feature provided to the customers help them decide whether a product is really desirable for them, because they can look through those comments given by other.
- Auto-reply in the chatting room can reduce staff's repeated work and allow customers to get answers faster.
- Translation for reviews and messages can help users from different countries to communicate freely, which contribute to globalization and help to attract international users.
- Besides of one of this website's main functions which is that costumer can choose whether they are going to pick up their goods or receive goods at home, customers can modify the orders that they have made as long as the orders have not been finished yet, considering that some information, such as recipient location, is likely to change due to the COVID-19 pandemic. At the same time, the staffs in the website have higher authority, which means that they can not only modify those orders made by customers, but also organize and prioritize them. As a result, staffs can make appropriate adjustments to some orders based on the local restrictions. For example, staffs can delay the deliver of a order when the destination of the order has banned imported goods. Furthermore, staffs can even take down a item sold in the website if it is impossible to import this item.
- Visual data analysis of employees' home pages helps them understand transactions.

1.2 System Architecture

Generally, the development of the website is based on FLASK, which our team also include many other techniques. Basically, the front-end in the website is given by bootstrap, while the services given in back-end are supported by flask. We apply JavaScript to make sure these two ends work closely together. The system architecture diagram is given below.

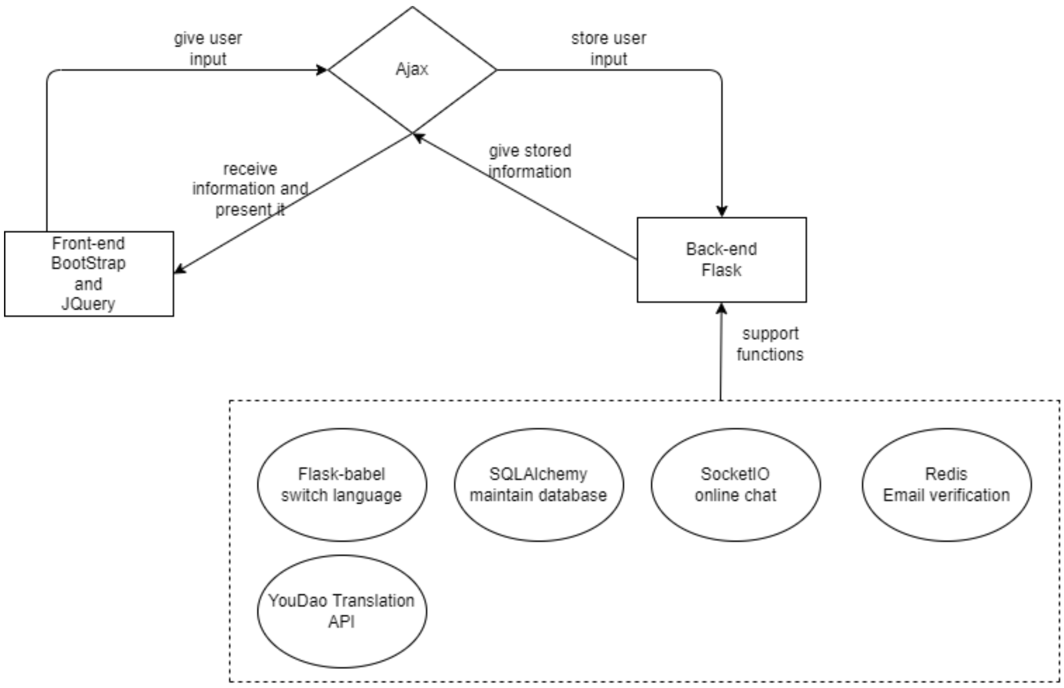

Figure 1: system architecture

1.3 How to access this system

URL: When the user wants to access the project directly over the network, please type IP: [redacted] in your browser. Our website is deployed on UCD server so that it could be easily tested.

Accounts: There are 2 suggested testing accounts, one for staff, the other for customer:

- Staff Account
  - username: 1234567
  - password: 12345678
- User Account
  - username: bjjjjjj
  - password: bqhbqhbqh

In addition, you can register for your own customer account for testing the authentication functionality. (detail in register part)

2 Group Work

2.1 How we work together

Our project has a clear division of labor between the front end and the back end. We divide the members into several groups, and the members of each group can cooperate well. We have a meeting once a week to assign everyone's tasks. These tasks are reasonable and orderly, which has well promoted the process of the project. On the connection of front end and back end, our team members will work together and cooperate with each other to complete the work.

2.1.1 Division of labor

Our team consists of six team members, each with their own responsible group, and each group will be responsible for a part of the whole project. Through the cooperation of each group, the project can advance steadily. Our team is divided into the following six groups:

- **Front-end Group**
  Group Manager: [redacted]
  Team Members: [redacted]
  Front-end Group is in charge of the front end of the website. Make the website attractive for customers. Make the website clear and convenient for staves. Send data in front end to back end and present information according to data from back end.
- **Back-end Group**
  Group Manager: [redacted] Team Members: [redacted] The aim of Back-end Group is to write the back-end, set up a database, connect the front end with the back-end, and realize the jump function and the functions required by the web page.
- **Customer Group**
  Group Manager: [redacted] Team Members: [redacted] The aim of Customer Group is to analyse user requirements, define acceptance tests, get feedback for each iteration and conduct a continuous user evaluation of the website
- **Leading Group**
  Group Manager: [redacted] Team Members: [redacted] The aim of Leading Group is to analyze the provided requirements, and to give specific directions for the project, such as the content of each weekly meeting, a refined plan for each delivery.
- **Test Group**
  Group Manager: [redacted] Team Members: [redacted] The aim of Test Group is to test the whole code and record the noticeable outcomes of executing.
- **Documentation Group**
  Group Manager: [redacted]g Team Members: [redacted] The aim of Documentation Group is to make preparations for project document like weekly update, overleaf report and coordinate the work of each member.

2.1.2 Weekly Process

We will release the team members' work this week on Monday and have a group meeting on Tuesday. Before Tuesday's meeting, team members can find information to estimate the difficulty of their tasks. At the same time, this means that when the team members have personal situation or feel that the task is relatively difficult, they can put forward their own situation and change their tasks in the meeting. Sunday is our task submission period. If a team member fails to complete his task on time, his contribution score will be deducted.

- **Meeting**
  Our meeting is mainly divided into four parts: the first part is to explore the details of this week's task and understand the ambiguous places. The second part is to evaluate the tasks of each team member. If it is too difficult, the tasks can be reduced appropriately, and some team members can also put forward other special circumstances. The third part is to solve the difficult problems encountered, and solve the problems by discussing together and asking the teaching assistant. The fourth part is to improve according to the feedback. We will discuss it uniformly and correct what we did wrong.
- **Process** For the process side, we work to a certain process each week. Our weekly task process is shown in the Figure 2. What the researchers found to be critical to team productivity was understanding the work of teammates (Amy, 2022). Since we conducted weekly meeting that clearly showed each team member's tasks, the entire group had a good understanding of everyone else's tasks. As a result, our progress throughout the project was steady and fluent.

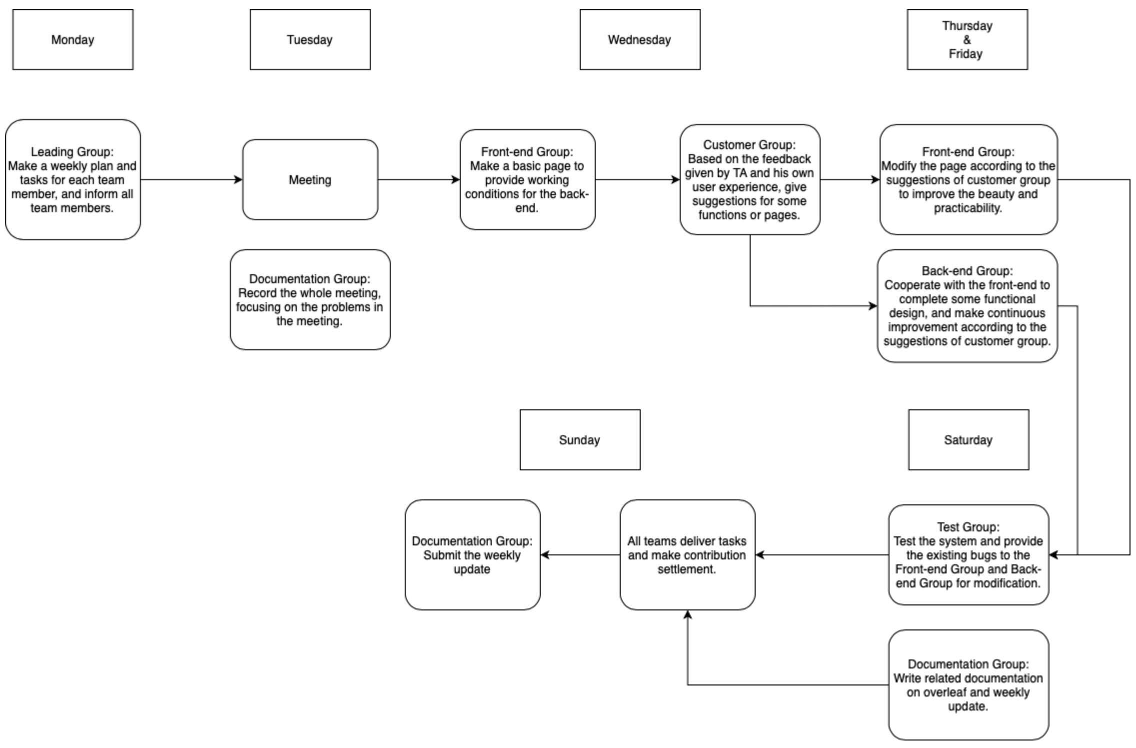
Monday
Tuesday
Wednesday
Thursday & Friday
Leading Group: Make a weekly plan and tasks for each team member, and inform all team members.
Meeting
Documentation Group: Record the whole meeting, focusing on the problems in the meeting.
Front-end Group: Make a basic page to provide working conditions for the back-end.
Customer Group: Based on the feedback given by TA and his own user experience, give suggestions for some functions or pages.
Front-end Group: Modify the page according to the suggestions of customer group to improve the beauty and practicability.
Back-end Group: Cooperate with the front-end to complete some functional design, and make continuous improvement according to the suggestions of customer group.
Sunday
Saturday
Documentation Group: Submit the weekly update
All teams deliver tasks and make contribution settlement.
Test Group: Test the system and provide the existing bugs to the Front-end Group and Back-end Group for modification.
Documentation Group: Write related documentation on overleaf and weekly update.

Figure 2: Weekly Process

2.1.3 Communication

There is an important idea from Amy "A fundamental challenge in software engineering is ensuring that everyone on a team has the same understanding of what is being built and why" (2022). Only by having useful and effective communications can we overcome this challenge from our views. Therefore, we try our best to address issues and conflicts inside our meetings, accompanied with careful discussions about any less clear tasks to ensure that our understanding is consistent. If team members meet regularly with each other to design error compensation together and exchange shared knowledge, their collaboration can be more fluid (Amy, 2022). During our project process, our groups have many frequent communications with each other, such as the front-end group and back-end group sharing their skills and work together to eliminate any errors on some features. Meanwhile, it makes team members have a comprehensive understanding of the whole project.

2.2 Collaboration Problems

2.2.1 Communication

- **Description** Due to covid-19, it is difficult for all group members to get together, so offline meetings that all group members attend become fewer. As a result, our ideas cannot be sufficiently understood by other group members only by sending messages. Additionally, some group members may be confused with their tasks without face-to-face communication.
- **Solution** We have a regular meeting every week, which consists of offline meeting and online meeting by using Tencent meeting. If a group member cannot take part in the offline meeting, he or she must enter online meeting to ensure the attendance. In the regular meeting, we allocate tasks to group members and discuss them. Moreover, we illuminate our own ideas and after the discussion and whether the idea is accepted is decided.

2.2.2 Connection

- **Description** The connection between two tasks is not smooth. Specifically, if at least two group members program for a functionality and a group member's work is based on the task of another group member, the first member may find the finished code whish his work is based on after a long time. In this situation, the time to test the functionality may become very limited.
- **Solution** Every group member must timely declare that the task is completed in the Wechat group when he or she finishes the work. If his or her tasks is the base of other group members' tasks, then he or she notifies related group members of beginning to program.

2.2.3 Conflict

- **Description** There are some conflicts in our group. What most cause these conflicts are different technologies to implement a functionality. Another significant reason is the different ideas about the plan of the implementation of functionalities.If conflicts fail to be resolved by an appropriate way, them may reduce the efficiency and the morale of all group members (Indeed, 2021a).
- **Solution**
  There are many skills to resolve conflicts, such as active listening and perspective taking. When a conflict arises, everyone in the group states own opinions and attentively others' views. Furthermore, who result in the conflict need to thought about why the other group member put forward that viewpoint, which may contribute to the improvement of the project (Indeed, 2021b).

  As for two main conflicts, there are four criteria to evaluate the technology or the idea. Whether the performance of a functionality meets the need for the requirement by using the technology. Whether the performance of a functionality is beyond the need for the requirement by using the technology and the technology costs too much time. Whether the functionality is in the main process. Whether the functionality is from the extension of the requirement.

2.2.4 Programming Capability

- **Description** Group members have different levels of the programming capability and are good at different aspects of the web development. If a task is allocated into an inappropriate member, the efficiency of our project will reduce.
- **Solution** In a sprint, group members often work in small teams, which consist of a front-end member and a back-end member. Group members with strong programming capability are responsible for more coding tasks while others are in charge of more tasks about documents.

2.2.5 Database

- **Description** Our group can be roughly divided into three small groups, which means that three members in charge of back end can operate the database. If two members modify database at the same time, there is a conflict when merging. Furthermore, one member needs to rollback the database in his storage, which has negative influence on the efficiency of the development.
- **Solution** Before modifying the database, a group member who wants to operate the database must notify other group members in the Wechat group. After modifying, the group member needs to notify other group members of that he or she finishes operating the database and reminder them to pull.

2.2.6 Dramatic Deterioration of The Pandemic

- **Description** In late April, the situation of COVID-19 pandemic begins to dramatically deteriorate in Beijing. A large number of tables gathering information need to be filled in. Additionally, we must participate in nucleic acid tests again and again. Moreover, there are many calls to investigate where we visit. All of these make us tired and reduce the productivity of the project.
- **Solution** In this circumstance, we need to keep an optimistic mentality and a healthy body, which contribute to the development in a normal progress.

2.3 Technical Problems

2.3.1 Test

- **Description** In the development, we usually only simply test every functionality. To be specific, a majority of functionalities are tested independently at most three times. As a result, there are many bugs that we do not find. Especially in week8, some bugs cause that we have to record videos again and again, which wastes so much time.
- **Solution** Before declaring finishing a functionality, the group member must ensure there is no bug in the unit test. After integration, we conduct comprehensive tests, such as integration test and system test.

2.3.2 Template

- **Description** Our project integrates several templates, so there are many conflicts resulting from JavaScript and Css. In addition, it is hard to know which attributes really impact the style of a widget when we want to change its style. Finding the reason causing these problems costs too much time.
- **Solution** The customer portal and the staff portal use two base.html. In the base.html, JavaScript and Css are imported as few as possible. The console is an efficient tool to help us find the cause of conflicts and what affect the style.

2.3.3 Repeated Work

- **Description** After a functionality is finished, a better technology may be found. If we want the functionality with higher performance, the code about it needs to be rewritten, so we will spend extra time.
- **Solution** Before we begin to implement a functionality, we read some blogs and articles to find feasible technologies.

3 Functional requirement implementation

3.1 Customer

3.1.1 Products display

1. **The layout switch and quick view**
   - **User story**
     As a customer, I want to change the layout of the products and quick view their details, so that I can have better experience on viewing products.
   - **Design idea**
     A shop page is created to show all of the products. On this page, users could use their preference to choose the pages, the number of products per page, and grid/list layout for themselves. In addition, they could see more details on the shop page rather than entering a new page.
   - **Technology implementation**
     Actually, we use the front-end template to make our pages more clear and beautiful, while the back-end and some logic requires our own effort. Overall, we create three routes for displaying products and rendering the same .html file, according to all products, region and category condition. Firstly, the pagination functionality uses SQLAlchemy.paginate and the html elements to achieve pages and the number of products.

     Secondly, in terms of the layout of products, the template gives us grid and list layouts in different .html files. Next, we used ajax to combine them into a single page so that there is no need to go to a new page. Listening to the click event on switch icon, ajax toggle and mark them when the event trigger so that the URL parameter can contains this and in the next page the layout will follow the last page.

     Finally, we create modals for each product by using the template front-end file. Each modal contains the detailed information of that product, which saves user a great amount of time and is easy for users to continue watching products.
2. **Searching products (support Chinese)**
   - **User story**
     As a customer, I want to be able to search and filter products according to some keywords or price, or etc., so that I can find what I need easily.
   - **Design idea**
     Some specific products matching the conditions (even the combination of them) will be displayed.
   - **Technology implementation**
     Price and keyword filter are inside the sidebar to post search request in the html page. Putting the flask.global.request which contains the URL parameters as a parameter, each route uses search function we defined. The search function leverages several condition statements to judge which filter is using, and even the combination of different filters to give the related products for users. Besides, since jumping to the next page is likely to miss the URL parameters, we selectively add the URL

parameters into the hyperlink to achieve querying the fore-mentioned information. Considering that Chinese use the website, we use _or to query both English name column and Chinese name column in the database.

3.1.2 Communication with staff

1. **Sending questions (including history)**

   - **User story**
     As a customer, I want to ask questions to staff, so that I can learn more information I am concerned about.

   - **Design idea**
     When starting to chat, the customer is put into a separate chatting room, where there are only the customer and staff. After inputting, customers press the send button to trigger the socket to transmit messages.

   - **Technology implementation**
     There are several ways of implementing communication function.

     HTTP is a one-direction network protocol. After the connection is established, only the Browser/User-Agent request resources from the WebServer, the WebServer can return the corresponding data. The WebServer cannot actively push data to the Browser/UserAgent. In this case, the only way to implement real-time data transfer between Browser and WebServer is through AJAX polling.

     WebSocket is a two-direction communication protocol. After the connection is established, both WebSocket server and Browser/UserAgent can actively send or receive data to objects, just like Socket. The difference is that WebSocket is a simple protocol that simulates sockets based on the Web.

     Socket.io separates the data transmission part, encapsulates WebSocket and AJAX polling, and forms a set of APIs. It can shield details and compatibility problems, and realize bi-directional data communication across browsers and devices. For this reason, Socket.io is chosen to implement the chat function.

     First of all, 'flask_socketio' and 'socket.io.js' need to be imported into .py file and html file respectively. The emit() function is used to send the message, and the on() function is used to receive the message, with the first argument indicating the name of send and receive events. For example, when a customer enters the chatting room, for .js part, message is sent from the front-end to the back-end.

     ```
     socket.emit('send msg', {user:username, room:room, message:msg})
     ```

     For .py part, message is received in the back-end by the on() function and sent to all users in the specific room by the emit() function.

     ```
     @app.socketio.on('send msg')
     def handle_message(data):
         room = data.get('room')
         app.socketio.emit('send msg', data, to=room)
     ```

     Then, the message is received in the .js part by the on() function and added into html by manipulating html elements in the following function.

     ```
     socket.on('send msg', function(data){...})
     ```

     However, socket.io just supports real-time communication, which means that customers cannot see previous messages next time they enter the chatting room. Due to this, a new table called 'chat' is created in database. When the message is received in the route `@app.socketio.on('send msg')`, message, time, chatroom and user_id will be stored in database. When a user enter the chatting room, history messages in the database are queried according to the chatting room, returned and displayed in the front-end first. The history of products will be introduced later in the implementation part of 2.

2. **Sending products**

   - **User story**
     As a customer, I want to send products to staff, so that I can make it more clear to explain my questions about this product.

- **Design idea**
  There is a button in shop page and product page, which can be clicked to send the information of this particular product as well as its image to the staff.
- **Technology implementation**
  When pressing the button, the product_id will be sent to `@main.route('/chat/id=<pro_id>')`, which is a new chat route. Then it will query information about the product and return the information of the product to the front-end. The information is displayed in a chat-message div following the connect info display.

  Products should also be stored in the database, so a new column called 'message_type' is added in the table. The product is identified with 0, while word message is identified with 1. If it is a product, the product_id is stored in the 'message' column.

  However, there is an issue that the product_id can be got in the front-end by returning the message query result, but other information of the product cannot be queried directly with jinja2 language. In this case, products need to be stored in a separate dictionary and then return to the front-end.

3.1.3 Personal space

To summarize the purchasing process, there are overall three main steps. Firstly, you need add your desirable products into your cart, and then select those you are ready to buy. Finally, after confirming the recipient information, you can get ready to receive your goods. The flow chart below3 will illustrate how the process move through in detail.

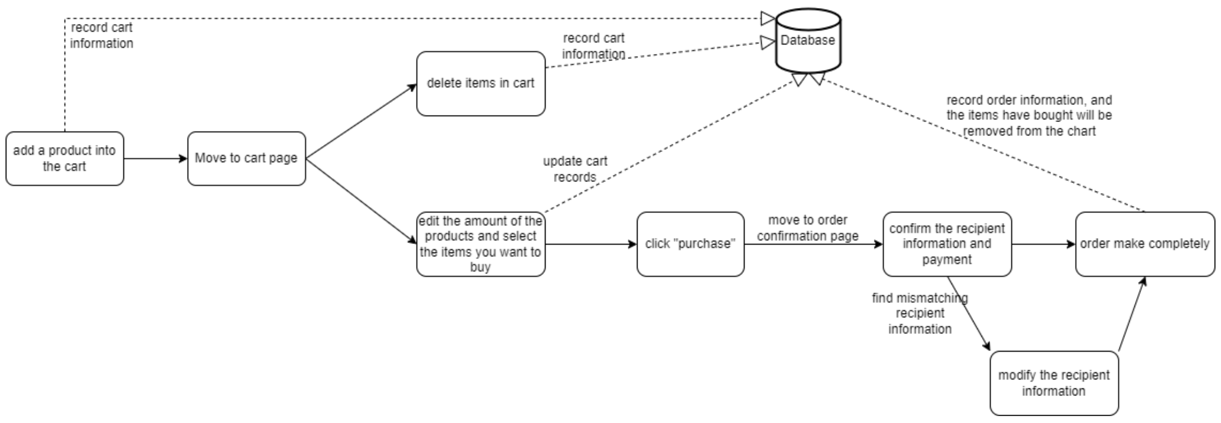
record cart information
Database
record cart information
delete items in cart
record order information, and the items have bought will be removed from the chart
add a product into the cart
Move to cart page
update cart records
edit the amount of the products and select the items you want to buy
click "purchase"
move to order confirmation page
confirm the recipient information and payment
order make completely
find mismatching recipient information
modify the recipient information

Figure 3: Purchasing Flow Chart

1. **Register and log in**
   - **User story**
     As a customer, I want to create an account and log into this website with this account, so that I can use the services from this website.
   - **Design idea**
     A form will be provided to user when creating an account. There are several rules for different input fields to check if the input from users meets the requirements. Once the user submit the form with all information accepted, this account will be created into the database.
2. **Viewing profile and updating personal information**
   - **User story**
     As a customer, I hope that I can look through my profile and update my details if necessary. Therefore I can access my information and make sure the information is correct.
   - **Design idea**
     In the profile page, we design a logical layout for presenting the personal information. Particularly, we give higher priority to some places where the customer can see and manage their orders.
3. **Adding products to the shopping cart**
   - **User story**
     As a customer, I hope I can add the products to my cart, so that I can buy all the desirable products

at once instead of buying them one by one.

- **Design idea**
  Users can add the products into their carts in multiple ways,such as adding items in detail page of each product or in the page showing total products. We design these different patterns because user will take much more consideration buying music instruments than their accessories where user will mostly add products to the cart in the detail page to access more connected information for the former situation and add them in the general page for the latter situation.

4. **Manipulating products in shopping cart**
   - **User story**
     As a customer, I want to manipulate the items in shopping cart such as changing the number of some products to buy, delete some shoppings.
   - **Design idea**
     There is no doubt that user has the demand to change the number of the products in their shopping cart or delete some items. We designed that there are two buttons beside the number of products where the left one for decreasing the number and on the right for increasing the number so that user can freely change the number they want to buy. Similarly, there is also one column having an delete button for delete each shopping.
   - **Technology**
     The implementation of changing the number is by setting the 'change' event handler of the number input. When user changing the number of each product, it will sent a post request to flask server by using ajax with the number and id of this product. If it changes successfully, a success server code will be sent back to front end, where the number will be changed firstly by using js to show on the html. The control flow and data flow of deleting event is similar.

     There is another useful technique which makes a div having the functionalities of selet_all, delete_selected and purchase fixed at the bottom of the window when the actual position of this div is outside of the window. It is implemented by using the IntersectionObserver, which observer this div. If not intersected with the window, change the style of the div to fixed while remove the style if intersected with the window.

5. **Making an order**
   - **User story**
     As a customer, after I selecting the products that I want to buy in my shopping cart, I should be able to make an order in which I can fill my recipient information.
   - **Design idea**
     At first, the consumer can directly making an order after selecting their desirable products without filling recipient information. We later realize another page that gives customers chances to edit their recipient information and provide a check to the products they are buying. This kind of 'middle' page will enhance user experience.

6. **Viewing orders and change the status of the order**
   - **User story**
     As a customer, after I making an order, I hope I can view my all orders in my personal page so that I know which order I have made and check if the recipient information is correct for each order.
   - **Design idea**
     Every time we create a new order in the database, we make a connection between it and the consumer. As a result, when the consumer open his or her personal page, the order information can be retrieved to present.

7. **Cancelling and deleting an order**
   (We changed and optimized this function to users in the the final version)
   - **User story**
     As a customer, I wish I can cancel an order in time if the order is not been delivered, and delete an order is it is useless, so that I can take remedial action if the made an wrong order.
   - **Design idea**
     There are two kinds of operation that can remove the record of an order. One is that cancelling an order because customer make an order that he do not want to by mistake. The other is that the

record of the order is useless to user. These two different situations should be handled by different mechanisms.

- **Technology implementation**
  At the front end, we use different icons to illustrate these two steps. A symbol of 'x' is considered as cancelling while a symbol of trash can is considered as deleting. More importantly, we set a 'mouseover' event to show a phrase of 'cancel order' and 'delete order' beside the corresponding icons, and the 'mouseout' event to hide these indications.

  At the back end, when these two buttons are clicked, there are two different events are triggered. For cancelling order, the status of the order will be changed into exception, waiting for staff to handle. Then the status on the page will be changed by using js when the post request gets a success response. More importantly, this ajax request should be sent within the condition that the current status of this order is 'Paid', which is implemented by simply comparing the string. Similarly, deleting order has the similar limitation that the current status of the order should be 'Finished'. Deleting is relatively easy because it only needs to delete it from database and the front-end DOM tree.

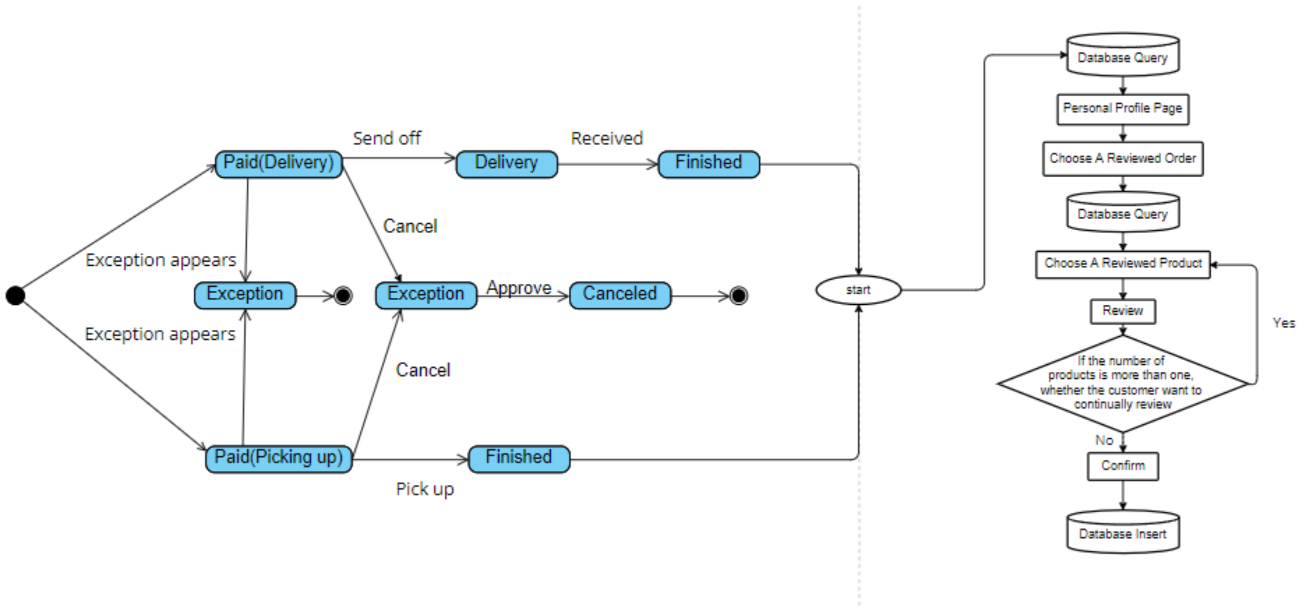

Figure 4: Key steps of order management

8. **Adding comments**

- **User story**
  As a customer, I wish I can add a comment to the products that I bought in an order so that I can share my shopping experience and evaluation to this products

- **Design idea**
  We design that customers can add comments to the products in the order that is received by them. When users want to write the comment, a modal will be shown for them, where they can choose a star level between 0 and 5, and input the comments to every product in the order.

- **Technology implementation**
  We thought the hardest part in the implementation of this functionality is on the front-end. Every time users click the 'add comment' button, a post request will be sent to the server by using jQuery to get the details of products in this order. Then the names of products are displayed in the divider shown at the left side of the modal as different 'name buttons'. Because the content of the modal is dynamically generated by calling $().html and put into the divider, we combined the product_id and order_id as the id of each 'name button' element to identify different star level and comment that user has given towards this particular item.

  The reason why we uniquely identify each 'name button' is that the star level and comments of each product should be distinguished and we use the DOM 'sessionStorage' to realize it. There are two reasons that we use 'sessionStorage'. One is that this storage method is achieved as key-value form so that we can simple make the id of each 'name button' as the key, the star level or comments as the value to temporarily preserve the details that user input, which is a great convenience for user if switching to writing comments for other products. Actually, these preserved content will be displayed in an divider labelled as 'previous comments' beside the input field. The other reason is that we can simply get the json that contains all the details user inputted in this page by loop through 'sessionStorage', which can be easily send to flask server by ajax to get a further processing.

3.1.4 Language switch

1. **Fixed strings language translation (e.g., caption, notification)**
   - **User story**
     As a Chinese/English customer, I want to the website localization, so that it is easy for me to go shopping.
   - **Design idea**
     A wider range of potential customers means different languages that they speak. Considering the cost of making different language directory to render, and the volatility of the API translation, we mainly use flask.babel to localize our website. By setting `@babel.localeselector`, fixed strings can be translated into the another language that we translate by ourselves.
   - **Technology Implementation**

     When users choose which language they like, they go to a new route that changes the cookie of locale, and refresh the `@babel.localeselector` function which localizes the strings on the website if and only if the user firstly enters the website or we refresh it. Next, flask.babel is going to translate the fixed strings we have complied for.
2. **Category name, product name and description localization**
   - **User story**
     As a Chinese/English customer, I want to know the product name in Chinese/English, so that the information is clear.
   - **Design idea**
     Since flask.babel could only solve the fixed strings, we add the additional Chinese name and description column in the database as a staff. By setting cookie of locale and the babel selector simultaneously, product name, category can be translated into the another language that we translate by ourselves. According to the cookie, we use the if statement to determine which column — Chinese or English version — we will query.
   - **Technology Implementation**
     As we mentioned above1, cookie and locale selector are simultaneously set. Rendering .html file page, we use jinja2 language to check which locale the cookie of locale belongs to.
3. **Review and communication translation**
   - **User story**
     As a Chinese/English customer, I want to watch comments and communicate without the language obstacle.
   - **Design idea**
     Since these data is hard to translate in real-time by human, we apply API to translate these dynamic messages.
   - **Technology Implementation**
     There is a button beside each message with a class called 'chat-message__tools'.When it is clicked, The corresponding text is obtained with Jquery, and sent to the back-end with ajax.

     Then it is translated with the help of Youdao Translation website.

     ```
     r = requests.post(url, data=data)
     answer = r.json()
     ```

     The contents that need to be posted is recorded in 'data' in the form of a dictionary. 'post' takes two parameters, a URL and a data, and returns a Response object. The translation results are stored in `answer['translateResult']` in the form of dictionaries or lists, and can be got through loop to form the final result.

3.2 Staff

3.2.1 Management of categories, products and orders

1. **Manipulating categories**

   - **User story**
     As a staff, I want to be able to add categories and delete some of them, so that I can change them flexibly.

   - **Design idea**
     After clicking on the page of adding instrument types, you can select a category (Chinese / Western), or manually enter a new instrument type. When deleting a category, you can select any kind of musical instrument from the drop-down box and click Delete to delete it.

2. **Manipulating products**

   - **User story**
     As a staff, I want to add new musical instruments, delete and modify some of them, so that the new products can be put forward, outdated products can be removed, and the details of products can be changed.

   - **Design idea**
     When browsing the products at staff side, each product is displayed as a row in a table. Every time staff add a product, remove a product and modify the details of a product, there will be an ajax request sending these information to the server. In the server, there are different mechanism to handle different situation. In addition, the number and price of musical instruments cannot be set to negative numbers.

   - **Technology implementation**
     There is a widget called 'bootstrap-fileinput' needed to be concentrated on because it is powerful and gives us a considerable optimization on the functionality of changing the images of product.

     We mainly use this widget in two ways. One is deleting the original images of a product, and the other is adding images. This plugin has already encapsulated functionality of sending an 'delete' ajax request and an 'add' ajax request. By setting the 'deletUrl' and 'uploadUrl' are the corresponding ajax requests sent to the route functions with the connected files. All these options can be achieved by calling the function \$().fileinput() with a json parameter indicating all related settings. For example, it provides 'encrypt', 'uploadExtraData', 'initialPreview', etc. We use this 'initialPreview' with an array of paths as value to show the previous uploaded images of the product and staff can delete them by click the cancel button of each image.

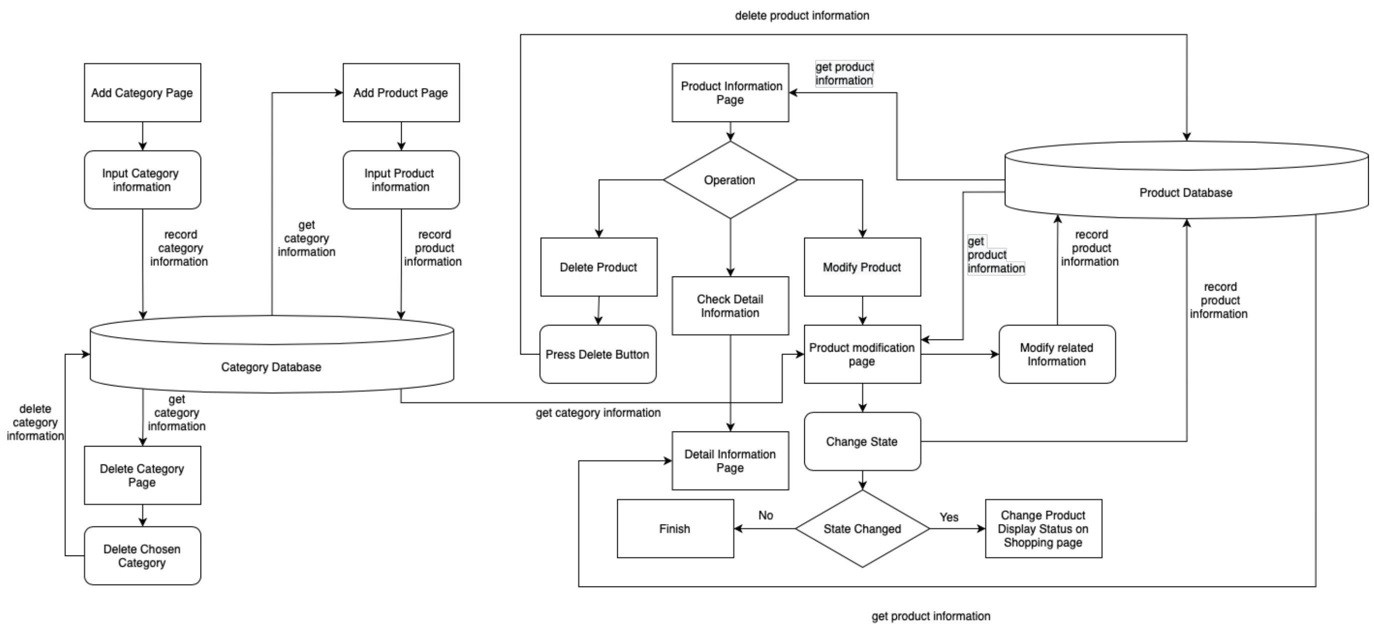

Figure 5: Manipulate Product And Category

3. **Manipulating orders**

   - **User story**
     As a staff, I want to have the ability to modify orders, so that I can handle the problems and mistakes of the orders.
   - **Design idea**
     When the staff click the 'Edit' button, the existing order data will be presented and any segment can be modified, including the status of the order, the receiving name and address of the customer, the phone number and order priority. When the physical store is closed, the status of the order cannot be changed to pick up.

4. **Searching products**

   - **User story**
     As a staff, I want to search for a musical instrument, so that I can find a musical instrument fast.
   - **Design idea**
     After the staff submits the searching product request, our ajax codes will store the content in the searching input field and sent the content to the server. In the server, if there is "#" in the content, we will query the ID in the database. In other situation, we will query the name in the database.

5. **Searching orders**

   - **User story**
     As a staff, I want to have a search order function, so that I can quickly find the order I want to find.
   - **Design idea**
     When the buyer's name or address and other information or information fragments are entered in the input box, the information will be retrieved, and relevant data will also be displayed to the inquirer.

6. Order priority

   - User story
     As a staff, I want those orders that are watched to rank ahead of other orders.
   - Design idea
     When employees set orders as priority orders, these orders will be marked with stars and appear at the top of the order list.

3.2.2 Communication with customers

1. **Answering different customers' questions**

   - **User story**
     As a staff, I want to send my answer to different customers, so that I can give customers more information.
   - **Design idea**
     This is similar to customers' chatting function, except that there is a list of chatting rooms that staff can choose from to alter communicatee.
   - **Technology implementation**
     A list of chatting rooms are queried from the database according to their name. The name is the same as the username to make it unique.

     When staff choose a chatting room, the roomname will be sent to `@staff.route('/chat/<roomname>')`, which is a new chat route. Then it will query information about the chatting room and return the information to the front-end. The message box, which is similar to the customers' chatting room, is displayed in the area on the right.

2. **Auto-reply**

   - **User story**
     As a staff, I want to get help from the system to reply the same customers' questions, so that repetitive work can be lessened.
   - **Design idea**
     Staff can add, delete and change auto-reply questions and answers. When customers enter the chatting room, some questions are displayed. If customers click one of them, the question and the corresponding

answer will be displayed in the message box. The language of communication between staffs and customers is English by default. We could use chat translation to achieve it.

- **Technology implementation**
  There is a new form to collect auto-reply questions and answers. These are stored in a new table called 'answer' in database, and identified by a 'answer_id'.

  When a customer enters the chatting room, three questions are queried and returned to the front-end. Questions are shown in a chat-message div following the connect info display. One of the div's classes is 'answer' and 'answer_id' of the question, and a class called 'question' is also added.

  If one of the questions with class called 'question' is clicked, the 'answer_id' is sent to the back-end through ajax. The answer is queried according to the 'answer_id', and three random questions are also queried. These are returned to the front-end and displayed.

4 Non-functional requirements implementation

4.1 Security

Considering that our website is a shopping website, so there is no doubt that transactions will involve in it. Therefore, we do design some mechanisms to improve our system security in order to counter illegal usage.

4.1.1 Authentication by email

- **Design idea**
  Before a customer finishing his or her registration, the customer is required to input a valid email, as our system will send a verification code to this given email. Then the customer should use the code sending to his or her email to complete the registration.

- **Technology implementation**
  Redis is utilized to store the key-value information about verification code. Specifically, for each registration email, the system will generate a 4 digits random number to produce a map item with the email account as key and the random number as value. Later on, the map item will be stored in the Redis which is somewhat like a temporary repository and will be refreshed in a fixed time. The last step in our system is sending an email containing the random number to the customer's email account. As a result, only when the verification code provided by the customer is exactly the same as that one in our Redis repository can the customer pass the verification.

  Since the Redis repository is not visible for the customer, or client-end, it is only possible for our users to get the correct code by checking their email (if the email account is existed). Thus, the security is improved due to the elimination of fake email accounts.

4.1.2 Verification of role

- **Design idea**
  We also take the problems caused by two portals into consideration. Particularly, some features performed in one portal should not be presented in the other portal. For instance, as a customer, orders prioritizing page should be hidden for them because they have no rights to deal with another customer's order. As a result, we decide that customers can not have the access to employee's pages including orders organizing and products managing. On the other hand, customer-exclusive pages also should not be included in staff portal, such as shopping cart, and making orders. What's more, if you are just a tourist, the features offered to you will be limited in looking through products.

  The purpose to make use of this page protection technique is making sure that all roles in this system behave in their track. Therefore, the system security will be enhanced.

- **Technology implementation**
  There are three roles of users. If the user has registered, there is 1 or 0 in the user table to identify staff and customer. When users log in, the role will be stored in session. If the user does not log in, it is a tourist. To make these forbidden pages clear and sensible for our users, there is a verification of the user's role. When users try to enter a forbidden page as their current character, an appropriate toast message will be raised to tell the users why he or she can not enter the target page.

4.1.3 Changing email or password

- **Design idea**
  The customer can also improve their account security by themselves. Our website provide some approaches for our customers. Generally, there are two ways for them to protect their accounts in their personal page.

  The first one is resetting their email account used to login this website. The resitting of email is almost the same as assigning a new one when a customer try to register an account. The customer are required to input a verification code sending to their old email account to set their new email account.

  Besides of resitting email account, the password for this account can be reassigned as well. To reassign a passport, the customers are required to input their old password first, and then they can give their new password. Changing passwords regularly can help to increase our system security through the decline in the possibility of account stolen.

4.2 Reliability

Generally speaking, there are many consequences affecting the reliability of a software such as the quality of requirements, mistakes embedded in codes, weak design ideas, etc. However, the reliability can also be interpreted as 'whether or not customers feel it is reliable' in the perspective of user. Thus, we mainly focus on three aspects that enhance user's shopping experience.

When users making an order, there will be an intermediate page illustrating the products that they are going to buy as well as the details of each product, like quantities, etc. If all correct, they can confirm to buy or they can return to the previous page to reselect. The other point is that the comments users have inputted to one product will be automatically preserved if they switch to other products, and they can see the them as well when they go back to continue writing the comments of this product. Eventually, the input value of price range might be modified into the wrong form or the reverse order, which causes the system to stop. Thus, we use Regular expression with JQuery to check the form, and if not, the input value will become the initial form.

4.3 Maintainability

4.3.1 Blueprint

The blueprint technology helps to implement flask application module partitioning. There are three modules in this project - main, customer and staff. After partitioning, the project structure is no longer a single script. Each module has its own directory of files, where the code associated with it is written.

4.4 Usability

4.4.1 Resource fully used

1. **Pictures load**

   - **Design idea**
     To speed up our system, the shop page are not going to loads all pictures of products in quick views. Instead, this page listens to each click on quick view and query the pictures that belong to the product. On this page, only basic information of a product is displayed in case resource load use too much bandwidth and time, because if products have many pictures and too many products are in this page, the page rendering is slow on unnecessary details, which causes users unpleasant feelings. Thus, when a customer quick view one of the products, the related pictures are soon queried from the database and load by using ajax.

   - **Technology implementation** Each product shows one of their pictures on the shop page if they have. Quick view of a product triggers .js file to get the product id and to query pictures in the database. Because the number of picture is dynamic, we append new queried pictures to the first picture using for loop to dynamically generate picture DOMs. For a better experience, we add a new class and use if condition to prevent the pictures from being queried again.

2. **Preview video load**

   - **User story**
     As a customer, I want to watch the video preview only if I want in the detail page, so that my bandwidth is not wasted.

   - **Design idea**
     Since the storage of our preview video is quite large and we don't have great compress technology in

uploading video, we take the strategy that if and only if the user clicks the preview button the video is loaded to him/her.

- **Technology implementation**
  If there is a preview video for that product, the video button will appear. Once the button is clicked, .js using the product id get the video path from the database, and then the page jumps a modal that contains the preview video. Besides, if you click exit, the video stops automatically, and once you click again, it will not load again, which saves bandwidth.

4.4.2 Expected user interface and styles

1. **Product display and layout**
   - **Design idea**
     As mentioned above1, most of our front-end files are based on templates, but these are required to fit our back-end logic and transformed to our own styles. Honestly, it brings us great convenience to design our unique website. We are capable of choosing which part is unnecessary, and adding new parts to develop. However, sometimes the decision is hard to make, because you are unsure whether the elements are necessary, and therefore we have a brainstorm and discussion to reach agreement on the choice of different parts. The related information of how we use the template is written above1.
2. **Provides user enough information**
   - **Design idea**
     We provides customers with reviews and posters of products. It makes customers convince that the information of product is detailed and willing to purchase products.
3. **Statistic analysis for staff**
   - **Design idea**
     There are diagrams in templates. Inspired by this and considering that the staff works for employers, we think that employers are willing to know the profits and the number of orders immediately. To achieve data visualization, we connect our database to the chart presentation so that every time the staff enters the index page he/she knows the overall of online-shopping by viewing the diagram rather than calculating by themselves.

5 Database Schema

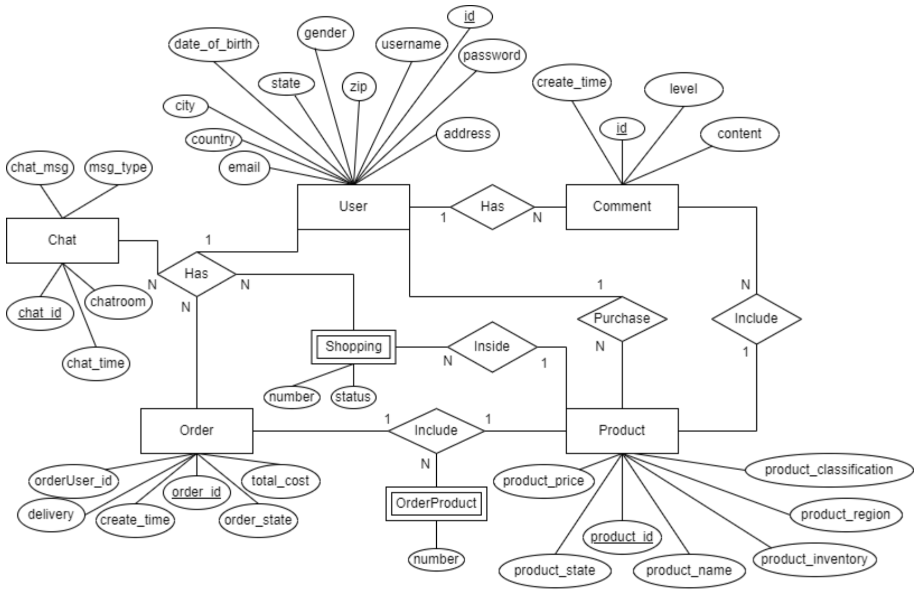

Figure 6: ER-diagram

There are 7 tables made up of the most important part in our database architecture showed in figure 4. The Shopping table and OrderProduct table can be considered derived because they are the intermediate tables when implementing many-to-many relationship. Moreover, to uniquely identify each row in these two tables, we use the foreign keys of its related table as the composed primary key and we also follow the third normal form to to eliminate the inner dependency in each table.

When we construct the database in coding level, we also build other tables like 'Image', 'Answer' to preserve the image paths of products and record the auto-responses to different questions, etc.

6 Conclusion

6.1 Strength

Overall, all group members cooperated to complete the given task.

- The website has met all the functions required by customers, and added comments, automatic reply and other functions to improve the user experience.
- All members shared the same goal, accomplished tasks and solved problems and conflicts actively.

6.2 Limitation and future work

Due to time constraints, what we can accomplish is limited. There are some suggestions for improvement.

- More useful functions can be added, such as:
  - communication between staff
  - financial management
- Non-functional requirements should be considered more, such as:
  - block some malicious attacks
  - a journal system to facilitate the diagnosis of the cause of defects
  - backup files to restore data in the event of a system failure
- More comprehensive testing with various test methods is needed.

6.3 Acknowledgement

Thanks to all the team members for their efforts in the past three months. Thanks to teachers and teaching assistants for the learning opportunities and assistance. We learned more technical knowledge and cooperation skills in the process of solving problems.

References

O Mark. Bootstrap file input. URL `https://github.com/twbs/bootstrap`.

J.Ko Amy. Cooperative software development, 2022. URL `https://faculty.washington.edu/ajko/books/cooperative-software-developme`.

E.T. Indeed. Four common types of team conflict and how to resolve them, 2021a. URL `https://www.indeed.com/career-advice/career-development/types-of-team-conflict`.

E.T. Indeed. Conflict resolution skills: Definition and examples, 2021b. URL `https://www.indeed.com/career-advice/resumes-cover-letters/conflict-resolution-skills`.

Krajee. Bootstrap file input. URL `https://github.com/kartik-v/bootstrap-fileinput`.