Online Game Store

An online game store supports costumer and administrator to interact with online purchase video games.

Date: 2022/Nov-2022/Dec

EECS-4413 Team project:

Mohanmond Alaili

Jingya Su

Zijian Huang

Zhengkun Lou

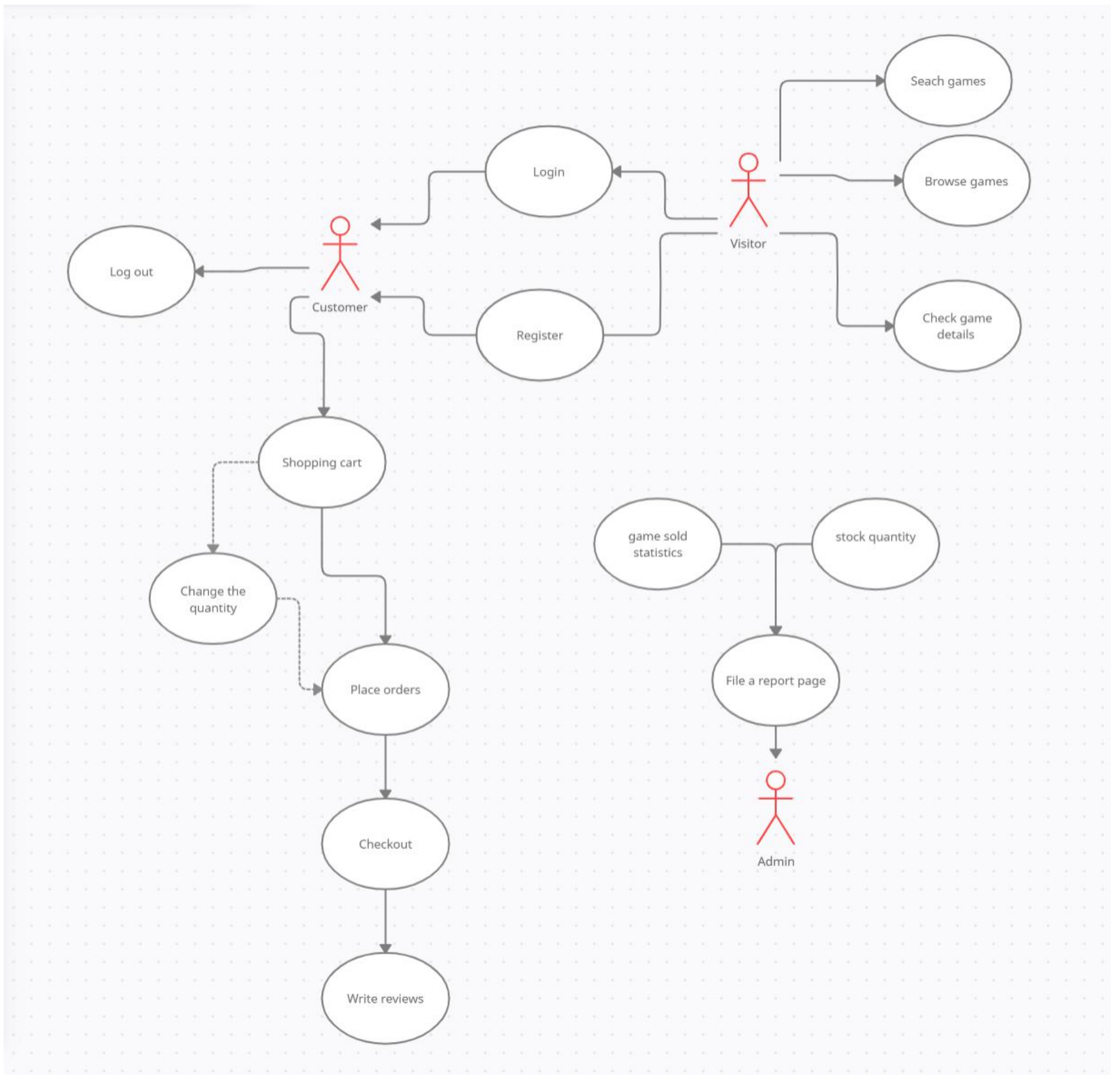
Our Project Schema And Design

Introduction:

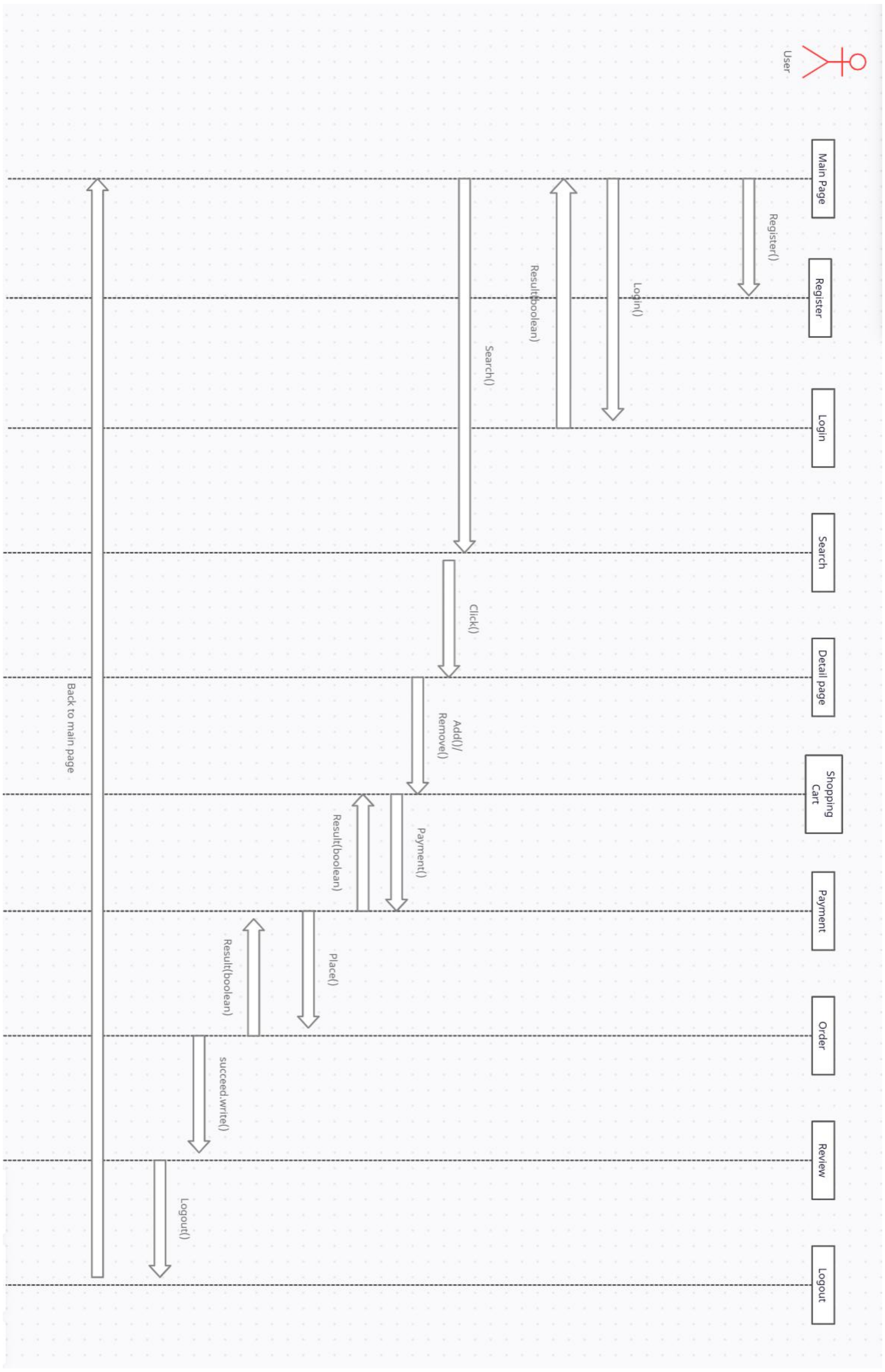
Our team's aim is design an Online Game Shopping Store that allow people to shop video games, read/write reviews, create shopping carts and so on. We also allow administers to analyze, operating the website.

User-case:

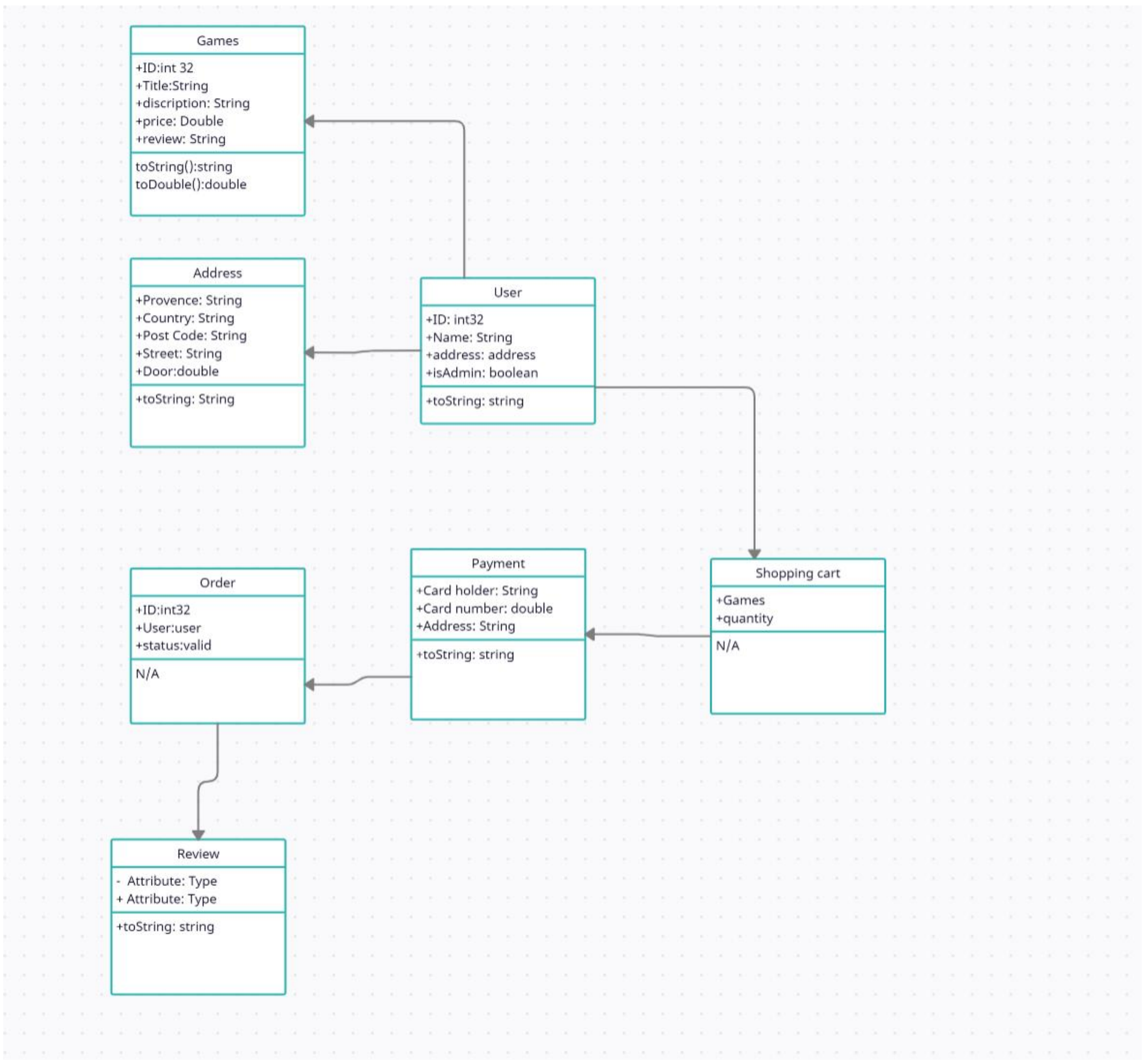
subsystem



Class Diagram:



Sequence Diagram:



Design:

Overall our project is developed by the JAVA framework with MVC pattern design. We use React to develop our front-end and Intellij for the back-end and API development is being developed through Java and Servlets Our war file could run on any server and cloud by requirements. The sequence diagram can simply explain our project’s architecture.

MVC:

The model-view-controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects. This is done to separate internal representations of information from the ways information is presented to and accepted by the user.

By applying MVC, we separate into three parts for four people, as front-end(View) back-end(Model), and API(Controller). Hence we spent more time than expected while connecting MVC together.

Build Pattern:

Front-end(By Alaili):

We chose to implement the front-end using React, since React simplifies creating complex user system by constructing components one piece at a time where those pieces come together to create a large complex component or can be reused elsewhere facilitating consistent design. However, even though React simplifies complexity, it doesn't completely eliminate it. Naturally, as components grew more complex, maintaining them becomes much more tedious especially since some components are complex enough to warrant being in completely separate files. Unfortunately, there's no perfect solution to this.

Moreover, state management was handled by Redux, and we used the Redux ToolKit (library similar to Axios) to send and receive HTTP requests to and from the API. Redux ToolKit groups all the request handlers in one file making maintenance easy in case there are modifications in the API. We also configured it to cache certain repetitive requests with output that does not change often like getting all the products or the cart. Afterwards, certain actions that would actually modify their output like modifying the cart trigger a hard refresh invalidating the cache.

But since we can't use his part, we made ourselves a simple front end

Front-end(We actually use for our project):

By using python and html, we made a simple text UI. With Python we can make a complete web page with a beautiful interface and rich interactive features. However, due to time constraints, we were not able to improve and beautify the interface in one day, our web pages have only the most basic interactive features and the most rudimentary page displays. The framework we use is flask. flask is a web development framework that is easy to learn, so it is also very easy to build web services with flask.

Back-end:

By using Intellij, we do have a better experience than eclipse, we would say this is a powerful, modern, and fancy IDE. In our project, we separated the back-end into two different parts: "Item/Shopping cart" and "User/Admin". For Showing in our files:

In this part, since our backend is done by two team members, we chose to divide the backend code into, controller, exception, interceptor, model, repository and service in order to facilitate collaboration. Each part is responsible for the functionality as their name tells, and the names of the different functions can be seen intuitively, so that we can easily add and change functions. However, this division of labor actually leads to duplicate functions or parts of duplicate code in our backend sometimes, but we fix this problem when do the final testing.

Database:

We decide to use MySQL workbench as our primary database system. 1. Store data from the database. 2. Store data from the client (form submission), and store data as an intermediary.

Data Access Object: data access object, the main thing to do is to add, delete, and modify a single table in the database, and the query may be multi-table management query.

Web API:

I mainly used the Java Servlet that learned from class, while using it, I've made the different get/post/put/delete interface to adapt the front end design, back end setting up. "Get" is mainly used for querying small volume data that requires fast return speed. However, the interface is exposed outside, so there will be certain risks. "Post" is for register, upload data.

Synchronous interaction and asynchronous interaction are also being used in API functions. In the login interface, when performing a login operation, the user name, password, token and other fields are encrypted and then verified by the interface. Only after the verification result is returned can the login be successful. Other functions are mainly use asynchronous interaction to make the website fast.

Non functional requirements:

Tester:

As a development team, we are running a tight time while making our project hence we might have some bugs not being find out/ fix.

Our tests are composed of many different parts, and the test-case test is only the final test of the program. For example, we use another database with auto-generated data to test our back-end and API during the development process; this database does not contain a lot of data, but it is the fastest way to check our current problems and make corresponding improvements. We also performed manual tests after integration to test the responsiveness and security of our web pages for different situations.

And we have tested our project according to MVC model without issue while running those user cases:

1. We tested the item-related functions for example sorting by type and brand.
2. We tested the user-related functions, including login, shopping cart and payment, etc.
3. We have tested the admin function and performed server related tests

Current issue found:

- Since the absent of front end. We could only make a simple text UI and not likely it can be finished before the deadline. So the function of admin maybe absent.
- And some operations execute more SQL queries and need to be optimized.
- The shopping cart is not automatically emptied after an order is placed

Run time:

Security:

The security test was being developed by HTTP Basic Auth setup, A way to provide a username and password when making a request. In basic HTTP authentication, the request contains a header field of the form Authorization: Basic, where the credentials are a Base64 encoding of the ID and password connected by a single colon. We have tested the function in different situations personally.

HTTP Basic Auth can access the API with the username and password of the visit, as the information will be exposed, we will see an extra string of characters encrypted with Base64 in the request header. This is the access credentials of the user.This is a basic way to make our website's security,even the level of security is low.

Collaboration:

Process:

As a development team, we use a variety of collaboration tools to complete our projects. We use discord/WeChat/in person to communicate, and we store our documentation and resources in google drive. Version control of our application is done through git and is hosted on GitHub. Features and issues are also managed through GitHub.

Overall, every team members were engaged passionately about contributing to the project, and there were no conflicts while we discuss technical issues.

Contribution:

Mohanmond Alaili:

Responsible for all front end design and implementation. This project reinforced the importance of good communication for me, since designing and implementing a frontend with no knowledge of what data it will show leads to wasting a lot of time. Thus, I had to sufficiently communicate with my teammates to properly understand what data I'm receiving and what the shape of it is, to actually create a functional and pretty front end that accurately presents this information. Since, requirements changed often during development, being flexible and building the frontend in an easily maintainable way to easily accomadate those new changes was also highly important.

Actually, **N/A**. Since front-end part not be successfully built.

Jingya Su:

Responsible for part of the back-end coding that with all user related and admin related, and the database part with all user and admin function. More specifically, Write reviews on items, register, sign in, sign out, run reports on sells, run reports on application usage. This website project systematically used all the content learned in the course and left a lot of room for exploration.

Zijian Huang:

Responsible for part of the back-end coding and a quick rewrite about front-end part. Including doing all the requirement in the document about items or products and the part about shopping cart. I also do the part of deploying our application in cloud, let database run in cloud and provide the corresponding curl testcase. I really learned a lot from this teamwork. I learned how important communication is in group work, timely communication with group members, and timely feedback on problems can directly affect the smoothness of cooperation.

Then I wrote a simple text UI out on the 16th, the basic functions are there, except for the user review function in the front-end did not achieve, because there is really not enough time.

Zhengkun Lou:

Responsible for all API design&coding including set up and test with security design coding , the whole "Controller" part in MVC pattern, write report document. I learned a lot from team working, while working as a team means you have to take your responsibility and make your work as good as you can, hence your work should benefit whole team, and make your code easy to read, your progress need catch up with team, my teammates are knowledgeable, I've learned other ways to thinking and writing code, I really enjoy with my team, Zijian is a passionate guy, he helped me to stove a lot of headache issues, he also spent a lot of time to communicate with all team members to make sure our team progress.

Attestations:

Member	Signature
Mohanmond Alaili	
Jingya Su	Jingya Su
Zijian Huang	[illegible]
Zhengkun Lou	[illegible]

Conclusion:

Overall we had a great experience working with each other, about our project: the MVC based online game store is working well even with some tiny issues. We learned a lot from working on this project and found out how to further optimize this product in the next update. In our project, first we use DigitalOcean which is a cloud hosting provider, this one is easy to use, thus good for beginner. But overall, it may not be the right choice for a real shopping Sites . Also we using Name.com while building our own webpage and having our own domain. What's more, using react is a better option for having a dynamic, aesthetic and flexible UI design, but not really friendly to group work, especially no other team member has even heard of react except for the team members who write the front end. But since none of us has experience with node.js beside our front end worker, unfortunately, we(API, back-end workers) are unable to build the front end as we planned by applying react and so on. Then we decide to make a simple front end to test and build our project to make sure our loss is as less as we can.. We are very optimistic about the future of this site, one of the reasons is we have a comprehensive back-end and database system to provide the necessary support for this website. But our website also has shortcomings, partly due to the lack of time. Our site does not automatically empty the shopping cart after placing an order, and some operations require the execution of many SQL queries, all of which need to be optimized step by step. By making our own EB Game, we observed and studied GameStop's website, including the various interactions and applications of the various features of this website. We also tried to replicate as many of GameStop's features as we could in the limited time we had.

Special Notes:

We reported Alaili for the following reasons: Firstly, Talking to him is just not an easy task, he often does not check our discord chat channel, didn't keep updated about his progress, and we need to proactively ask him about his progress every time, we even need to ask him about his progress via private chat. And often the cycle of communicating with him about one question is a day, which is extremely inefficient. The second point is that he said at the first group meeting that he could finish the front end within a week, so we gave him the front end with confidence. On the day of the presentation, We asked him about his progress, and he said he could finish writing all the user pages this weekend, but on the 11th when we finished testing the backend and were ready to connect the front end for final testing, he said his program was still short of all the user pages including the admin page. He refused to continue writing his part on the day of the 11th because he says he has to review for the final thus he has no time to complete his part and he never updated his program after that. By this time I was ready to help him finish the part he hadn't written, but the third and most serious problem arose: he was writing front-end code in a language that none of us had ever touched, he used React, and because of he rarely takes the initiative to communicate with us, we don't have chance to know that he is using React until the day we asked him to upload his work. We didn't know how to take over his program, we never learned to use this. And one of our team members asked if he finished his part 15th afternoon on the discord channel but he didn't reply until 7pm, we really tried our best to try to talk to him. And sadly if he decides not to finish his part, then we had no choice but to make a text UI first, then try to supplement the CSS in the next days, and do our best to write a front-end out in these in one day. And everyone was busy at the end of the semester, but we also finished our individual parts within the limited time we had, and there was no such thing as refusing to finish our parts because we had to review for the final. So I think his actions were extremely unaccountable to the other members of the group. And this irresponsible behavior of his may seriously affect our final results, because our shopping site cannot be fully completed, even in the case of the deadline extension.

For the contribution part, we(Jingya, Zijian, Zhengkun) have different opinion with alaili's note, we tried our best to resopse alaili as soon as we can even at mignight, for the design of front end we never ask alaili to switch to our idea (using traditional HTML files) we give enough time and space for him to build it, however until the deadline he did not make the front end connect to whole project and still asking for API and database change. He rearly tell us anything proactive even once. And now in his contribution part, he accuse us for not talking to him.

Reference:

https://en.wikipedia.org/wiki/SQL

https://en.wikiversity.org/wiki/E-Commerce

https://en.wikipedia.org/wiki/API

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

https://en.wikipedia.org/wiki/Redux_(JavaScript_library)