Team Number: 4 **Date:**

Lab Exercise: Ajax & JSONP

Lab Objectives

In Lab4, we have tried to take advantage of the default actions of the form submit button. In this lab, we need to exercise XMLHttpRequest API instead and learn to use JSONP.

Code snippet for the client-side script:

```
//Ajax request the next question
getJSONP("/EvalTool/JSONP/next"+uQuery, handleJSONP);
```

```
function getJSONP(url, handler) {
    // Create a unique callback name just for this request
    var cbnum = "cb" + getJSONP.counter++; // Increment counter each time
    var cbname = "getJSONP." + cbnum;      // As a property of this function
    // Add the callback name to the url query string using form-encoding
    // We use the parameter name "callback", which is required by Node.js.
    if (url.indexOf("?") === -1)   // URL doesn't already have a query secti
        url += "?callback=" + cbname; // add parameter as the query section
    else                           // Otherwise,
        url += "&callback=" + cbname; // add it as a new parameter.
    // Create the script element that will send this request
    var script = document.createElement("script");
    //script.type="application/javascript";

    // Define the callback function that will be invoked by the script
    getJSONP[cbnum] = function(response) {
        try {
            handler(response); // Handle the response data
        }
        finally {              // If handler threw an error
            delete getJSONP[cbnum];      // Delete this callback function
            script.parentNode.removeChild(script); // Remove script
        }
    };
    // Now trigger the HTTP request
    script.src = url;                    // Set script url
    document.body.appendChild(script); // Add it to the document
}
getJSONP.counter = 0;  // A counter we use to create unique callback names
```

Code snippet for the server-side script:

```
var app      = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extended: true
}));
app.set("jsonp callback", true);
```

Add this line to webserver.js to enable

On the server side, you can get the request

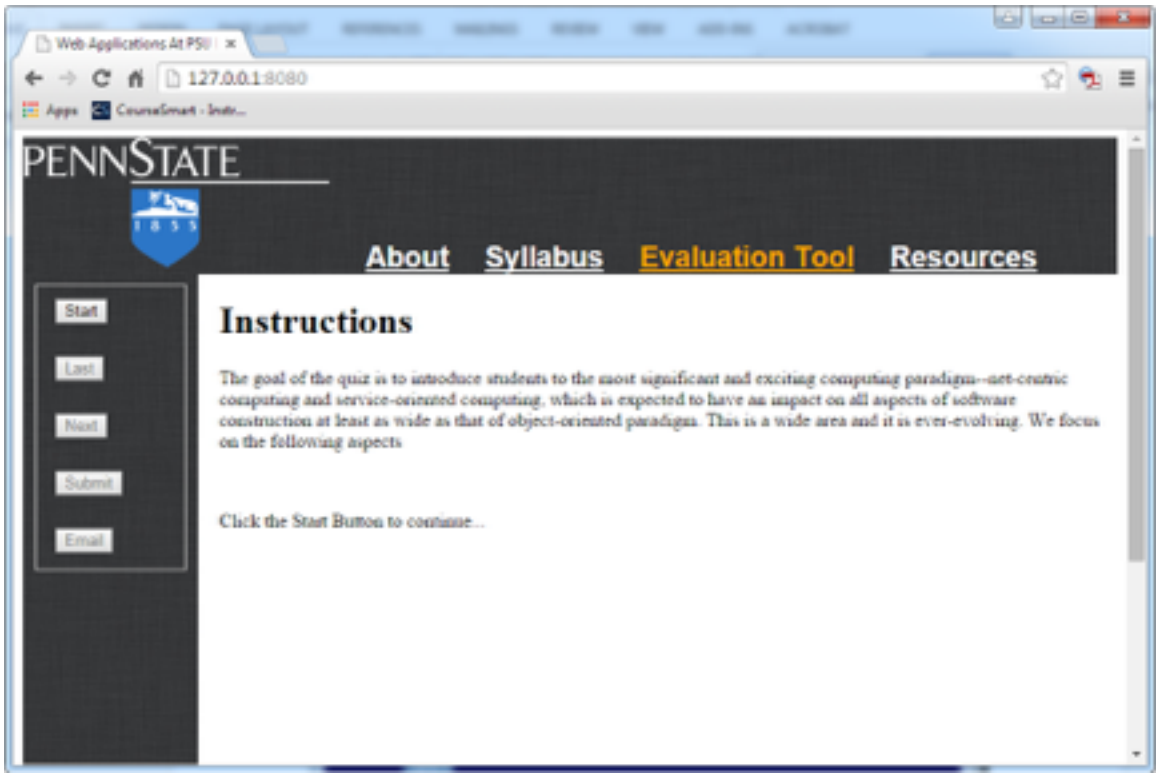
```
function handleJSONP(req, res){
    //console.log("JSONP Path: "+req.path);
    //console.log('JSONP params:  ' + JSON.stringify(req.params));
    //console.log('JSONP body:  ' + JSON.stringify(req.body));
    //console.log('JSONP query:  ' + JSON.stringify(req.query));

    if (req.path==="/EvalTool/JSONP/next"){
        if (req.query.answer!=null)
            updateAnswer(req.query.userID, req.query.answer);
        increaseIndex(req.query.userID);
        sendJSONP(res, req.query.userID);
    }
    else if (req.path==="/EvalTool/JSONP/last"){
        if (req.query.answer!=null)
            updateAnswer(req.query.userID, req.query.answer);
        decreaseIndex(req.query.userID);
        sendJSONP(res, req.query.userID);
    }

}

function gettool(req, res) {
    if (/JSONP/.exec(req.path)!=null) handleJSONP(req, res);
```

```
function sendJSONP(res, uid) {
    var nextObject = {};
    nextObject.Q = questions[getIndex(uid)-1];
    var pAns = getAnswer(uid);
    if (pAns!=='') {
        nextObject.previousAns=pAns;
        //console.log("## "+pAns);
    }
    //res.type('application/javascript');
    res.jsonp(JSON.stringify(nextObject));
}
```

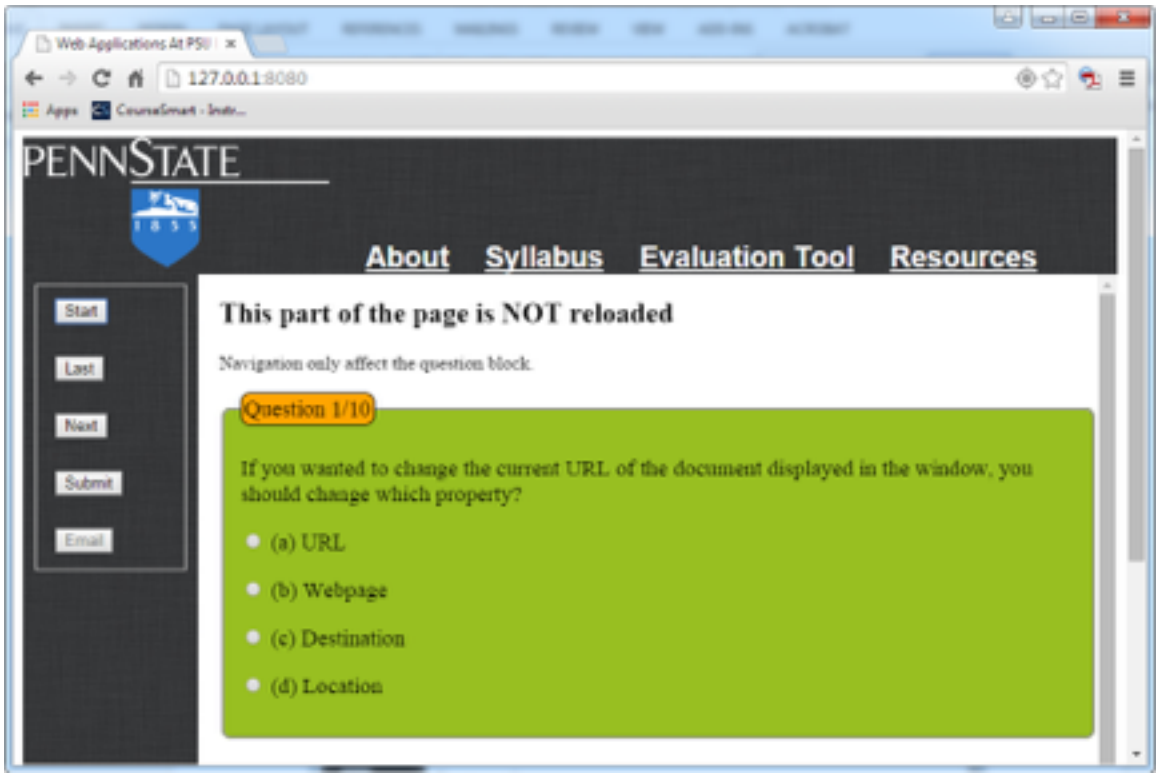
Send a JSON object by JSONP

Description of the Problem

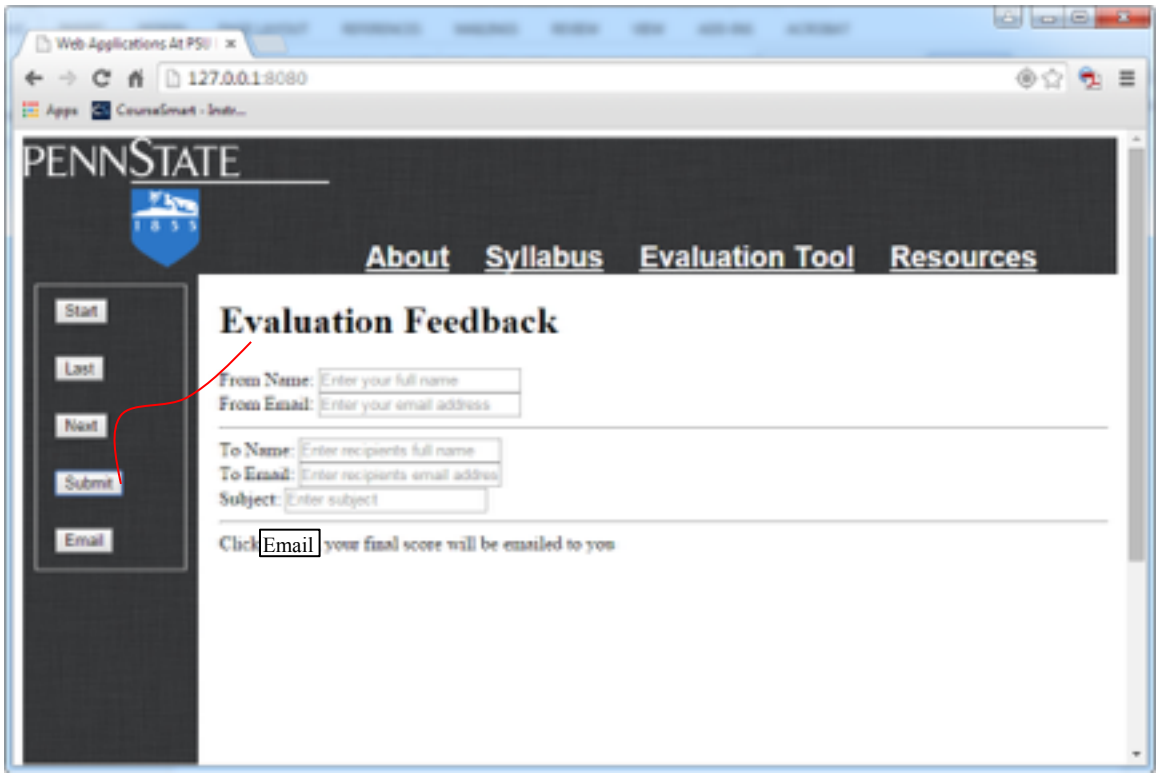
1. Continue to refine your web application.
    a. If your lab4 has bugs, fix it first.
    b. You are encouraged to apply creative styles.
2. In lab4, we used EvalTool as the folder. For this lab, use a different folder "**EvalJSONP**" for the evaluation feature.
3. [**10 points**] You should make the server flexible enough such that it can easily be configured to use either the implementation in EvalTool or the implementation in **EvalJSONP** (maybe just a different function call, or a few lines of code change).
4. The following requirements apply to the implementation in **EvalJSONP.**
5. [**20 points**] In a web browser, when a user clicks the "Evaluation Tool" link in the navigation bar, the client should load the /EvalTool/firstpage.html to the content iframe. Upon load complete, the script should load /EvalTool/side.html to the side navigation region. This side.html should include scripts for all Ajax processing, and it should support typically functions for using the evaluation tool, such as Start, Pre, Next, Submit, Email, goto X, etc.



6. [**20 points**] Extend the evaluator tool of your web server to support multiple users from the same client computer (they may be using different browsers or different tabs of the same browser). To implement this feature, you need to use sessionStorage to uniquely identity a user of the evaluator page on your website. When a user clicks the "**Start**" button, the browser should
   a. Generate a unique user ID for that user, and save the user ID to the **sessionStorage** object.
   b. Assume that all the buttons displayed in the side region are components of a form, you can add a hidden input element to the form, and set its value to the generated user ID. This user ID will be sent together with a GET request (for the question page /EvalTool/QuestionPage.html) to the server. Upon this request, the server should create a record for this user (all the user's activities should be associated with his/her user ID), and send /EvalTool/QuestionPage.html back to the user's browser. QuestionPage.html contains the first question, but will be used as a template for all the ensuing questions as the user navigates back and forth.



7. [**20 points**] Use **JSONP** for the buttons "**Last**" and "**Next**". For example, when a user press "Next", Your scripts should generate a JSONP request containing information such as user ID, user's answer, and navigation command. Once the server receives the JSONP request, the information will be used to direct the server to respond appropriately (say, save user's answer, send a new question back). When the client receive the server's response, it should update the question block to display the new question. If the user has answered that question before, the corresponding radio button should be checked. Note that this time you should NOT save user's answer at the client side.
8. [**20 points**] Use **XMLHttpRequest POST** for the buttons "**Submit**" and "**Email**". You should use the **JSON** format for the request body. The "Submit" button click should lead to the load of /EvalTool/ summarypage.html page and the "Email" button click should lead to the load of feedback information (say, scores earned).



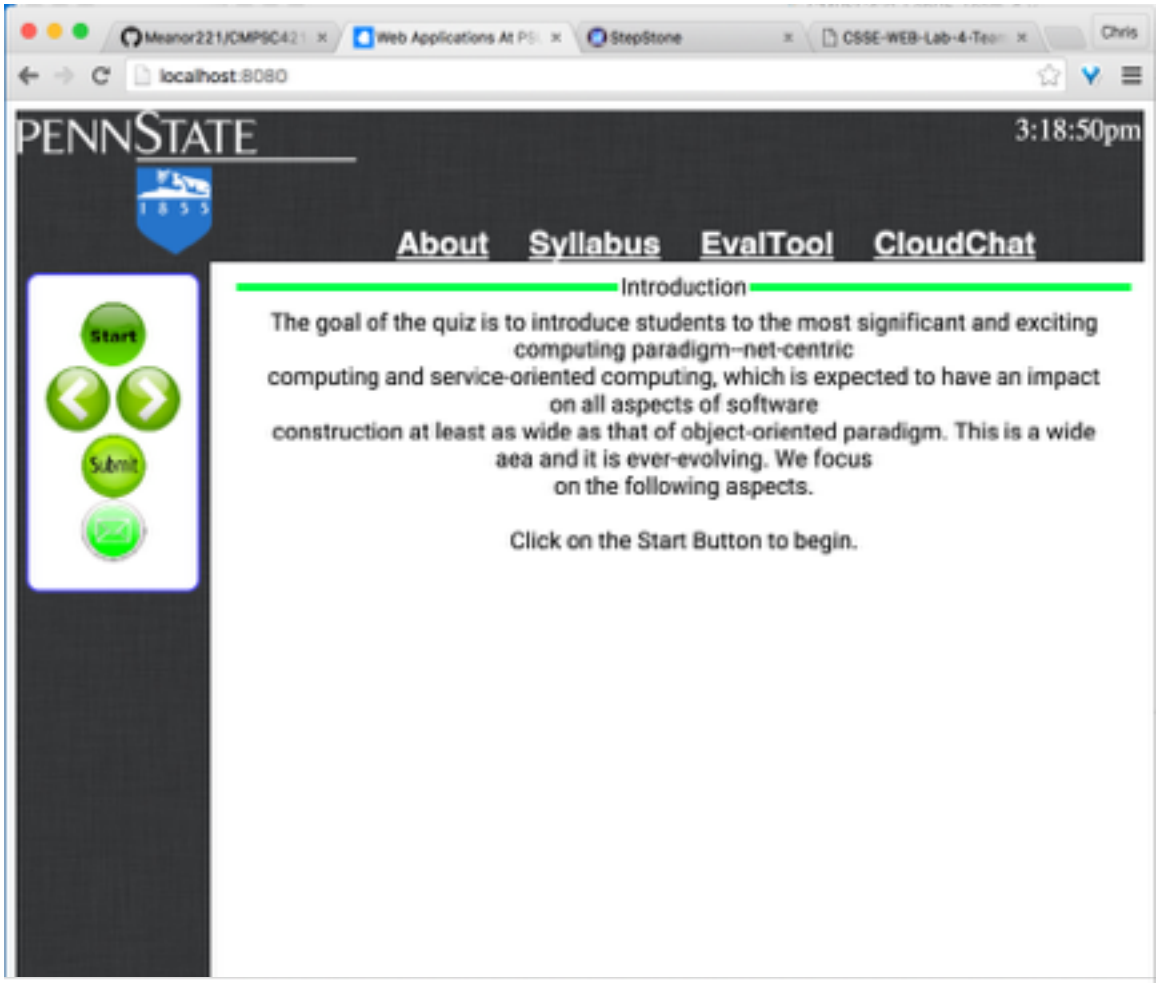
9. [**10 points**] For better user experience, the buttons in the side region should be enabled or disabled appropriately. For example, enabling the "submit" function only when it comes to the last question.

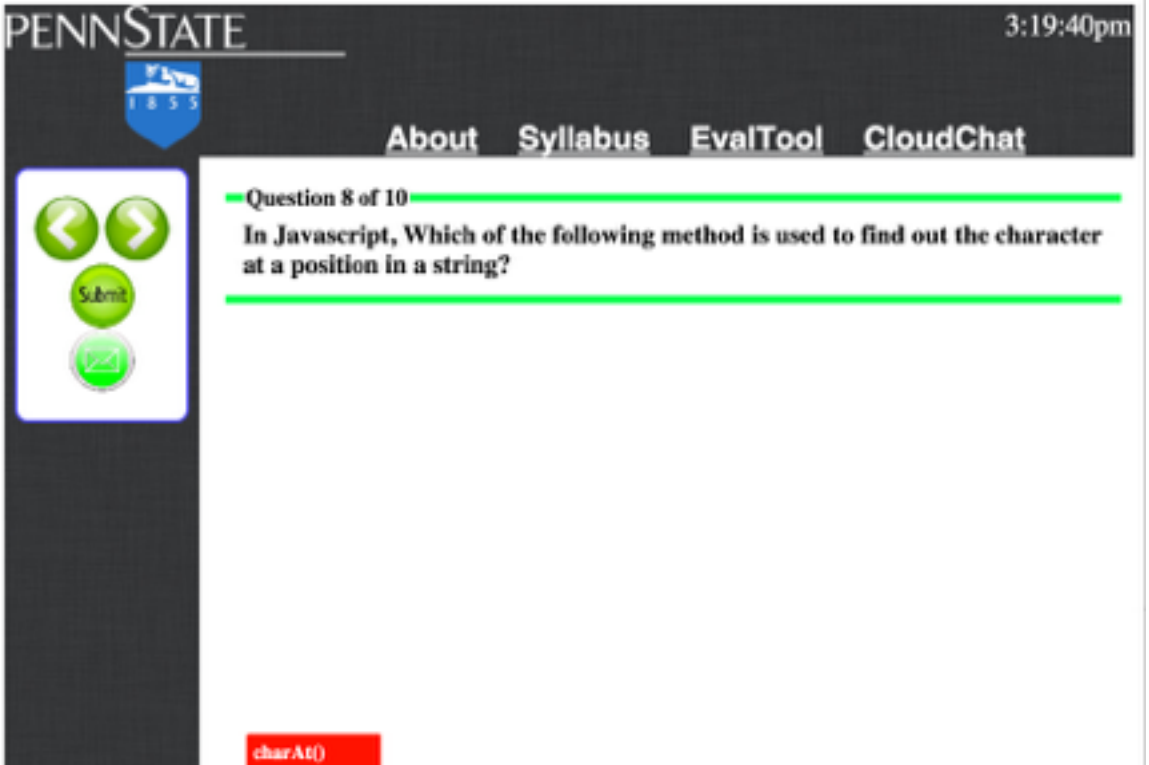
Lab Submission:

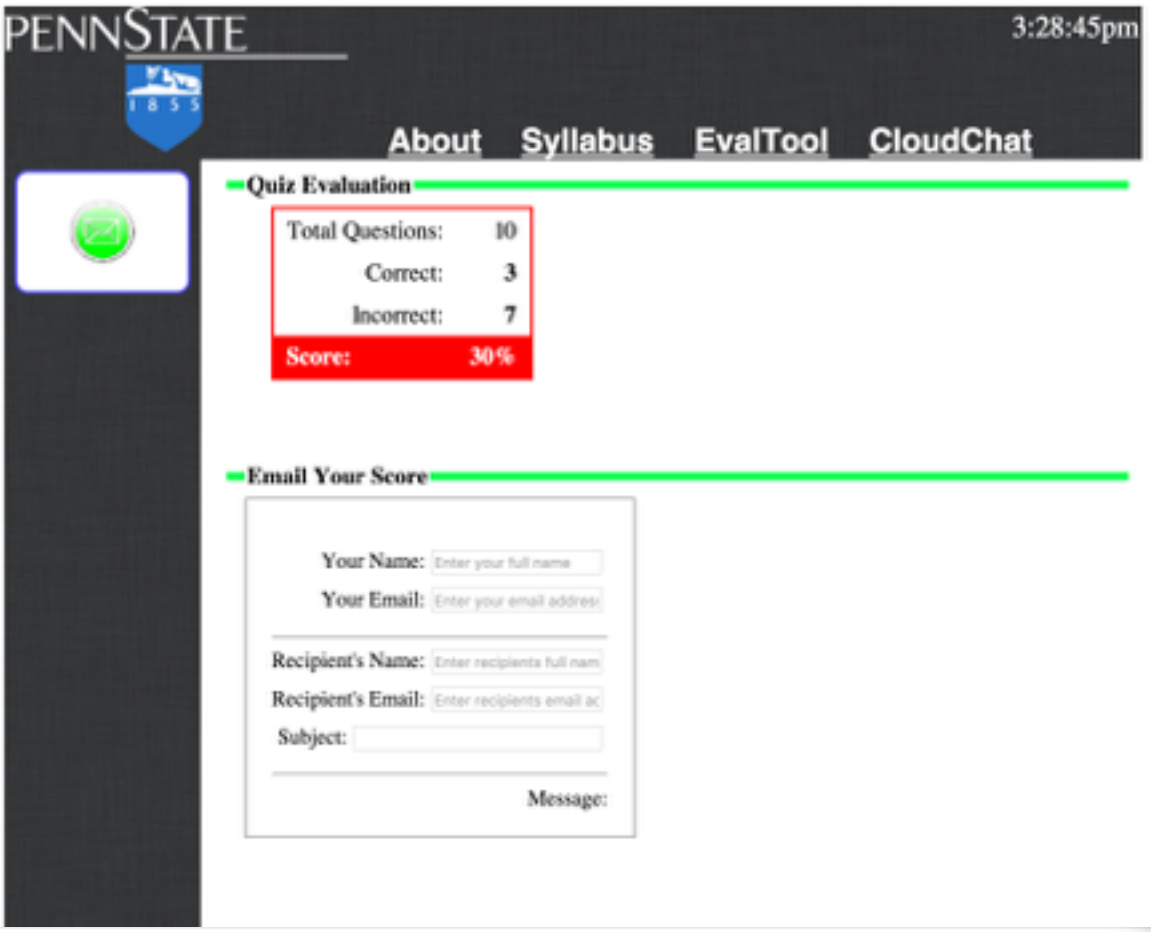
1. **Provide individual performance in the table below (performance factor is a real number between 0.0 and 1.0, individual grade is lab grade times his/her performance factor). [Check the "ACM code of ethics, if you do not know it. If you do not honestly record teammates' performance factor, you are actually encouraging them to be lazy. In doing this, you are NOT helping them, but preparing them to fail in their career.]**

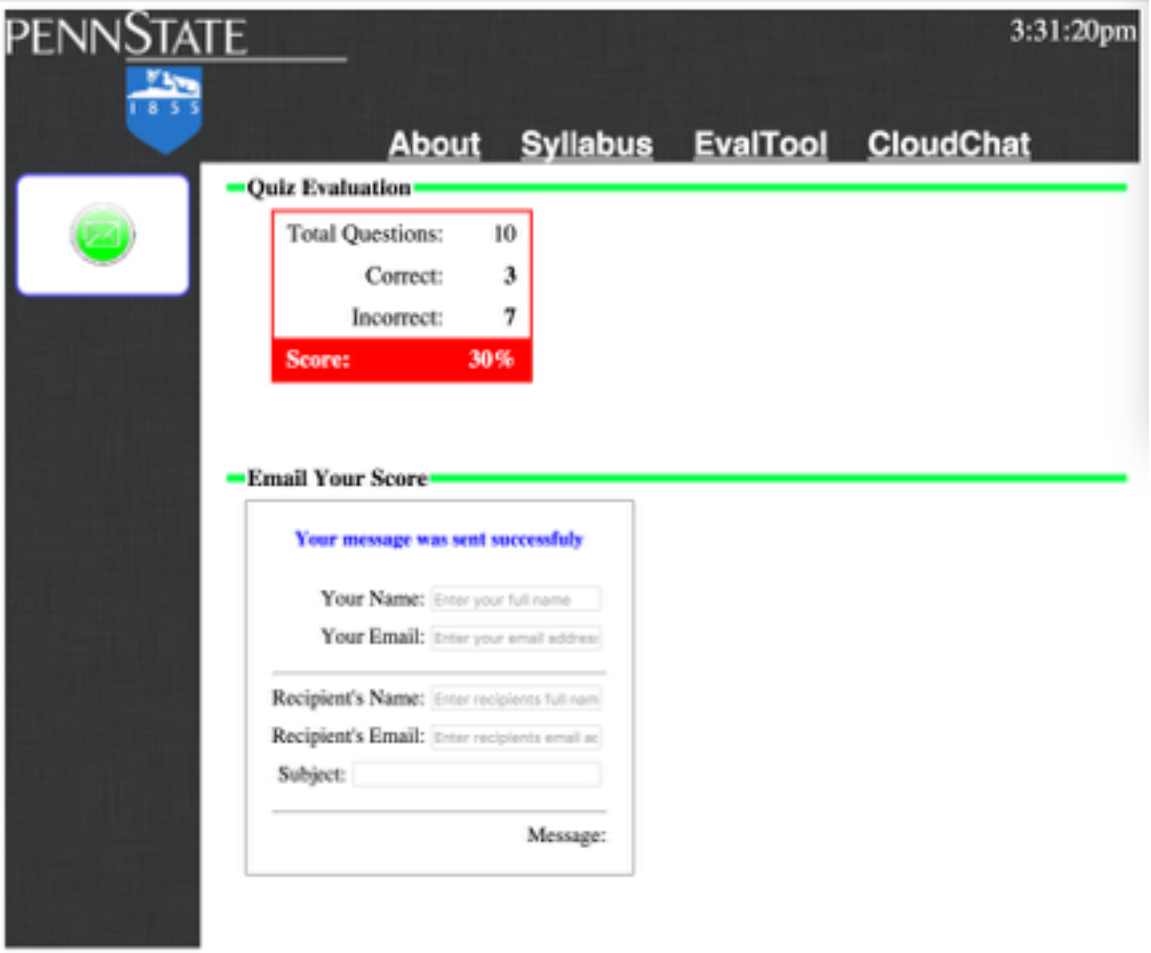
Student Name	Performance factor
Chris Andrejewski	0.34
Adam Meanor	0.32
Jordan LaRiccia	0.34

2. Paste a few screenshots inside this lab report to demonstrate how it works.






PENNSTATE
1855
3:28:45pm
About
Syllabus
EvalTool
CloudChat
Quiz Evaluation
Total Questions: 10
Correct: 3
Incorrect: 7
Score: 30%
Email Your Score
Your Name:
Your Email:
Recipient's Name:
Recipient's Email:
Subject:
Message:


PENNSTATE
1855
3:31:20pm
About
Syllabus
EvalTool
CloudChat
Quiz Evaluation
Total Questions: 10
Correct: 3
Incorrect: 7
Score: 30%
Email Your Score
Your message was sent successfuly
Your Name:
Your Email:
Recipient's Name:
Recipient's Email:
Subject:
Message:

3. Save this report to a PDF file with the name **CSSE-WEB-Lab-5-Team-X.pdf, where X is your team number**;
4. Submit to Angel->Labs->Lab1 -> Submissions. Your submission should be one zip file with the name **CSSE-WEB-Lab-5-Team-X.zip** (where **X** is your team number) that includes:
    a. **CSSE-WEB-Lab-5-Team-X.pdf**;
    b. The Website folder, excluding the sub-folder node_modules generated by npm;