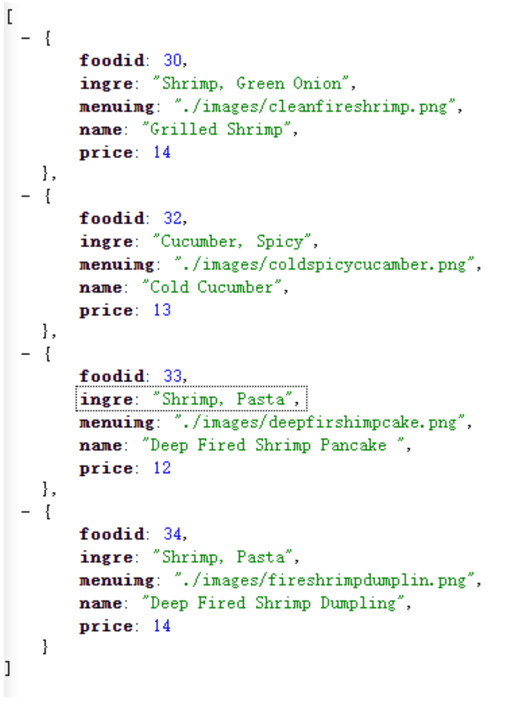
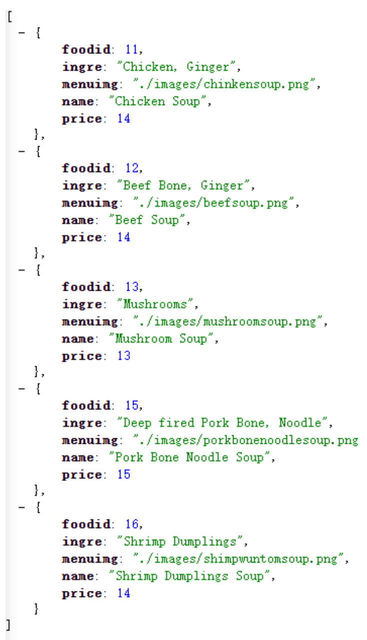
Test 1.

1. Purpose: Test loading data in the database
2. Pre-Condition: There exists data in reservation, food, and drink table.
3. Post-Condition: API server will transfer them to local, just check the localstorage.

check API first: http://awsl370.hestech.cn/API/food & http://awsl370.hestech.cn/API/drink



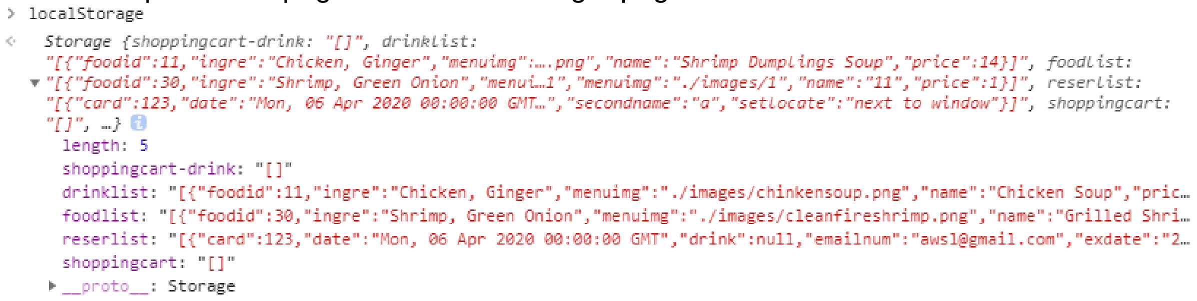
```
[
 - {
       foodid: 30,
       ingre: "Shrimp, Green Onion",
       menuimg: "./images/cleanfireshrimp.png",
       name: "Grilled Shrimp",
       price: 14
   },
 - {
       foodid: 32,
       ingre: "Cucumber, Spicy",
       menuimg: "./images/coldspicycucamber.png",
       name: "Cold Cucumber",
       price: 13
   },
 - {
       foodid: 33,
       ingre: "Shrimp, Pasta",
       menuimg: "./images/deepfirshimpcake.png",
       name: "Deep Fired Shrimp Pancake ",
       price: 12
   },
 - {
       foodid: 34,
       ingre: "Shrimp, Pasta",
       menuimg: "./images/fireshrimpdumplin.png",
       name: "Deep Fired Shrimp Dumpling",
       price: 14
   }
]
```



```
[
 - {
       foodid: 11,
       ingre: "Chicken, Ginger",
       menuimg: "./images/chinkensoup.png",
       name: "Chicken Soup",
       price: 14
   },
 - {
       foodid: 12,
       ingre: "Beef Bone, Ginger",
       menuimg: "./images/beefsoup.png",
       name: "Beef Soup",
       price: 14
   },
 - {
       foodid: 13,
       ingre: "Mushrooms",
       menuimg: "./images/mushroomsoup.png",
       name: "Mushroom Soup",
       price: 13
   },
 - {
       foodid: 15,
       ingre: "Deep fired Pork Bone, Noodle",
       menuimg: "./images/porkbonenoodlesoup.png
       name: "Pork Bone Noodle Soup",
       price: 15
   },
 - {
       foodid: 16,
       ingre: "Shrimp Dumplings",
       menuimg: "./images/shimpwuntomsoup.png",
       name: "Shrimp Dumplings Soup",
       price: 14
   }
]
```

For loadFood(), loadDrink()

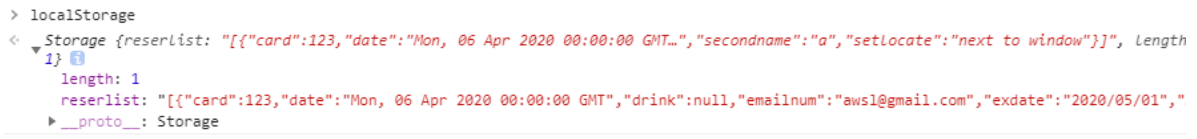
when we open menu page and menu-manager page:



```
> localStorage
<  Storage {shoppingcart-drink: "[]", drinklist:
   "[{"foodid":11,"ingre":"Chicken, Ginger","menuimg":….png","name":"Shrimp Dumplings Soup","price":14}]", foodlist:
 ▼ "[{"foodid":30,"ingre":"Shrimp, Green Onion","menui…1","menuimg":"./images/1","name":"11","price":1}]", reserlist:
   "[{"card":123,"date":"Mon, 06 Apr 2020 00:00:00 GMT…","secondname":"a","setlocate":"next to window"}]", shoppingcart:
   "[]", …}
    length: 5
    shoppingcart-drink: "[]"
    drinklist: "[{"foodid":11,"ingre":"Chicken, Ginger","menuimg":"./images/chinkensoup.png","name":"Chicken Soup","pric…
    foodlist: "[{"foodid":30,"ingre":"Shrimp, Green Onion","menuimg":"./images/cleanfireshrimp.png","name":"Grilled Shri…
    reserlist: "[{"card":123,"date":"Mon, 06 Apr 2020 00:00:00 GMT","drink":null,"emailnum":"awsl@gmail.com","exdate":"2…
    shoppingcart: "[]"
  ▶ __proto__: Storage
```

For loadReservation()

when we open reservation-manager page or menu page



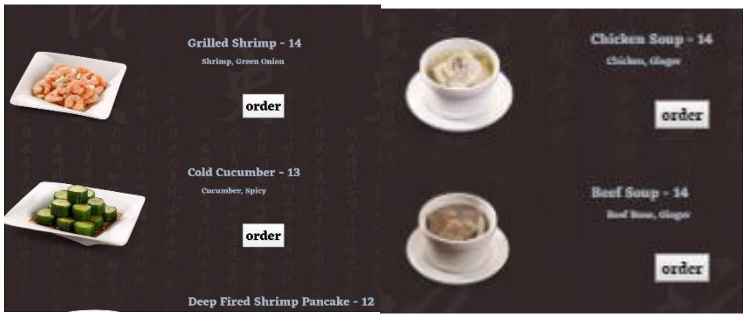
```
> localStorage
< ▼Storage {reserlist: "[{"card":123,"date":"Mon, 06 Apr 2020 00:00:00 GMT…","secondname":"a","setlocate":"next to window"}]", length
   1}
    length: 1
    reserlist: "[{"card":123,"date":"Mon, 06 Apr 2020 00:00:00 GMT","drink":null,"emailnum":"awsl@gmail.com","exdate":"2020/05/01","
  ▶ __proto__: Storage
```

we get the data from the database successfully!

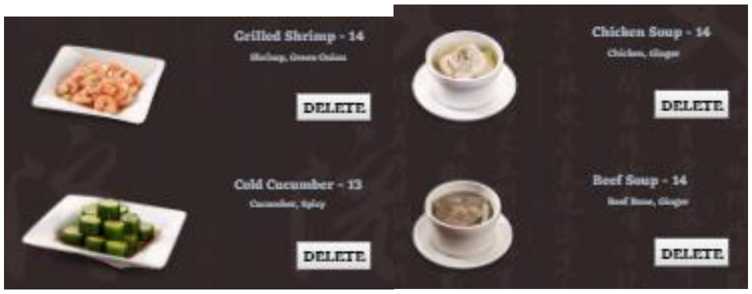
Test 2.
1. Purpose: Test the food menu and drink menu generated into html
2. Pre-Condition: There exists date in food menu and drink menu list
3. Post-Condition: food menu and drink menu items being generated in html and display correctly

generatemenu() and generatedrink()

As we open up the menu page, we can observe the result:



generatemenu_manager(), generatedrink_manager()

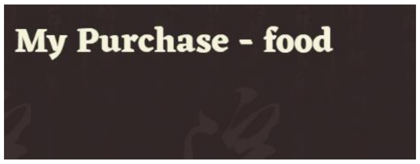


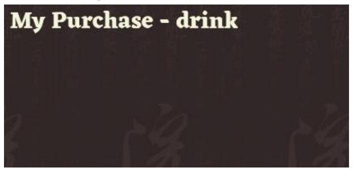
The menu item generated correctly.

Test 3.

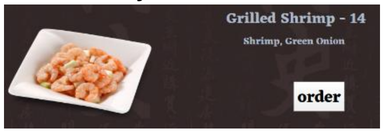
1. Purpose: Test the order button in food menu and drink menu page working or not
2. Pre-Condition: The “My purchase” box is empty
3. Post-Condition: The selected menu item should display in “My purchase ” box.

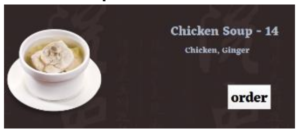
First we check is there any exist item in purchase box:



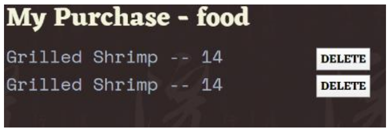


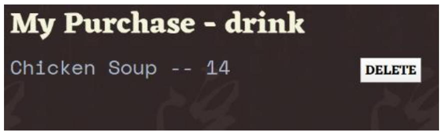
Next we try to order two Grilled Shrimp and one chicken soup by clicking each button.

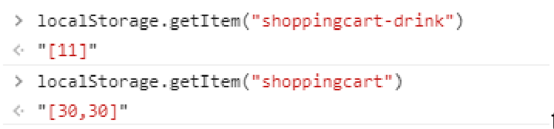




Observe the purchase box and we found the order is correct.







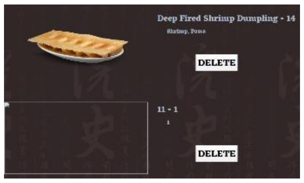
```
> localStorage.getItem("shoppingcart-drink")
<· "[11]"
> localStorage.getItem("shoppingcart")
<· "[30,30]"
```

their id were stored in local successfully

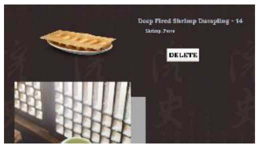
Test 4.

1. Purpose: Test the deleted button in the food menu and drink menu page working or not for manager.
2. Pre-Condition: There exist items in the food menu and drink menu.
3. Post-Condition: The item selected is remove from the menu

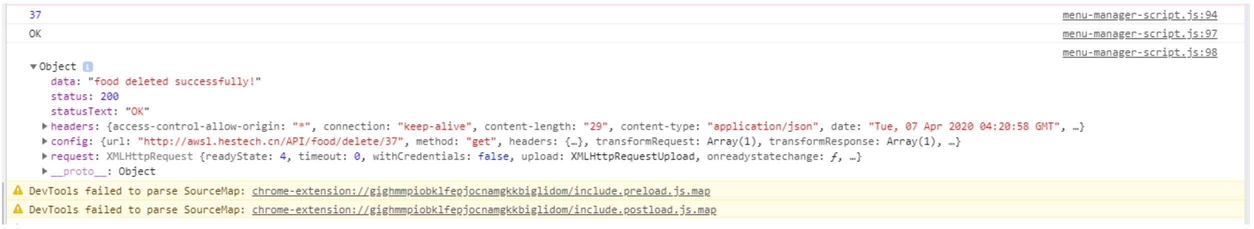
First we observe and target the menu item name “11-1” as our target



Next we click the DELETE button beside “11-1” and observe the outcome.



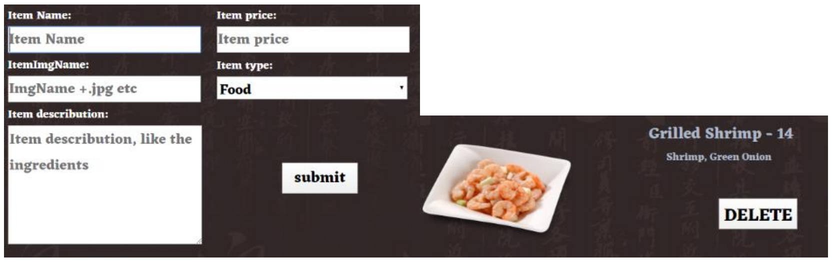
We can see the item “11-1” is being removed and the outcome matches our expective.
We can also observe the console log to see the website’s response.



```
37                                                                                      menu-manager-script.js:94
OK                                                                                      menu-manager-script.js:97
                                                                                        menu-manager-script.js:98
▼Object
    data: "food deleted successfully!"
    status: 200
    statusText: "OK"
  ▶headers: {access-control-allow-origin: "*", connection: "keep-alive", content-length: "29", content-type: "application/json", date: "Tue, 07 Apr 2020 04:20:58 GMT", …}
  ▶config: {url: "http://awsl.hestech.cn/API/food/delete/37", method: "get", headers: {…}, transformRequest: Array(1), transformResponse: Array(1), …}
  ▶request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, onreadystatechange: ƒ, …}
  ▶__proto__: Object
⚠ DevTools failed to parse SourceMap: chrome-extension://gighmmpiobklfepjocnamgkkbiglidom/include.preload.js.map
⚠ DevTools failed to parse SourceMap: chrome-extension://gighmmpiobklfepjocnamgkkbiglidom/include.postload.js.map
```

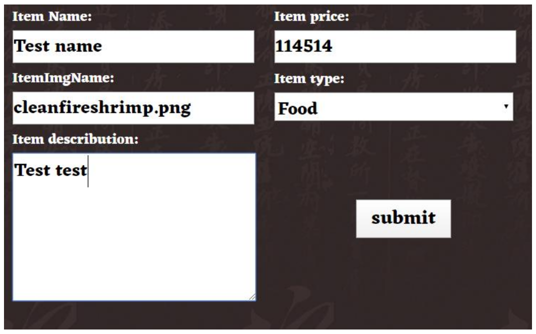
As the console states, the item id 37 is deleted successfully.

Test 5. Adding a normal item to the menu list.
1. Purpose: To add an item into the menu list.
2. Pre-Condition: Picture of the food and valid description about the menu item.
3. Post-Condition: A new item is being added.
First to add a new item we need a item name, item price, where we want to added (food menu or drink menu) , description about the item and a valid picture path.



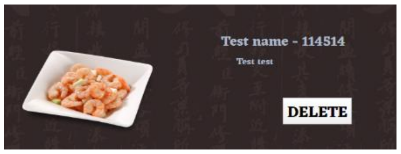
Item Name:	Item price:
Item Name	Item price
ItemImgName:	Item type:
ImgName +.jpg etc	Food
Item describution:	
Item describution, like the ingredients	

We can use existing material to accomplish this test.
We used the same picture as Grilled Shrimp but different text and price.

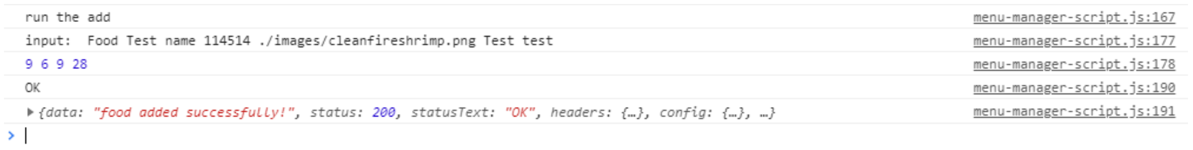


Item Name:	Item price:
Test name	114514
ItemImgName:	Item type:
cleanfireshrimp.png	Food
Item describution:	
Test test	

As expected a new item with the same image but different text is being added to the food menu.



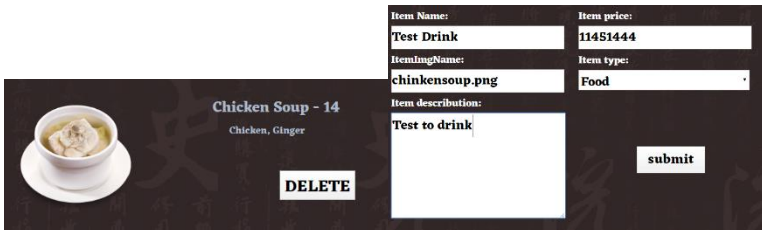
We can also observe the console log to see the website’s response.



```
run the add                                                               menu-manager-script.js:167
input:  Food Test name 114514 ./images/cleanfireshrimp.png Test test      menu-manager-script.js:177
9 6 9 28                                                                  menu-manager-script.js:178
OK                                                                        menu-manager-script.js:190
▶{data: "food added successfully!", status: 200, statusText: "OK", headers: {…}, config: {…}, …}   menu-manager-script.js:191
> |
```

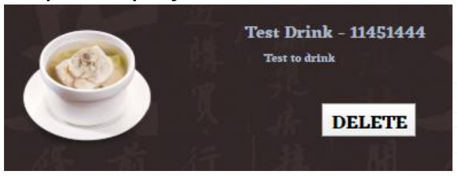
As the console states, the item is added successfully.

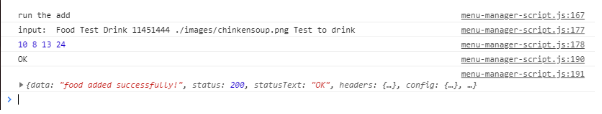
We can also try to add a new item to the drink menu with similar steps.
This time we use the same image as chicken soup but difference text and price.



Item Name: Test Drink
Item price: 11451444
ItemImgName: chinkensoup.png
Item type: Food
Item description: Test to drink

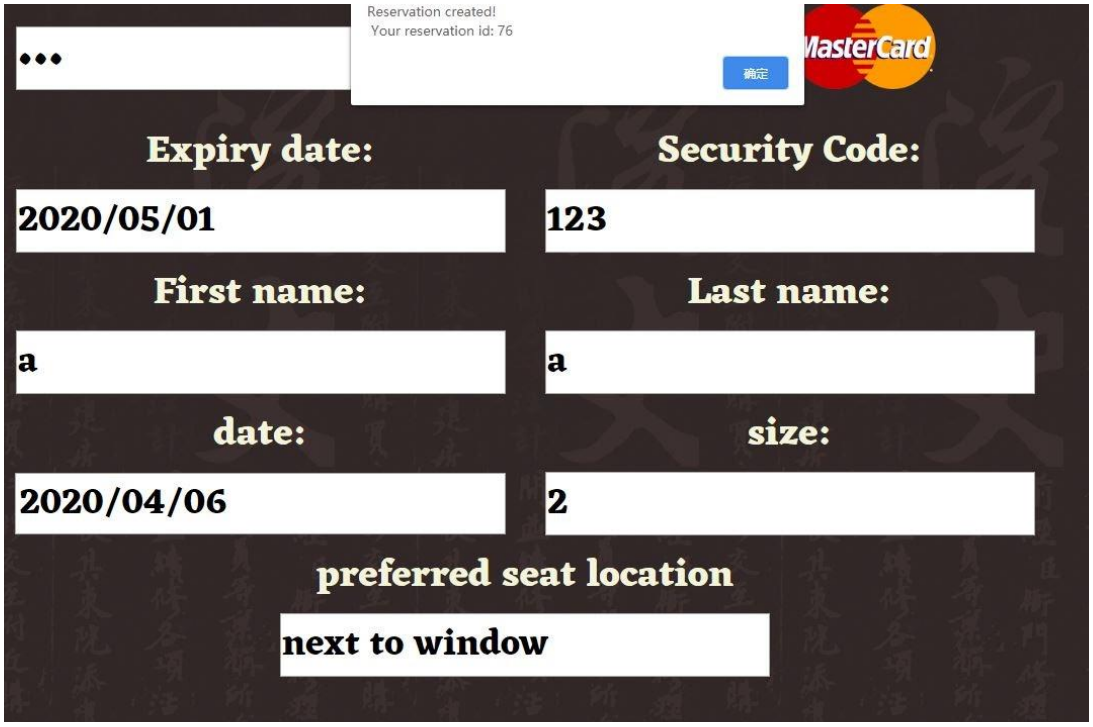
After clicking the submit button we can see a new item is add to drink menu and the console output display without error.





```
run the add                                                                 menu-manager-script.js:167
input:  Food Test Drink 11451444 ./images/chinkensoup.png Test to drink     menu-manager-script.js:177
10 8 13 24                                                                  menu-manager-script.js:178
OK                                                                          menu-manager-script.js:190
                                                                            menu-manager-script.js:191
▶{data: "food added successfully!", status: 200, statusText: "OK", headers: {…}, config: {…}, …}
> |
```

Test 6.   Making a normal reservation (customer)
1. Purpose: testing for initial local data and making a reservation.
2. Pre-Condition: none.
3. Post-Condition: the reservation information(reservation id) about reservations created should be displayed to the user.
4. Action: Input information about reservation and click submit.
Output:


Reservation created!
Your reservation id: 76

•••

Expiry date: 2020/05/01

Security Code: 123

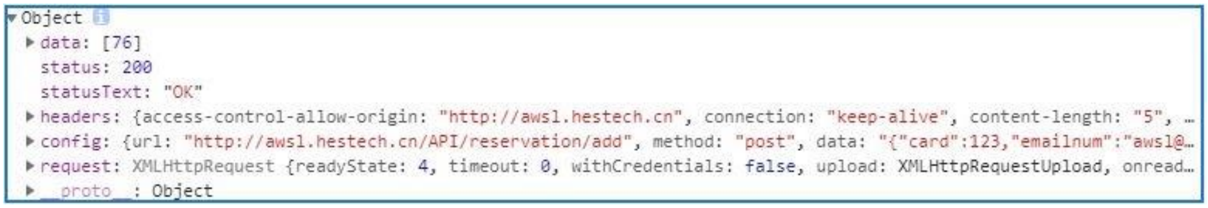
First name: a

Last name: a

date: 2020/04/06

size: 2

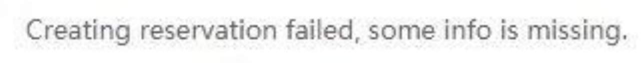
preferred seat location: next to window

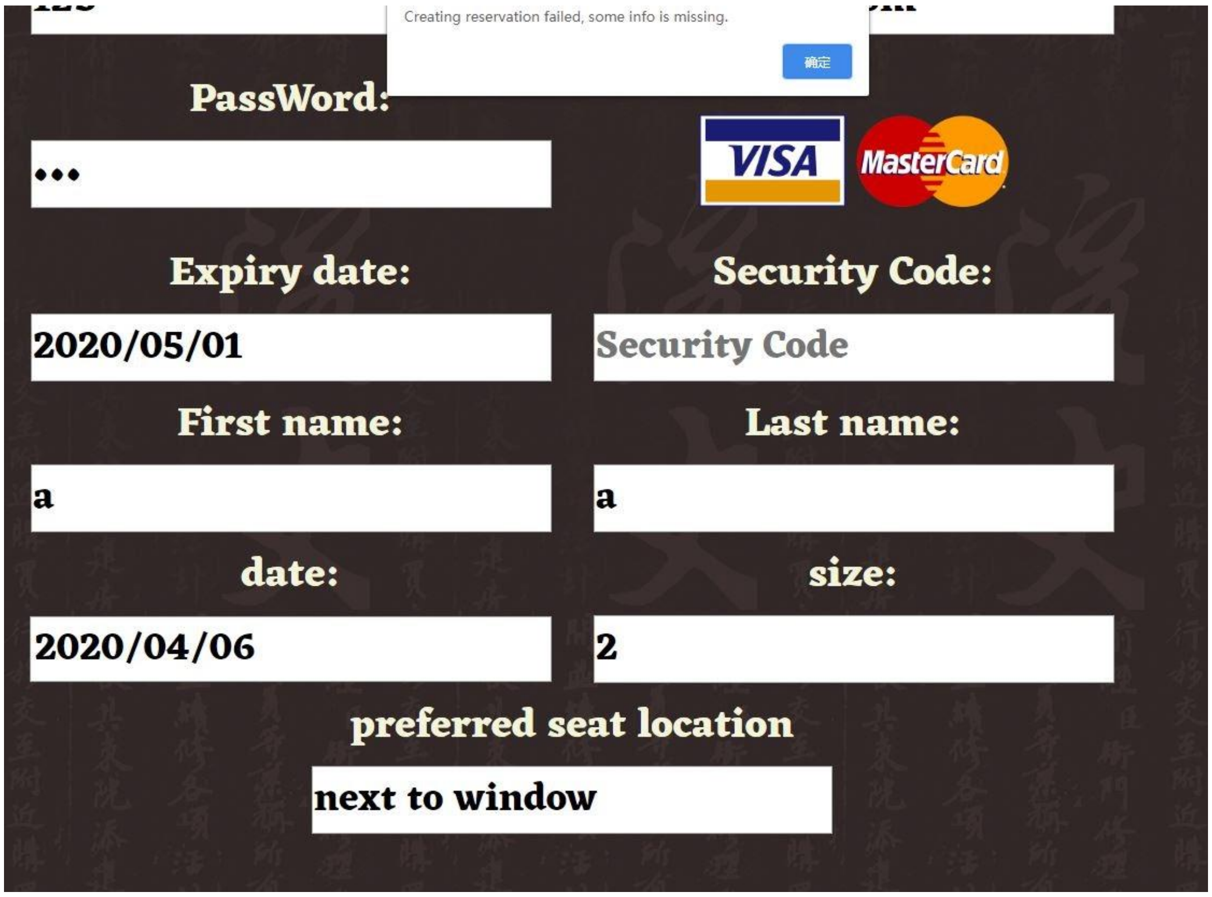

```
▼Object
  ▶data: [76]
   status: 200
   statusText: "OK"
  ▶headers: {access-control-allow-origin: "http://awsl.hestech.cn", connection: "keep-alive", content-length: "5", …
  ▶config: {url: "http://awsl.hestech.cn/API/reservation/add", method: "post", data: "{"card":123,"emailnum":"awsl@…
  ▶request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, onread…
  ▶__proto__: Object
```

Test 7. Making a reservation with at least one empty input box (customer)

1. Purpose: testing for making a reservation with incomplete information. The reservation should not be created.
2. Pre-Condition: none
3. Post-Condition: a pop box with alerts shown to user
4. Action: Input information about reservation and leave at least one blank field. Then click submit.

Output:


Creating reservation failed, some info is missing.


Creating reservation failed, some info is missing.

PassWord:

•••

Expiry date:

2020/05/01

Security Code:

Security Code

First name:

a

Last name:

a

date:

2020/04/06

size:

2

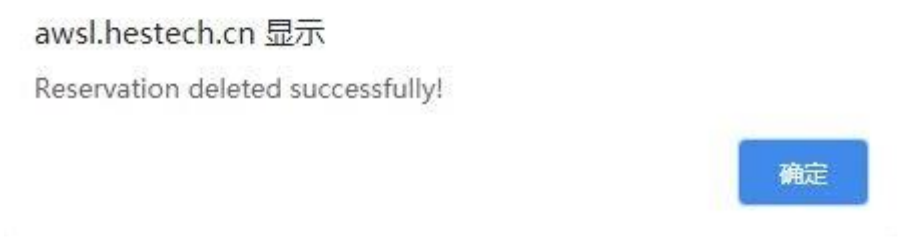
preferred seat location

next to window

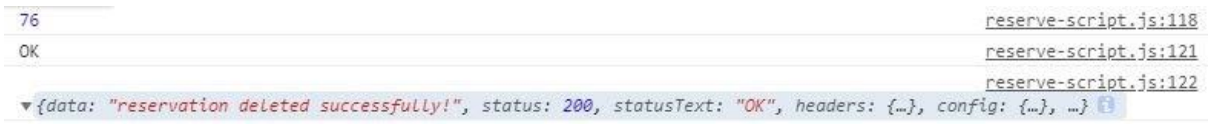
Test 8. Deleting the reservation (Manager)

1. Purpose: testing for deleting a reservation
2. Pre-Condition: there exists a reservation
3. Post-Condition: a pop box shown to user and the reservation does not exist
4. Action: click the “delete reservation” button.

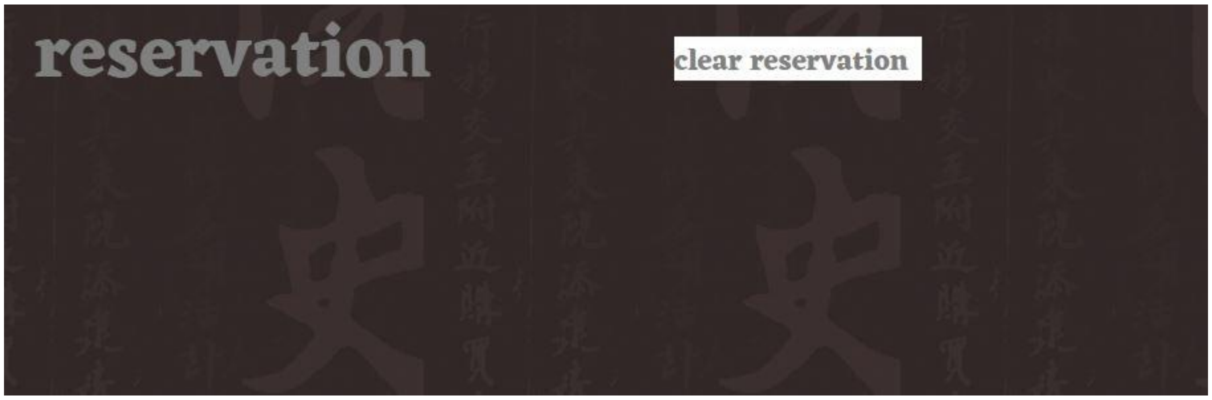
The Output:


awsl.hestech.cn 显示

Reservation deleted successfully!

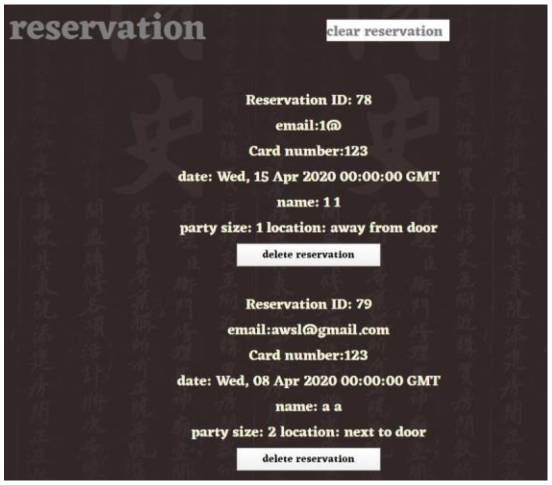


```
76                                                                reserve-script.js:118
OK                                                                reserve-script.js:121
                                                                  reserve-script.js:122
▼{data: "reservation deleted successfully!", status: 200, statusText: "OK", headers: {…}, config: {…}, …}
```



Test 9. Clear all reservations (Manager)
1. Purpose: testing for clearing reservations
2. Pre-Condition: there exists at least one reservation
3. Post-Condition: a pop box shown to the user and there is no reservation.
4. Action: click the “clear reservation” button.

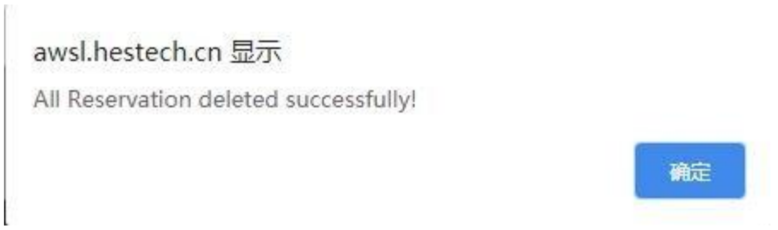
The Output:
Before:



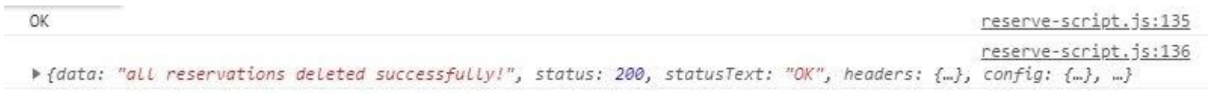
Reservation ID: 78
email:1@
Card number:123
date: Wed, 15 Apr 2020 00:00:00 GMT
name: 1 1
party size: 1 location: away from door

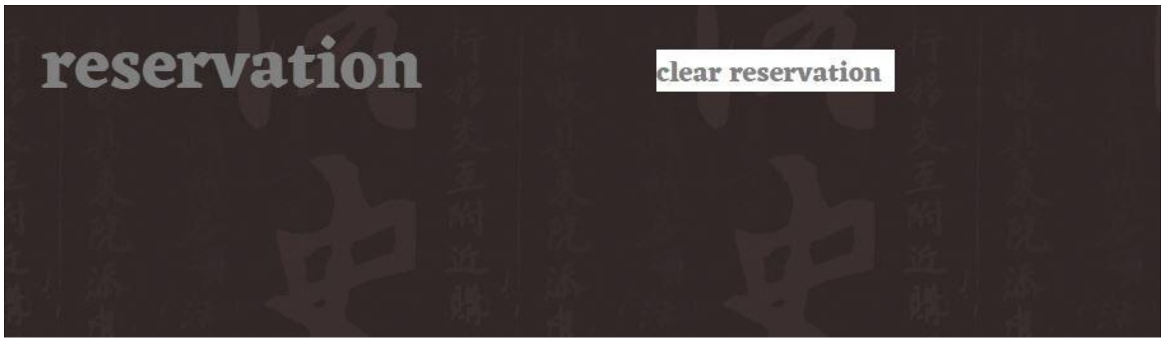
Reservation ID: 79
email:awsl@gmail.com
Card number:123
date: Wed, 08 Apr 2020 00:00:00 GMT
name: a a
party size: 2 location: next to door

After:


awsl.hestech.cn 显示

All Reservation deleted successfully!

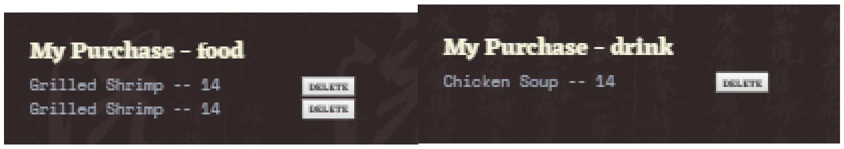

```
OK                                                                    reserve-script.js:135
                                                                      reserve-script.js:136
▸{data: "all reservations deleted successfully!", status: 200, statusText: "OK", headers: {…}, config: {…}, …}
```



Test 10.

1. Purpose: using reservation id to order food
2. Pre-Condition: There exist items in the food menu and drink menu, also valid reservation
3. Post-Condition: reservation table will be updated

first, order food and drink:


My Purchase – food

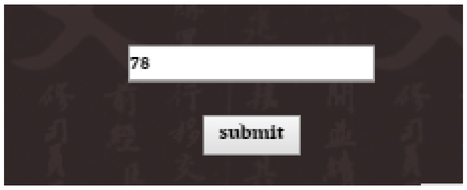
Grilled Shrimp -- 14

Grilled Shrimp -- 14

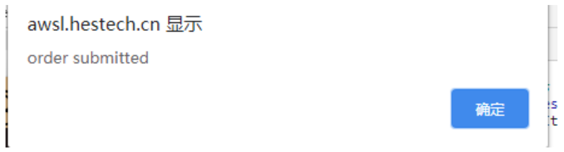
My Purchase – drink

Chicken Soup -- 14

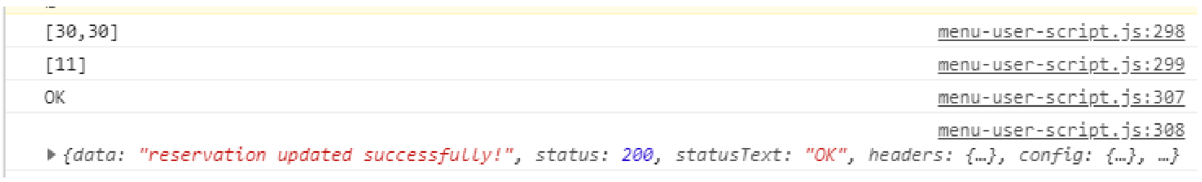
second, send it reservation id: 78



we will get a notification, and console will show us some feedback

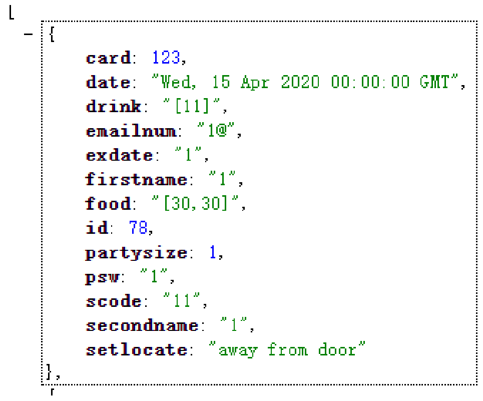


order submitted



```
[30,30]                                                                  menu-user-script.js:298
[11]                                                                     menu-user-script.js:299
OK                                                                       menu-user-script.js:307
                                                                         menu-user-script.js:308
▸{data: "reservation updated successfully!", status: 200, statusText: "OK", headers: {…}, config: {…}, …}
```

let’s check it using API server: http://awsl370.hestech.cn/API/reservation

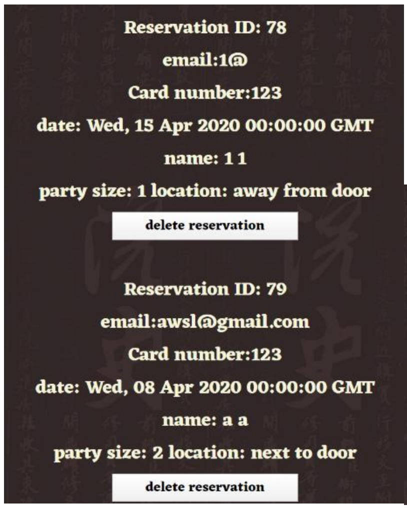


```
[
- {
      card: 123,
      date: "Wed, 15 Apr 2020 00:00:00 GMT",
      drink: "[11]",
      emailnum: "1@",
      exdate: "1",
      firstname: "1",
      food: "[30,30]",
      id: 78,
      partysize: 1,
      psw: "1",
      scode: "11",
      secondname: "1",
      setlocate: "away from door"
  },
```

ordered food and drink has been stored in the database

Test 11.
1. Purpose: using reservation id check their order
2. Pre-Condition: There exist items in the food menu and drink menu, valid reservation list
3. Post-Condition: reservation info and user’s order will be shown on the page

There exist reservations, ID 78 and 79. lets try this two reservation in the checkorder.html page:


Reservation ID: 78

email:1@

Card number:123

date: Wed, 15 Apr 2020 00:00:00 GMT

name: 1 1

party size: 1 location: away from door

Reservation ID: 79

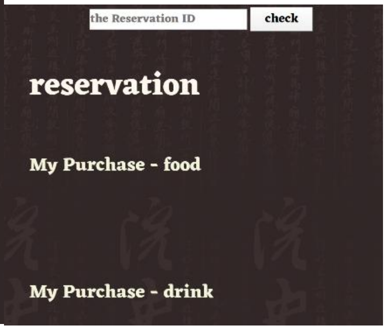
email:awsl@gmail.com

Card number:123

date: Wed, 08 Apr 2020 00:00:00 GMT

name: a a

party size: 2 location: next to door

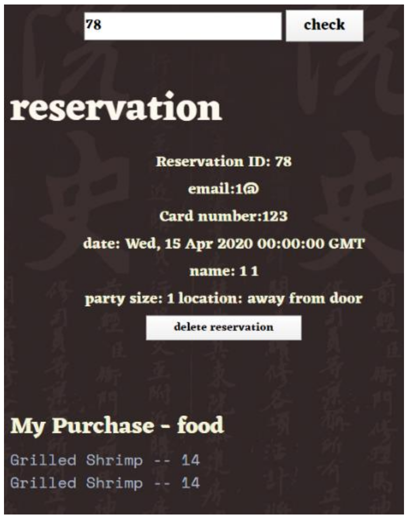


reservation

My Purchase - food

My Purchase - drink

If we enter 78, we can see the reservation information about ID 78 is displayed. This reservation has   ordered food.



reservation

Reservation ID: 78

email:1@

Card number:123

date: Wed, 15 Apr 2020 00:00:00 GMT

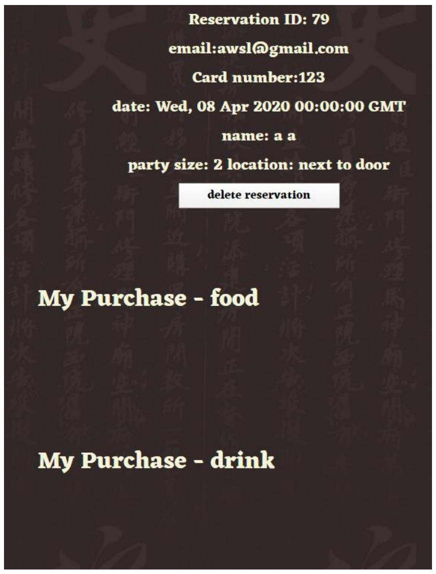
name: 1 1

party size: 1 location: away from door

My Purchase - food

```
Grilled Shrimp -- 14
Grilled Shrimp -- 14
```

If we enter 79 we can see this reservation ordered nor food or drink.



Reservation ID: 79

email:awsl@gmail.com

Card number:123

date: Wed, 08 Apr 2020 00:00:00 GMT

name: a a

party size: 2 location: next to door

My Purchase - food

My Purchase - drink