Project created on 09.09.2016 00:08.

Report for project Senior Design

Task created on 24.02.2017 05:46.

Verification and Validation

No due date

No description

Task tags: *No tags*

Brainstorming Created by Yanlin Ho on 24.02.2017 06:28.

We have begun brainstorming methods we will use to validate whether or not our website meets the needs specified by the client early on in the project. Our preliminary ideas are to have the teachers use the website during certain class periods, then answer a questionnaire that will specify issues in their ease of use. After verification that our website indeed addresses their needs, we will then move forward into trial sessions with the students, to see whether the website is capable of keeping their attention span, easy for them to use and navigate.

Task created on 10.02.2017 00:22.

Website Development

No due date

We will be coding the website in html, css, and javascript according to our finalized mockups.

Task tags: *No tags*

Student Homepage (modified, week of 2/23) *[hmpg_student.png]*
Uploaded by Tong Yu on 24.02.2017 00:54.

Comments for result Student Homepage (modified, week of 2/23)

Tong Yu on 24.02.2017 at 01:00: The teachers wanted audio as well as visual cues for the buttons. I recorded myself saying ‘Songs’, ‘Videos’, ‘Games’, and ‘Stories’ on my phone and incorporated the m4a clip to the page. Now, when the mouse hovers over each button, the audio clip automatically plays. I’ve spaced out the buttons more so students can see the buttons more separately. In addition, the teachers wanted a picture/symbol to accompany each button. For now, I’ve put a placeholder photo (music symbol) inside each button. The teacher preferred not to have the settings button for the students to use, so I’ve removed that from the bottom right.

Modify Songboard page *[manage_songboard.png]*
Uploaded by Tong Yu on 24.02.2017 00:54.

Comments for result Modify Songboard page

Tong Yu on 24.02.2017 at 01:20: I created the “Manage Songboard” page. There is a song selections form that the user can check off. When the ‘submit’ button is pressed, song names that are checked off are presented in text format at the bottom, after “Current playlist:”. Upon clicking the Reset button, all boxes are unchecked. There are also preview buttons next to the first two songs’ names. Upon clicking ‘preview’, the Youtube video of the song is displayed at the top of the same page.

Teacher Page *[teacher.png]* Uploaded by Tong Yu on 24.02.2017 00:54.

Comments for result Teacher Page

Tong Yu on 24.02.2017 at 01:13: I created the teacher page. On the sidebar, there are option buttons like “Manage Videos” and “Manage Games”. When the selections are moused-over, they are highlighted in yellow. When one is clicked, that active selection’s button becomes orange and the title of the option is displayed at the top of the middle section (gray, main part). Upon clicking the “Manage Songboard” option, the user is directed to the ‘Manage Songboard’ page. The turquoise color scheme was chosen to match the Updated Activities Menu.

Activities Plan (modified, week of 2/23) *[AP.png]*
Uploaded by Tong Yu on 24.02.2017 00:53.

Comments for result Activities Plan (modified, week of 2/23)

Tong Yu on 24.02.2017 at 01:05: I improved the design of the first Activities Plan draft by aligning the buttons more and spacing out the side bar from the main screen so there is less distraction for the students. I put in a trial video (a Youtube video) in the main section that is embedded within the page.

Installing Node JS Created by Carlie Abraham on 24.02.2017 06:46.

Node.js is a JavaScript runtime environement for a diverse array of server tools and applications. It allows for the installation of node modules to run on a webserver, allowing many different functions that would otherwise be impossible using html and css. Specifically, we will be using Node.JS for non-blocking I/O. This means that many requests to the server can be performed without blocking or stalling anything running client-side.

Node.js was successfully installed on our server. In order to do this, first some work had to be done on changing security access to the server. This was done by opening a few ports to the public. A simple program (hello.js) was used to test if the Node.js server was working:

var http = require('http');

http.createServer(function (req, res) {

```
res.writeHead(200, {

    'Content-Type': 'text/plain'

});

res.end('Hello World\n');
```

}).listen(3456);

Port 3456 had to be opened in order for the public to have access to the content in hello.js. Hello.js needs to be run server-side with the command “node hello.js,” and once the server is running properly, the content can be viewed on the website.

❋ Serving Static Files with the mime Package

Created by Carlie Abraham on 24.02.2017 07:11.

Running a Static Fileserver via Node.JS

Node.JS does not automatically allow access to static files on the server (such as html files). In order to serve up these files, another package needs to be installed with the Node JS package manager. The “mime” package can serve up files similar to the Apache/PHP scheme as was previously used to access html files on the server. To test if the mime package was working properly, the following bit of code was used:

var app = http.createServer(function(req, resp){

```
// This callback runs when a new connection is made to our HTTP server.

fs.readFile("mongoTest.html", function(err, data){

    // This callback runs when the client.html file has been read from the filesystem.

    if(err) return resp.writeHead(500);

    resp.writeHead(200);

    resp.end(data);
```

});

app.listen(3456);

This code will serve up the file “mongoTest.html,” so that it can be accessible to the public. This was confirmed to work through tests. It is also possible to serve up multiple files, even directories of files, but these tests on the mime package are not shown above.

❋ Installing Socket IO Created by Carlie Abraham on 24.02.2017 07:20.

Societ.IO allows for asynchronous communication between a server and a client. This is important as information will be requested from a database on the client side, and the server will have to process and return that information without blocking the client side from performing other actions. Because data may take a few seconds to process (or longer, but this is unlikely), it is important to make sure that any requests to the database server is done asynchronously, as with Socket.IO.

Socket IO was installed using the Node JS package manager onto the server. A simple test was done on the server to prove that it was up and running. First, a script in the client-side html file (mongoTest.html) was written as below:

```
var socketio = io.connect();

socketio.on("connect",function() {
    alert("here");

});

function myFunction(){

    alert("my func");

    socketio.emit("getData");

}
```

Second, a script was written in the JavaScript file that is run on Node JS to serve up the file mongoTest.html, and allow for communication between client and server via Socket.IO. Some of the code from this file is written below::

var io = socketio.listen(app);

io.sockets.on("connection", function(socket){

```
console.log('a user connected');

socket.on('getData', function() {

    console.log("in get data");

    console.log("hey there");

});
```

});

In this piece of code, the client side will send out a “getData” message to the Node JS server, which will be received via Socket.IO, where processing will occur. This was confirmed to work, showing that Socket.IO was installed properly and is running correctly on the server.

Installing and Testing MongoDB Drivers on Node JS *[server.js]*
Uploaded by Carlie Abraham on 24.02.2017 07:27.

Comments for result Installing and Testing MongoDB Drivers on Node JS

Carlie Abraham on 24.02.2017 at 07:31: To allow access to the MongoDB database, the MongoDB drivers for Node JS were installed using the Node JS package manager. After this was completed, a test was performed to see if data was being accessed properly from the database. In this file, when “getData” is called from the client-side, a call will be made to the MongoDB database “test,” which is already on the server. “test” was a sample database used to initially test that MongoDB was installed on the server. Now, via Node JS, the test database was accessible, and a simple query was performed to find all restaurants from the Manhattan borough from the test database that contained restaurant information. The console returned that no errors occurred and the data was accessed properly.

Client Feedback 2/17/17 Created by Carlie Abraham on 24.02.2017 06:35.

Student Homepage:

• Client liked the bright colors, high contrast

• Liked the simplicity of the page – little distractions, easy to navigate

• Liked the size of buttons, number of buttons on the screen

Teacher Login

• Client wanted logins for each teacher, but not necessarily logins for students

• After teacher logs in, they should be presented with a list of students to chose from – clicking on a particular student will “log them in,” meaning the teacher will have control over who is accessing their account before they use it

Activity Menu Page

• Liked that there was a small button in the corner to bring up the menu to go to other pages on the website

o Students wont need to click on this button, so it should be kept small and not very noticeable

o Teachers will be the ones to use the menu to navigate through the website

• Display of activity selections could be made a little bigger – too many buttons displayed on the page

Activity Plan

• Some formatting issues of the page

• The default should not allow students to see the left hand menu on the screen when they are progressing through their activity plan

• Like the activity menu page, the scheduled list of activities should be part of a sidebar that is only selectable by a small button in the top left hand corner, which will only need to be pressed by students

General Notes and Comments

• All settings to control the website (background color schemes, size of buttons, content available to students) should only be able to be accessed by the teacher on their own login page

o Student access to some settings such as background colors was discussed, but ultimately the teachers believed that it was only necessary for the teachers to change student user settings

• Sound should be incorporated into the website – for example, when pressing a button, the name of the button should be spoken in order to reinforce a word with a specific action

• Client is working on gathering some data to put on the website such as animated stories, videos, and other content that is not copyright protected

• Activity plans should be assigned by either class, or by individual student – this ensures the easiest way to plan out lessons and activities in the students leisure time

❋ Connecting the website Created by Yanlin Ho on 24.02.2017 05:45.

Work was also done on connecting various parts of the website. We started inputting images into the blank squares previously seen on the activity menu page, which linked to various different interactions. We also connected all currently designed pages so that they are accessible from anywhere on the website.

❋ Interaction Page Created by Yanlin Ho on 24.02.2017 05:43.

An interaction page was created, where the actual activities would be carried out. There was an issue regarding embedding videos from Youtube, since some of the copyrights from companies on their music videos meant that the videos could not be shown on our website. However, further research showed that this problem should be fixed as soon as we obtain our domain name, as Youtube is not allowing access to the private server, since Youtube cannot detect its location. A javascript function is being created to allow for different content to be input into the same stock interaction page.