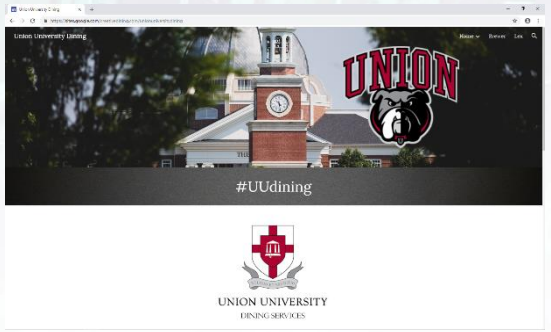
Lexington Inn Ordering App*

Joel White

Conceptual Outline



Web page form

Validation

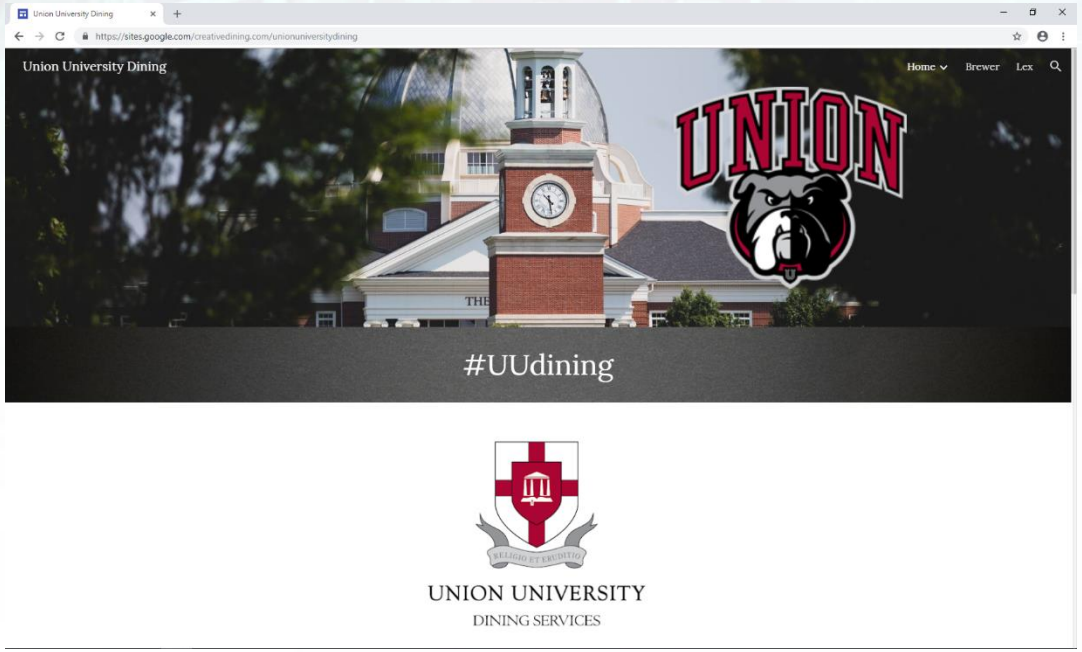
Union's server

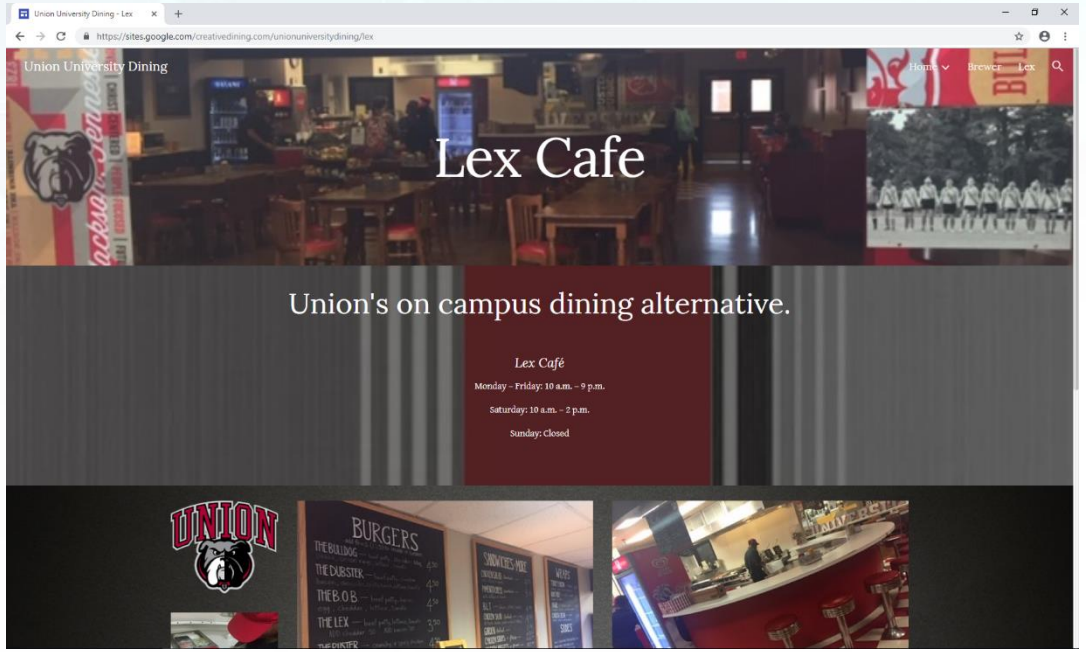
Receipt/confirmation

Transaction

Lex's cash register

The Current Website





Web Form Objectives

- Create a secure form for entering information
  - Student email and password (security)
  - Order
  - Student ID number
  - Time
- Restrict the form dynamically
  - Time must be within the Lex's hours of operation
  - Provide variable based time scheduling
- Verify User input (by validation with the server)
  - Validate user email and password with server.
  - Validate user ID
  - Check balance against order total
- Submit User input
  - Send order to Lex's register
  - Send User receipt via email

Creating the input Form

```
<form id="order">
    <h4> Lexington Inn Menu</h4>
    <h5>Burgers</h5>
    <p>The Bulldog - Beef patty, cheddar, BBQ sauce, onion rings, lettuce, tomato   4.50 <input type="checkbox"></p>
    <h4>Please enter a time for your food to be prepared</h4>
    <select id="weekday">
    <option id="defaultWeekday" value="-1">Please Choose</option>
    </select>
    <select id="time">
    <option id="defaultTime" value="-1">Please Choose</option>
    </select>
</form>
```

```
function loadMenu(){
    var n = 0;
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function(){
        if(this.readyState == 4 && this.status == 200){
            var myObj = JSON.parse(this.responseText);
            n++;
            console.log(myObj);
        }
    };
    xmlhttp.open("POST","\menu");
    xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
    xmlhttp.send(n);
```

Creating the input Form

```
51      } else if(request.url === "/menu"){
52        var requestBody = '';
53        request.on('data', function(data) {
54          requestBody += data;
55          if(requestBody.length > 1e7) {
56            response.writeHead(413, 'Request Entity Too Large', {'Content-Type': 'text/html'});
57            response.end('<!doctype html><html><head><title>413</title></head><body>413: Request Entity Too Large</body></html>');
58          }
59        });
60        request.on('end', function() {
61          fs.readFile("lexMenu.json",function(err,data){
62            var objArray = [];
63            var rawtxt = toString(data);
64            var length = parseInt(rawtxt.split("|")[0]);
65            var rawjson = rawtxt.split("|")[1];
66            for(x=0;x<length;x++){
67              objArray.push(JSON.parse(rawjson.split(",")[x]));
68            }
69            console.log(requestBody);
70            //if(x<length && x>0){
71            response.writeHead(200,'',{'Content-Type': 'application/json'});
72            response.write(objArray[0]);
73            //}
74            response.end();
75          });
76        });
77      }else{
78        response.writeHead(404, 'Resource Not Found', {'Content-Type': 'text/html'});
79        response.end('<!doctype html><html><head><title>404</title></head><body>404: Resource Not Found</body></html>');
80      }
```

Securing the Input Form

```
22  } else if(request.method === "POST") {
23    if (request.url === "/inbound") {
24      var requestBody = '';
25      request.on('data', function(data) {
26        requestBody += data;
27        if(requestBody.length > 1e7) {
28          response.writeHead(413, 'Request Entity Too Large', {'Content-Type': 'text/html'});
29          response.end('<!doctype html><html><head><title>413</title></head><body>413: Request Entity Too Large</body></html>');
30        }
31      });
32      request.on('end', function() {
33        var formData = qs.parse(requestBody);
34        response.writeHead(200, {'Content-Type': 'text/html'});
35        if(formData.email === "joel.white@my.uu.edu" && formData.password === "password"){
36          fs.readFile('Main Form.html', function(err, data){
37            response.writeHead(200,{'Content-Type': 'text/html'});
38            response.write(data);
39            response.end();
40          });
41        }else{
42          fs.readFile('login.html', function(err, data){
43            response.writeHead(200,{'Content-Type': 'text/html'});
44            response.write(data);
45            response.end('<p style="color:red">Incorrect Password or Username!</p>');
46          });
47        }
48      });
```

Sign in using your student email and password:

Email:

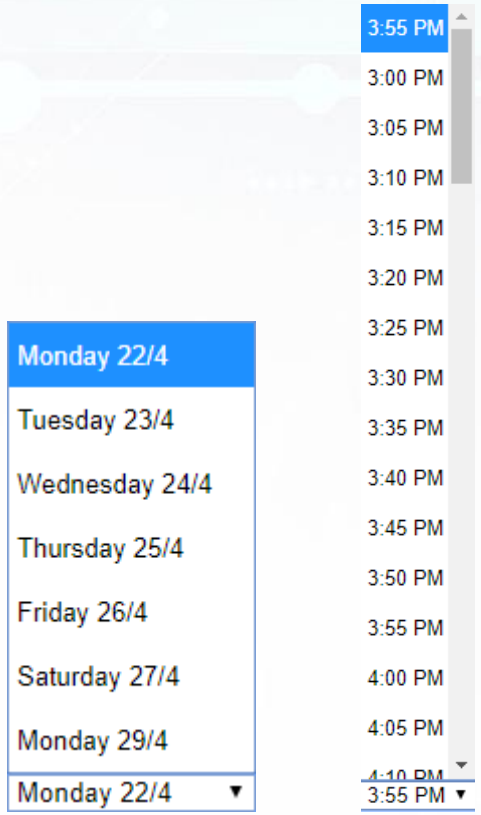
Password:

submit

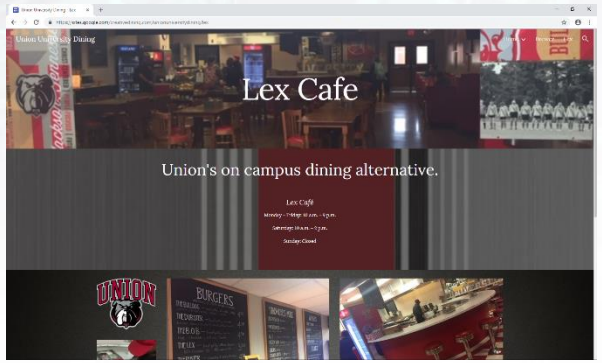
Form Variable Control

```
21          var minimumOrderSpacing = 5; //The minimum time in minutes between orders.
22          var earliestOrder = 5; //The earliest that a user can place an order in minutes from the current time.
23          var daysInAdvance = 7; //The maximum days in advance that a user can place an order.
```

Client-side JavaScript updates the html elements based on the current time and user based parameters.



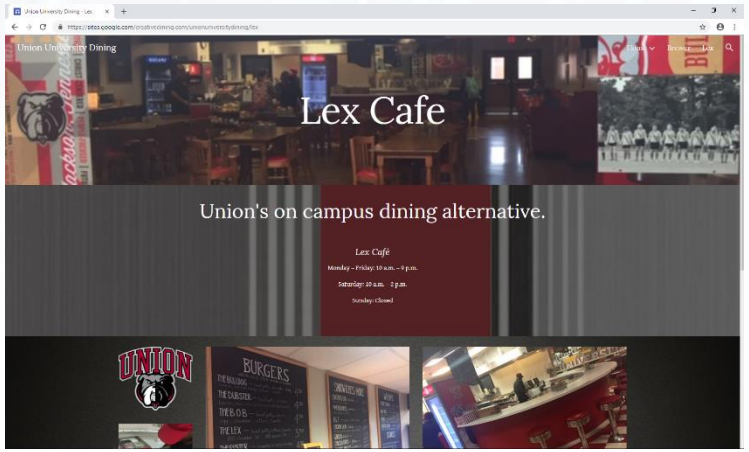
Verify User Input



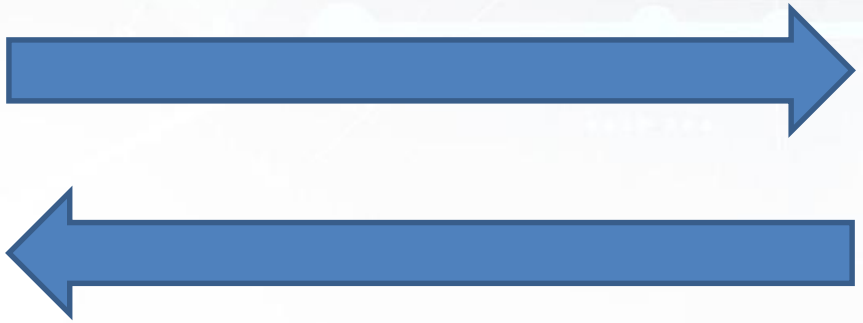
Student ID, email, password, order total

Confirmation that the student ID matches the email-password combination.
Student's balance is greater than the order total.

Submit User Input



Receipt of order with timestamp



Confirmation

Sends receipt to student email upon confirmation

Conclusion: Implementation

- The Lex’s register needs to interpret the data received and confirm with the webpage.
- The code needs to be altered to validate form content with the Union server.
- Both of these steps are straightforward with IT authorization.

Conclusion: Lessons Learned

- PHP
- Server/Client side programming
- Node.js
- Server Side JavaScript
- Document Object Model (DOM)