|  |
| --- |
| TMA4: Application development for the internet |
| BIT695 Web Technologies  Due Date June 11 2017 |
| Penelope Ann Williamson  3431274 |

|  |
| --- |
|  |

## **Task 1: Security and availability**

**a)** The case study web application we were given for TM1 and expanded on for TM2 was a simple web application, with no login screen, session ids or authentication. It does contain a sign-up form with form side validation, that was expanded to contain server side validation in TM2. We were also asked to develop an edit/update and delete page that used retrieved data from the Players database.

The lack of login, session id, session management and authentication means that none of these can be hacked, by something like a SQL injection attack, such as id = 1+1, or miss use of a session id token. Lack of login also means there are no passwords or user names to encrypt or hash and salt. The lack of tokens means there is no exploit for cross site forgery.

The web application as it is given contains very little in the way of exploitable security flaws, with the exception of the form template. While the form template has form side validation, the e-mail address field, and the phone number field both allow non-standard characters, using a simple regular expression check. This still allows both SQL injection attacks and cross site scripting attacks (XXS) to exploit these fields. The e-mail field is especially vulnerable to this as it allows the use of the @ and . symbols, making the use of a malicious web address, or SQL code possible

The server side validation that was added in TM2 attempts to address these problems, by introducing server side validation. However, as special characters are not explicitly escaped in the e-mail field this vulnerability still exists, although the use of the FILTER\_VALIDATE\_EMAIL does lower the risk.

The lack of a login, especially in the modifications for TM2, allows for data privacy breaches, with the edit and update for retrieving the contents of the entire players database, which exposes members names, e-mail addresses and phone numbers. This opens it up to data theft, identity theft and gives hackers information for direct attacks on members, using information that is contained in the form as a starting point.

The fact that the edit and update players can be used to change details and delete players, with no controls in place means that the data integrity of this site and the databases it uses, is questionable, and open to malicious attacks.

The site needs to address the issues of data integrity and security, before adding in the desired components. It also needs to look at using a parameterized API interface for the form data, or using escaped characters for the e-mail and phone number fields (albeit the risk is far lower with the phone number fields).

(The OWASP Foundation, 2014)

(The OWASP Foundation, 2017)

(Word Count 446)

**b)** E-commerce relies heavily on the availability part of the CIA triad of confidentiality,integrity and availability. When an e-commerce site is unavailable, the company losses money, and can lose customers (about half of an e-commerce sites customers will only wait three seconds before moving on (Ali Hodroj, 2015)

From within the context of the CIA triad, the online realtor behind the e-commerce site needs to maintain reliability of service, and maximize the uptime. Downtime and timeout errors, not only cost a company directly in terms of sales (even forcing them out of business), but also affect the perception of the sites integrity from the perspective of the customer. After all, a site plague by timeout errors and downtime (or outages), loses the trust of the consumers.

Availability is an important aspect of an e-commerce sites functionality, and should be managed via a multi-pronged approach using such tools and strategies as redundancy, using a server system that can rapidly scale to handle extra demand and maintaining the server or RAID systems that are used. The failure to minimize downtime and maximize uptime can have a large impact of the perceived integrity of a site, the sales figures (and profit margin), which can ultimately impact on the long-term survival of the e-commerce site.

Within the terms of today’s applications, the CIA triad model has limitations, and needs updating. E-commerce consumers are more aware of security issues, and take into account a sites reputation before trusting it with its credit card details.

Nonrepudiation is a concept that is missing from the traditional CIA triad. We need to be able to trust that the site we are interacting with is who they say they are, and the site needs to be able to trust that the ‘person’ they are interacting with is not only who they say they are, but also a human.

In today’s world of identity theft, scams and fraud, which utilize, or fool websites, we need to be confident that we can ‘act in good faith’ when dealing with a faceless, largely unknown entity. Nonrepudiation can provide us with the level of trust that we need. Some sites are already doing this by using things like two factor identification for account verification and logging on. This is a factor that is of growing importance to people, as more and more sites require our credit/debit card numbers and a large amount of personal details, alongside an awareness, and media coverage of, of scams and identity theft.

(Charlton, 2010), (Henderson, 2017), (Margaret Rouse, 1999 - 2017)

Word Count 411.

## **Task 2: Version control**

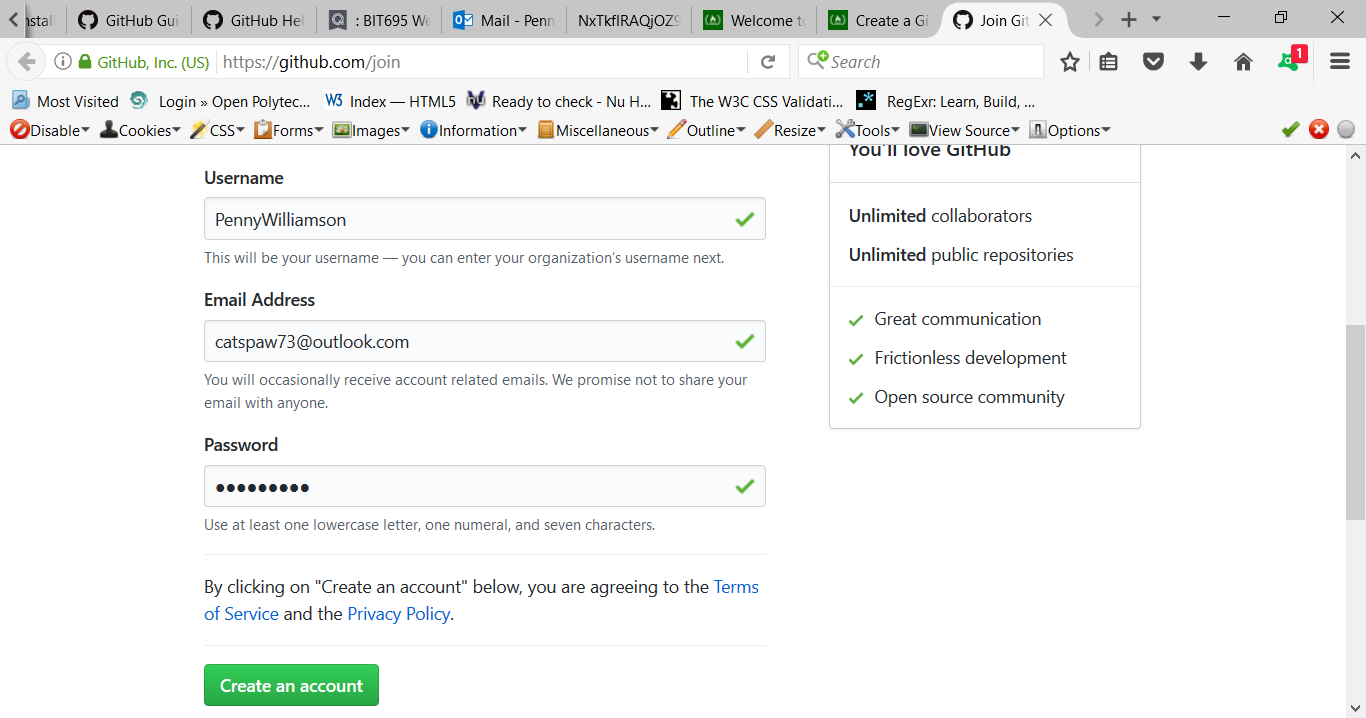
**a)** 

figure 1. Setting up GitHub account

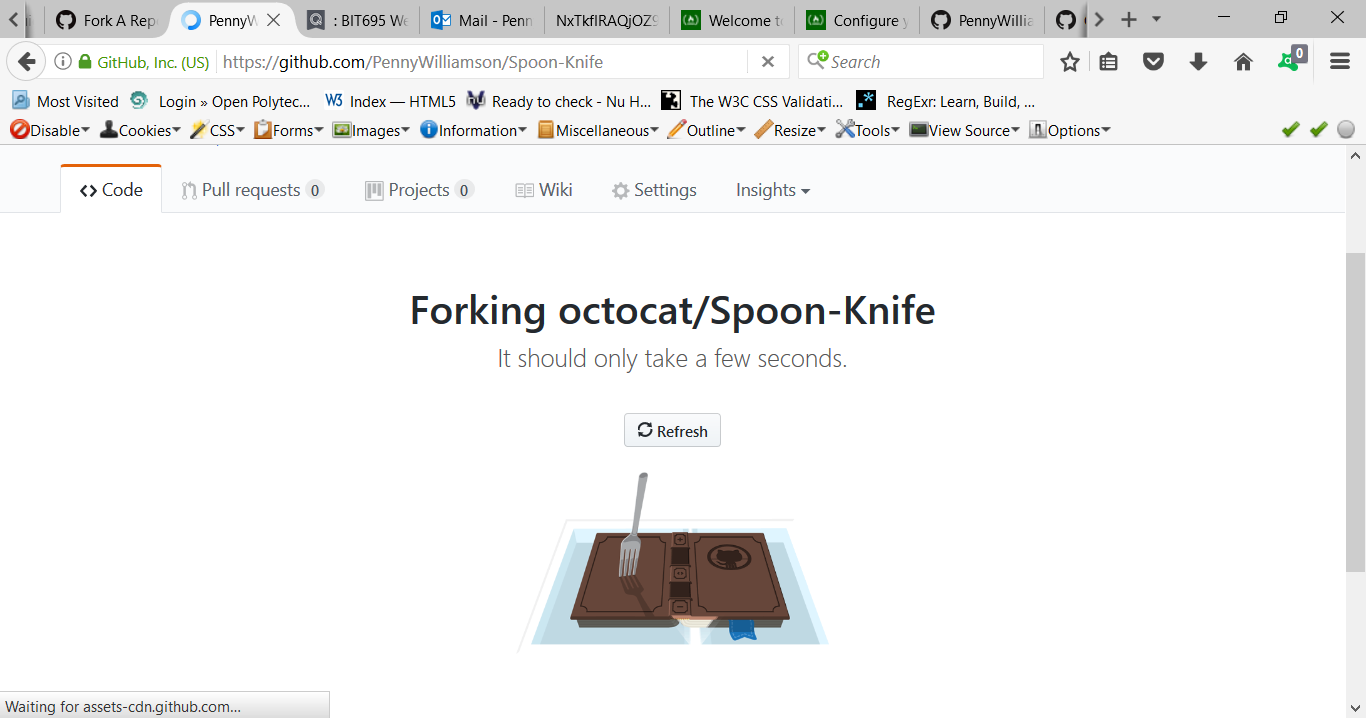


Figure 2. Forking the octocat/Spoon-Knife repository

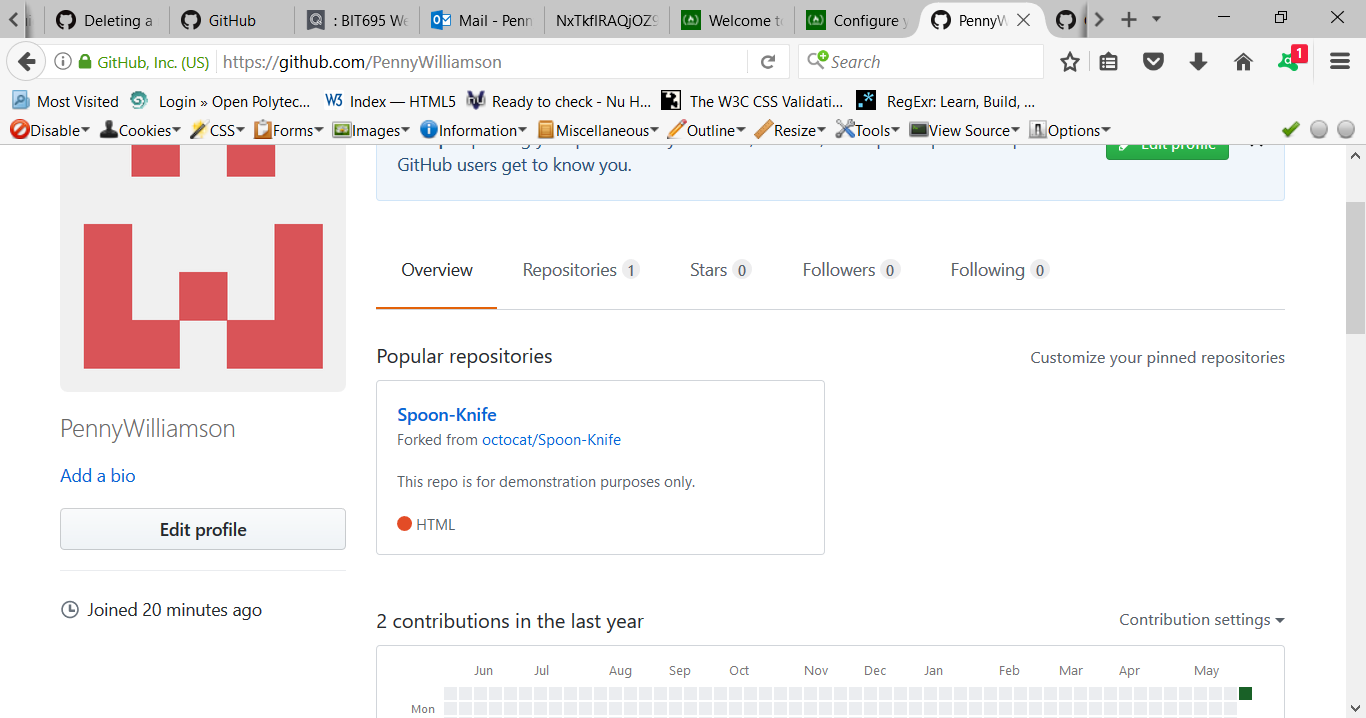


Figure 3. The forked Spoon-knife repository.

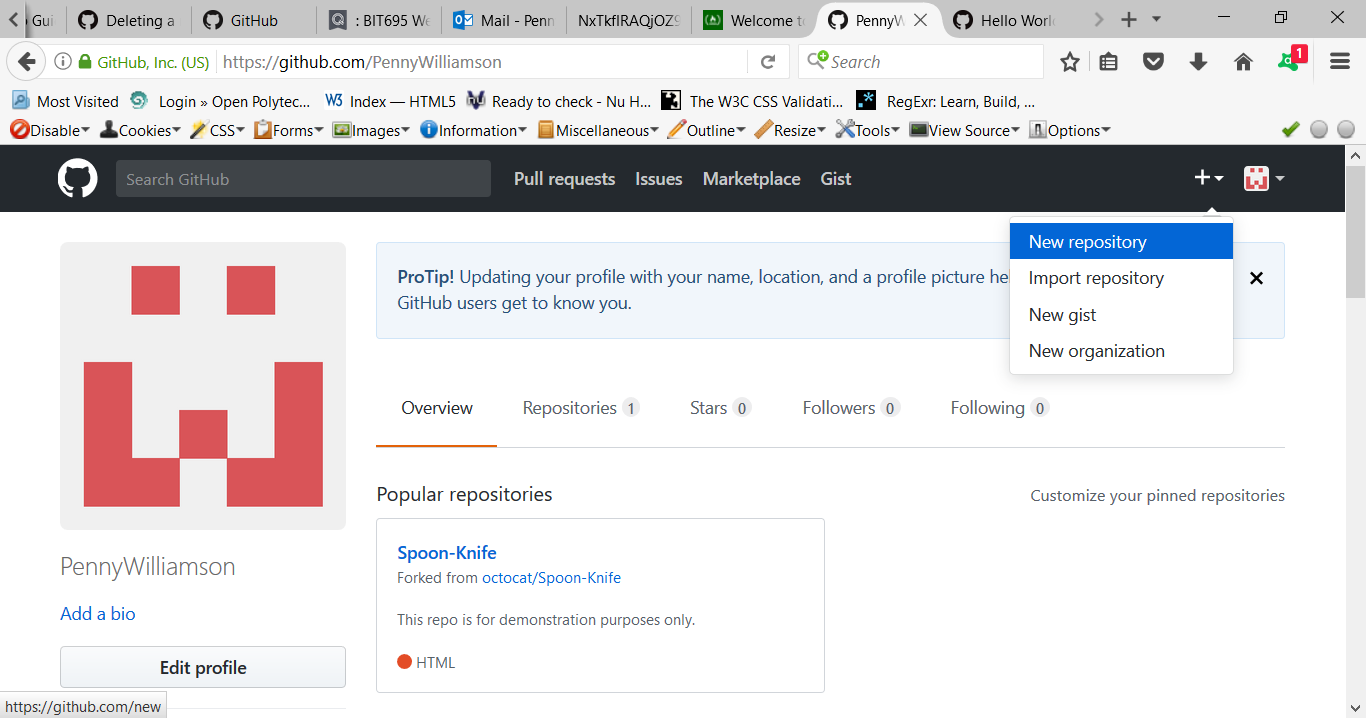
**b)** 

Figure 4. Choosing new repository from menu.

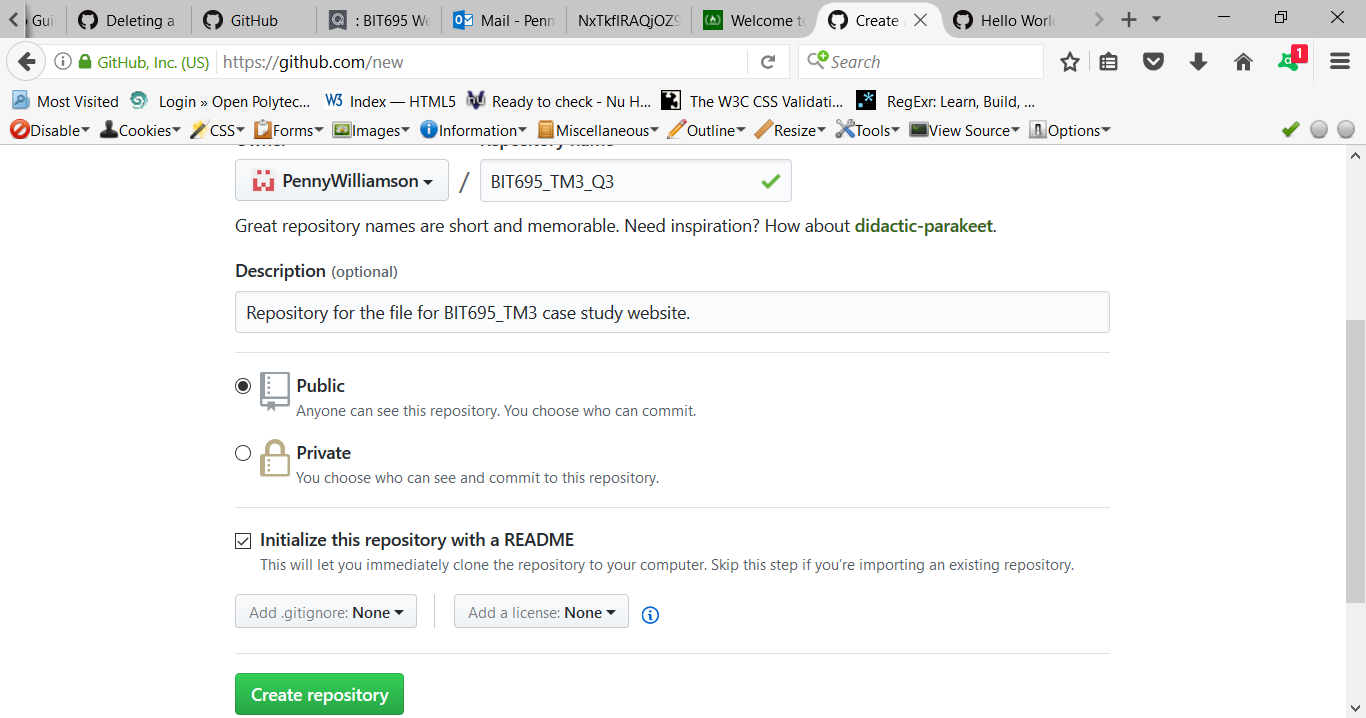


Figure 5. Setting up new repository for Task 3 code.

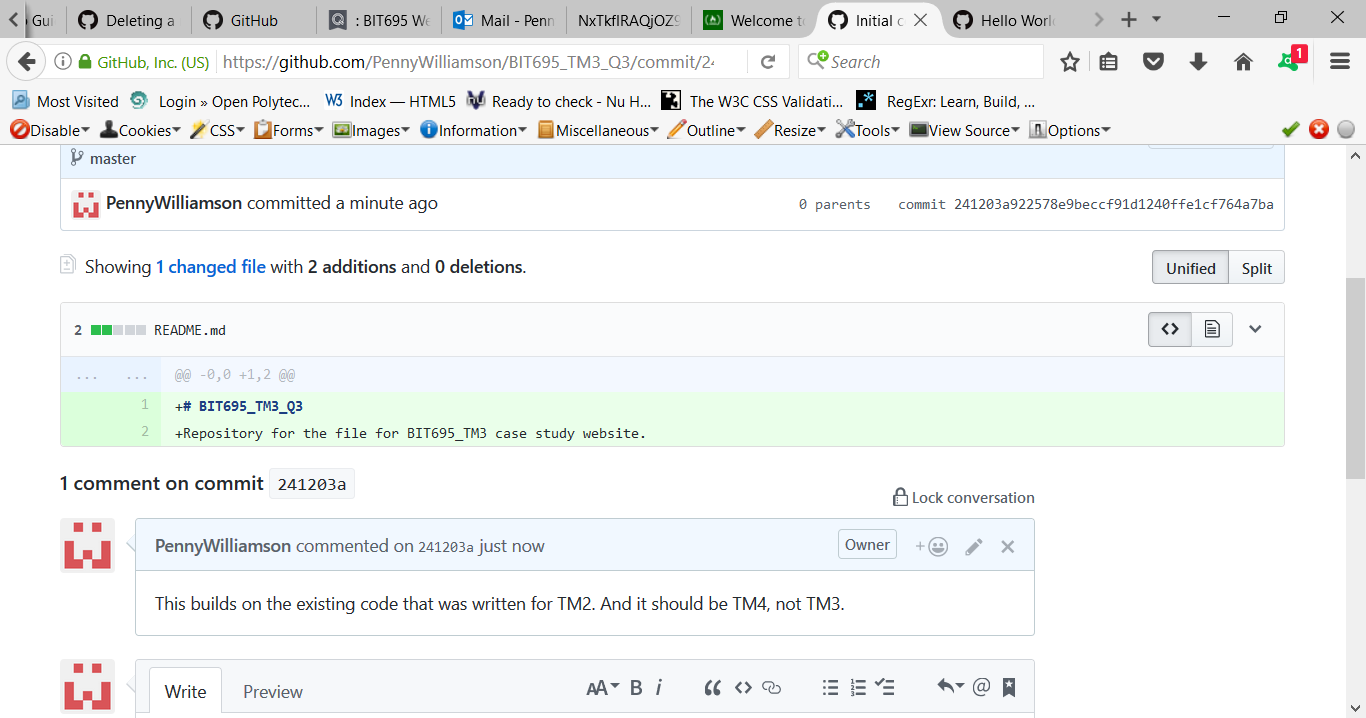


Figure 6. Commenting on the commit, shows the repository.

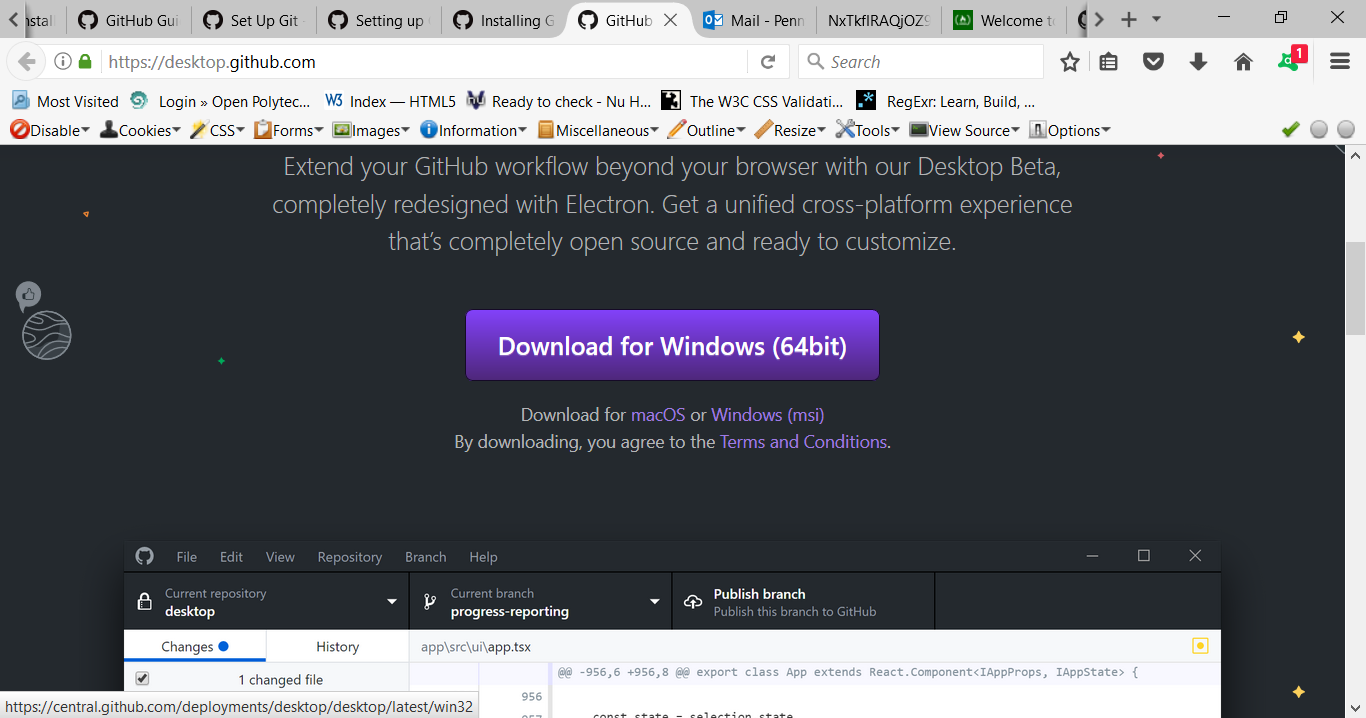


Figure 7. Download button for GitHub desktop client.

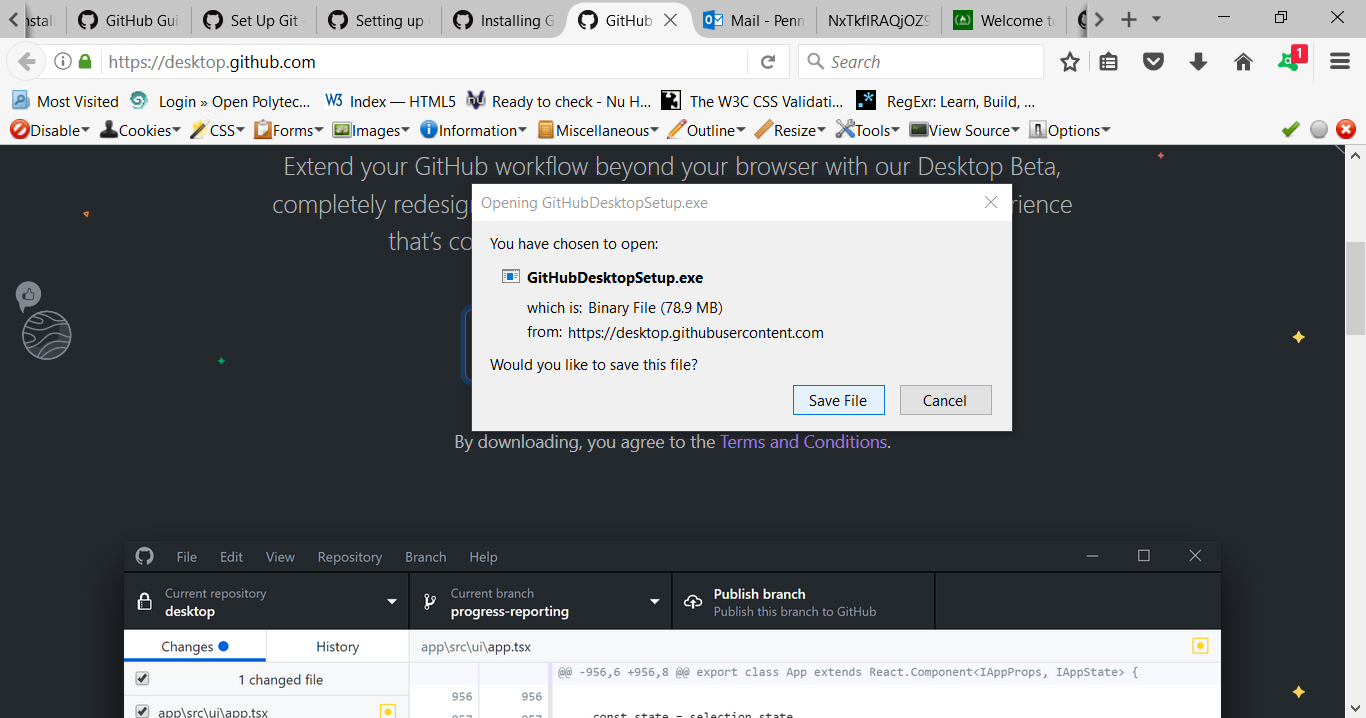


Figure 8. Downloading the client.



Figure 9. Installing the client.

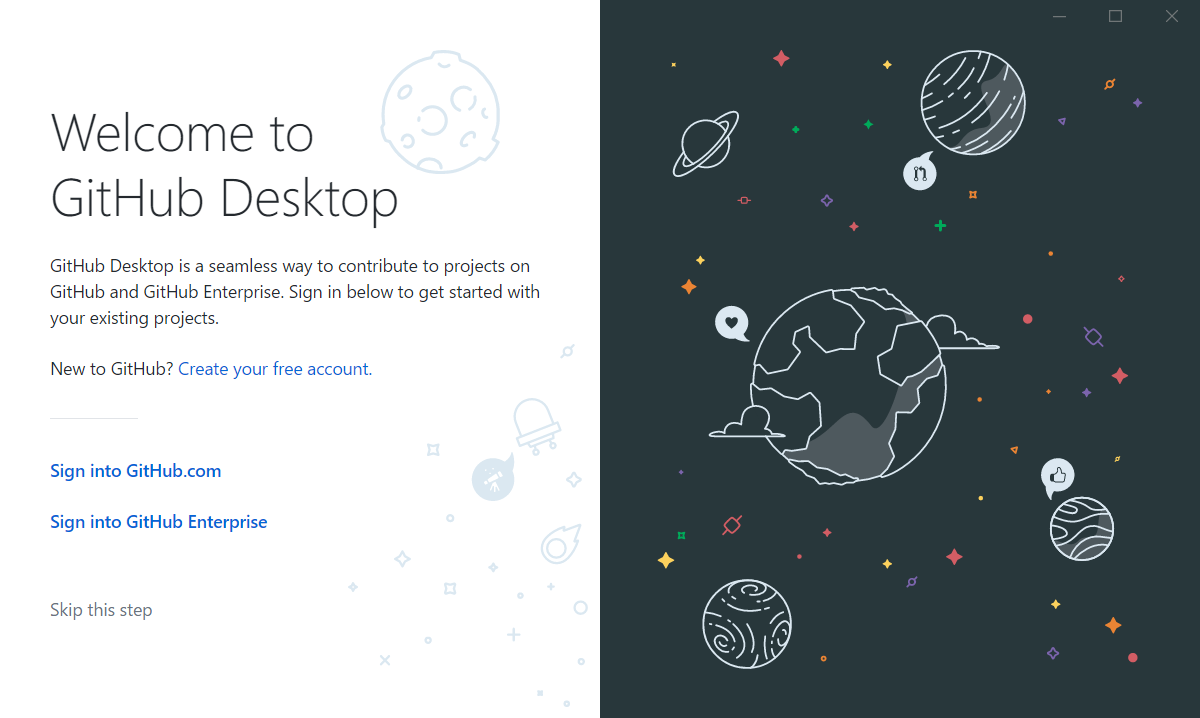


Figure 10. Sign into GitHub desktop.

**\**

Figure 10. Signing into GitHub to connect desktop client with my account.

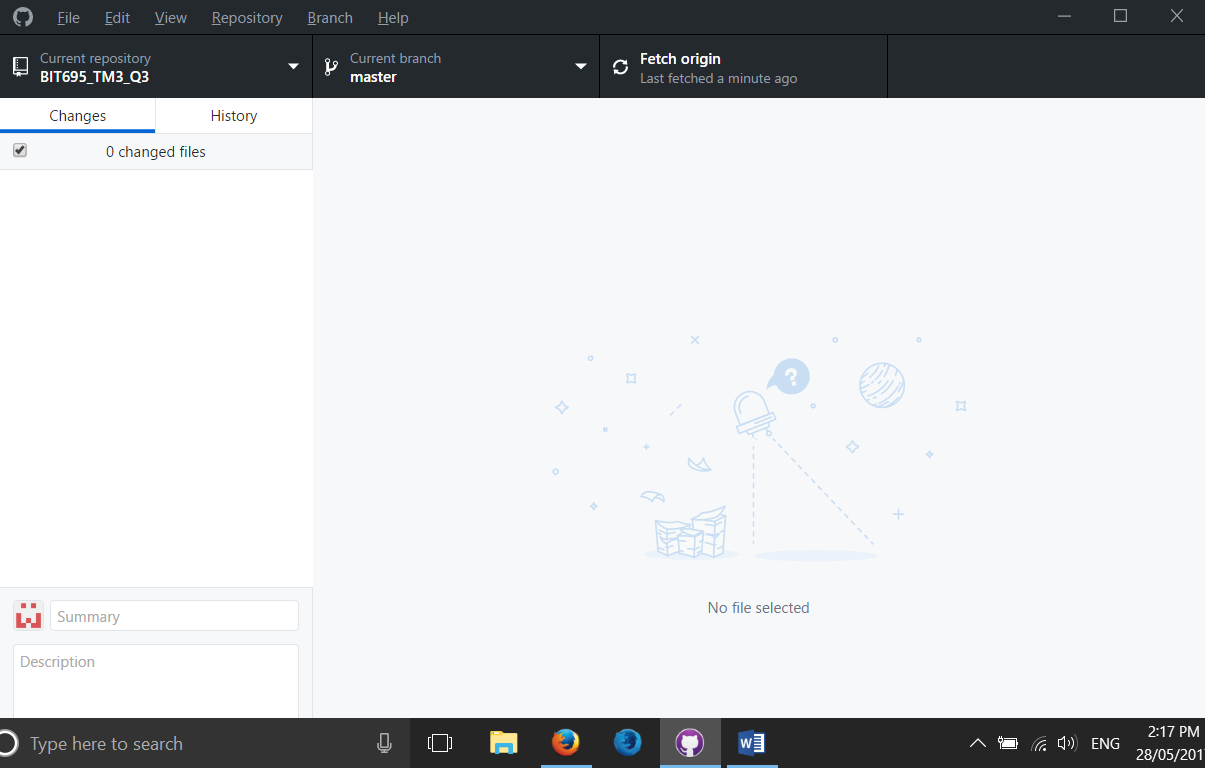


Figure 11. Showing my new repository.

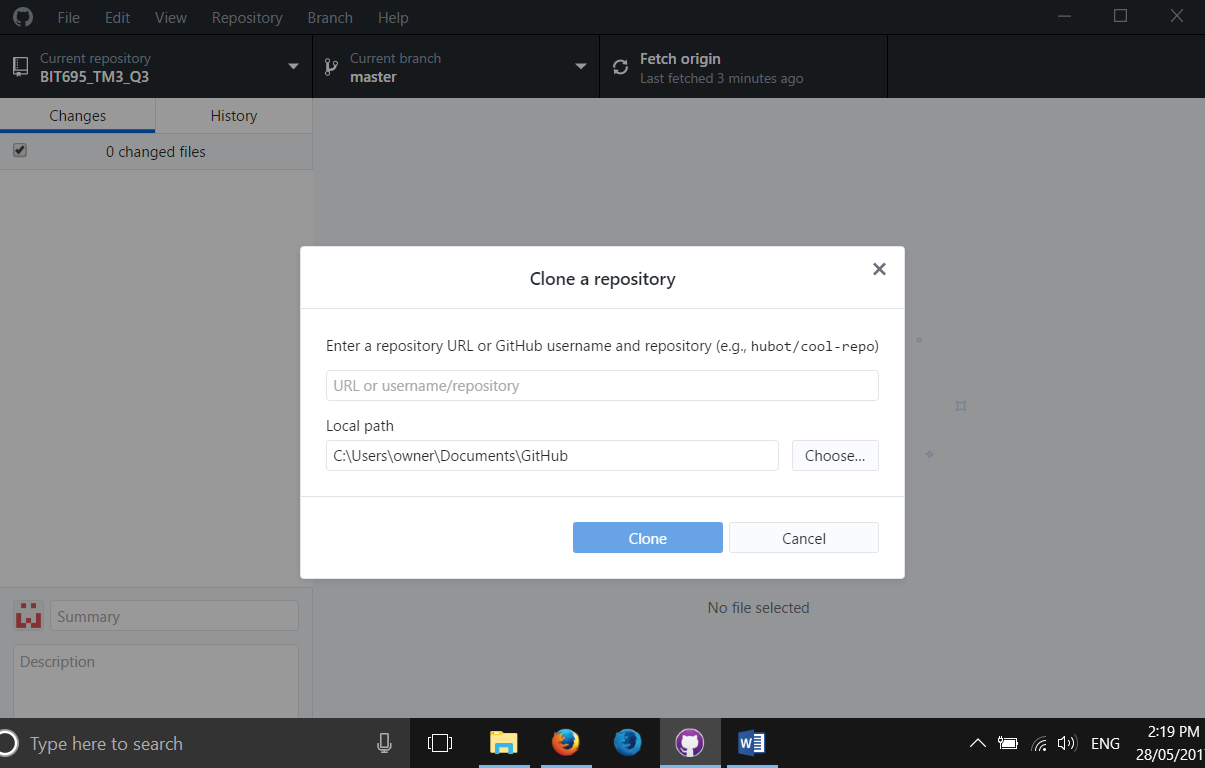


Figure 12. The clone dialogue.

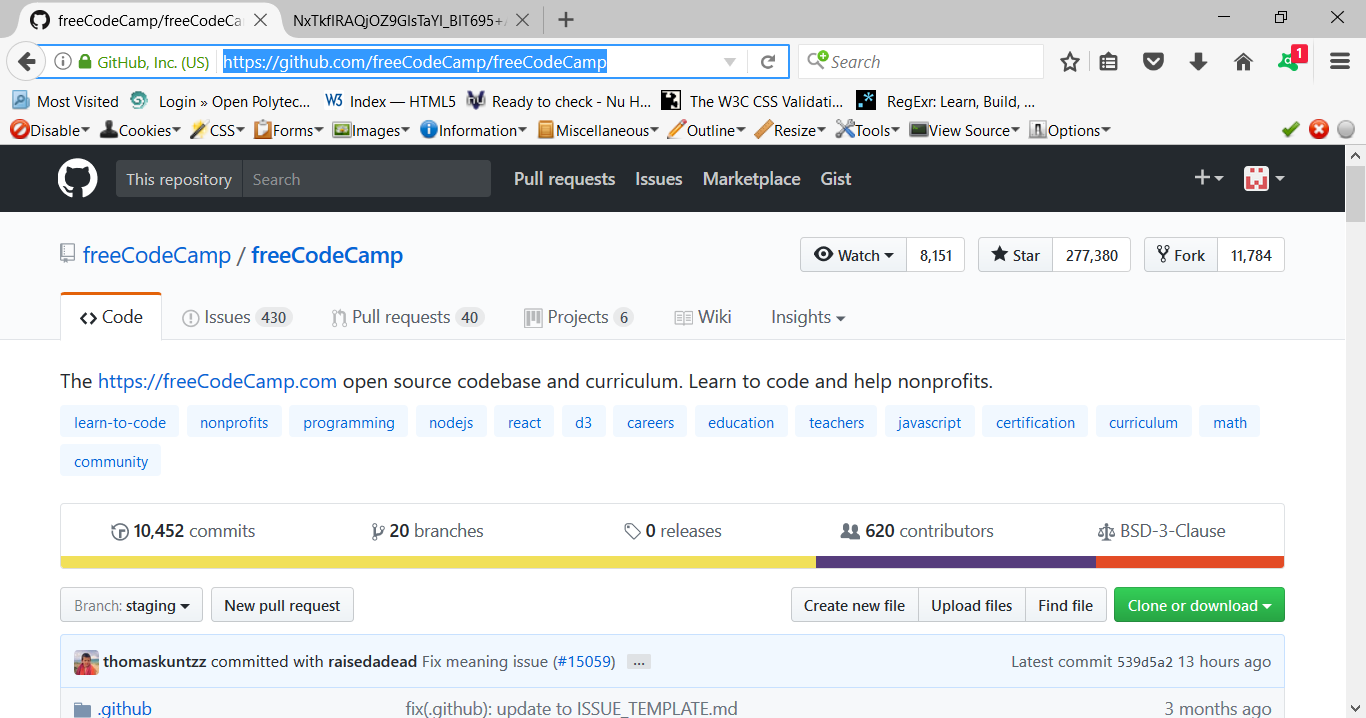


Figure 13. Finding the URL for the freeCodeCamp repository.

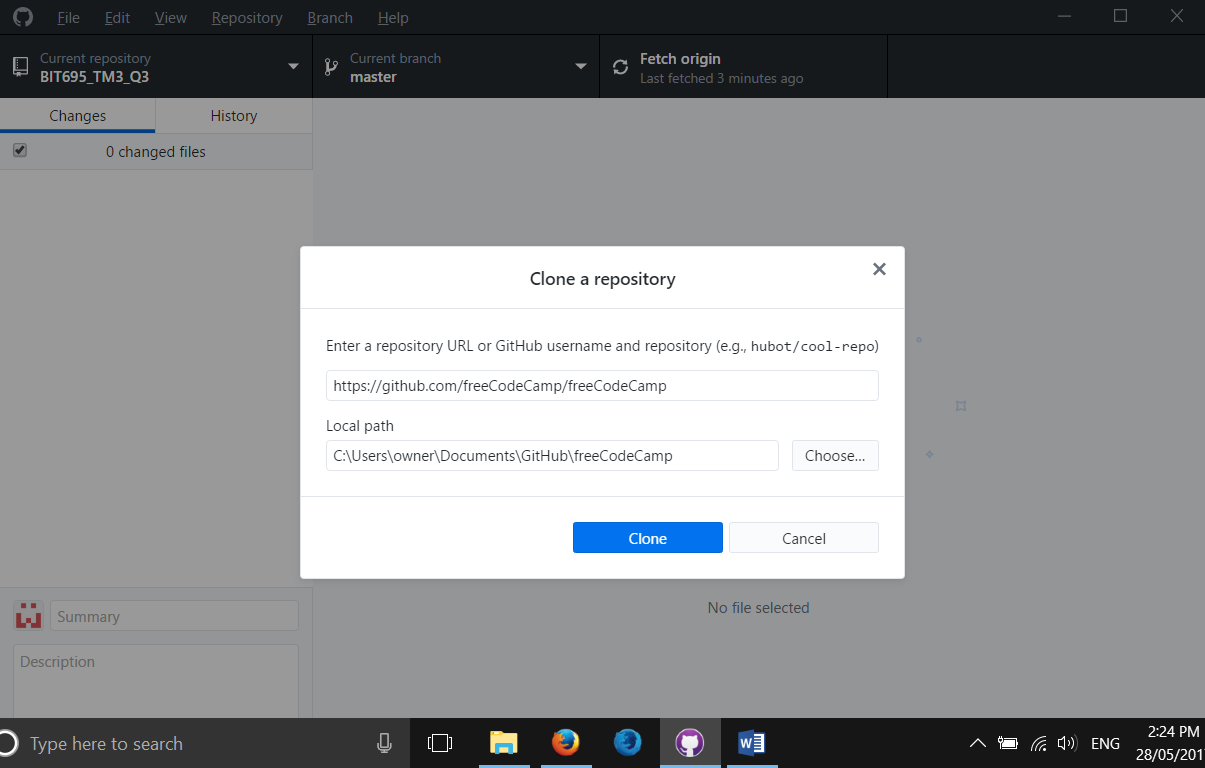


Figure 14. The URL and local path for the clone.

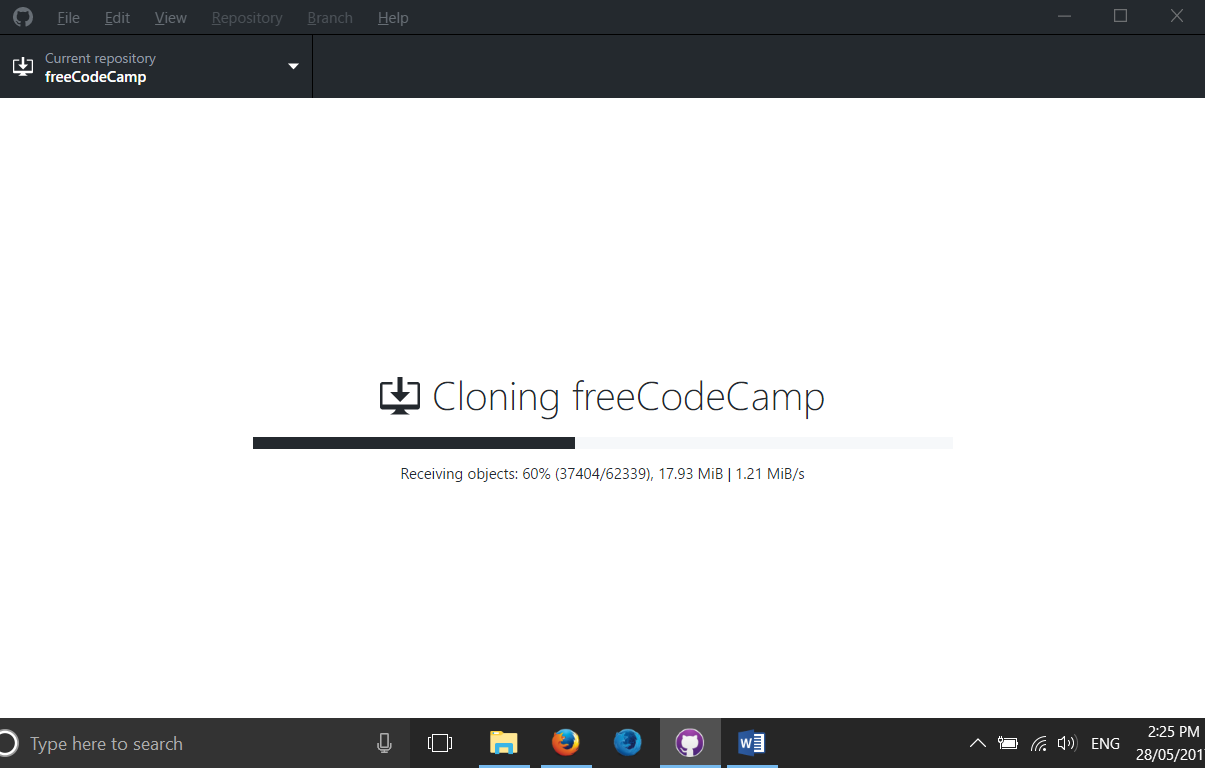


Figure 15. Cloning freeCodeCamp.

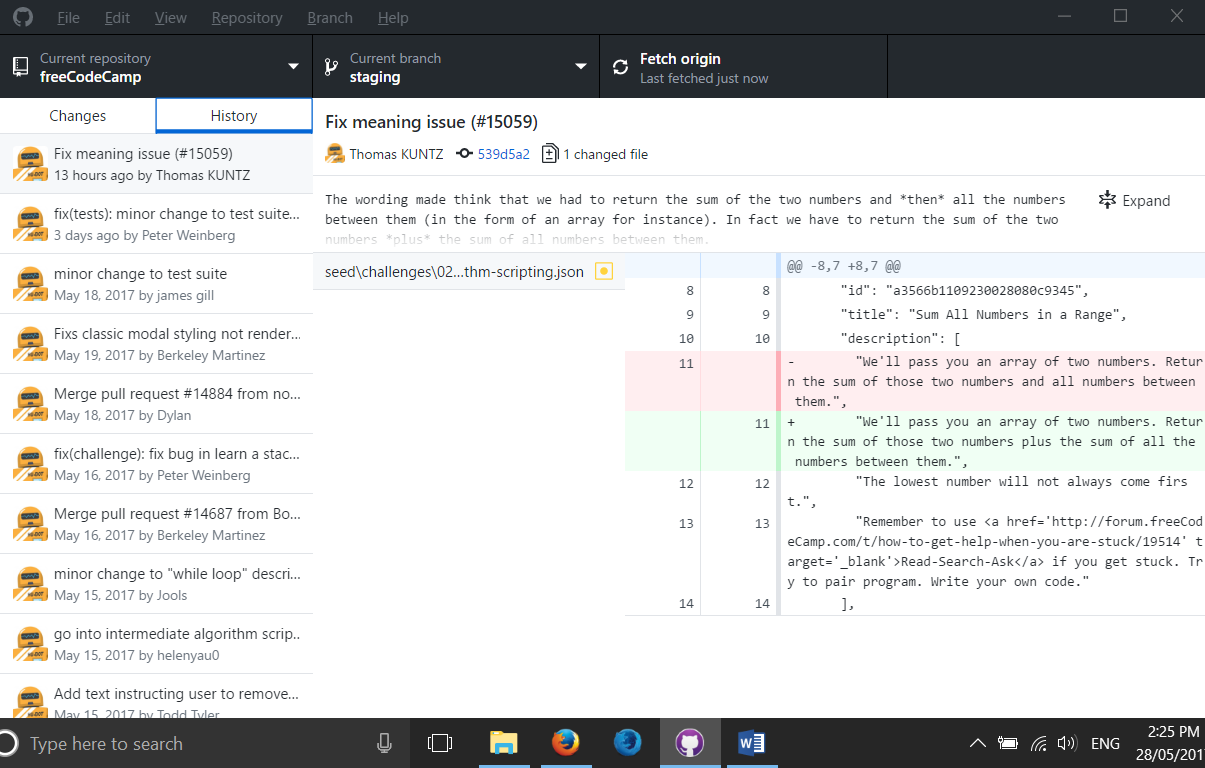


Figure 16. The cloned freeCodeCamp repository.

**c)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **GitHub** | **Bitbucket** | **Comments** |
| Private Repository | Paid only. Unlimited developers, but you pay by number of repositories. | Free for small teams (5 developers or less), unlimited private repositories.  Paid plans are by developer, with unlimited private repositories. | When you wish to keep your code “in-house” and private, you need a private repository. |
| Supports Project management tools | Also works with flow.ci and slack (for chat).  Has a wide range of tools available from the GitHub marketplace. | JIRA and HipChat (a chat tool) are integrated.  Also works with flow.ci | JIRA allows issue tracking and project management. |
| Wikis | Has a Wiki function for repositories.  Can be worked in online, or offline in a text editor. They are essentially a repository, so information is pulled and pushed. | Has an easy to use wiki feature for each repository. Can direct import markdown from code into it. They can be private or public. | Wikis are often used as a shared knowledge base for a team or group of developers.  They can also be used for code documentation. |
| Collaboration on changes | Can make changes on another fork, so long as you have push authority to the main repository. | Spooning, lets you work with someone else making changes on a fork. | Useful for team work, or working with a third party. |
| Pull Request | Allows you to do a pull request with multiple commits in it. Code can be reviewed from within this, and conflicts resolved, before doing a merge with branch | Sets up a forum so the code can be discussed, commented on and tweaked, with the changes tracked, before merging it back into the branch. | This allows for code to be reviewed before doing a commit, which avoids having to do a merge of two different copies of a file. |
| Code Review | Integrated tool set that shows changes, differences and history (either partial or full). There are also tools available from the marketplace. | Only really available on pull requests, unless you use a tool from the market place. | Being able to review code, and track changes and history can be useful for both work flow refinements, and for debugging and bug fixes. |
| Online editing | You can edit your code via the editor, from within your repository, you do need to do a pull request still. | You can edit from within the file on your repository and do a commit, without having to do a pull request. | This is good for fast, last minute fixes and tweaks. |
| Issue tracking | Has an integrated card system, for tracking issues. | JIRA integrated, and has bitbucket cards | Issue tracking is useful for managing a project, and workload/staffing resources. |
| Markdown support | Has integrated checklists. Also has tools on marketplace. | Has tools available on the marketplace.  Can also markup code in situ. | Allows a new developer to see what code has been removed or changed, without having to look at history. |
| Code fragments | Called Gists, can be shared | Called snippets, can be shared | Sometimes there re small pieces of code that are frequently reused. Having a place to store them, that is easily accessible speeds up work flow. |

(GitHub, 2017), (Atlassian, 2017)

## **Task 3: Application planning and development**

**a) Planning Activity.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Start date and time** | **Expected Completion date and time** | **Notes or comments** |
| Planning and risk assessment | 2pm 3/06/17 | 4pm 3/06/17 | Stop, pause review, then finalise |
| Database design | 4pm 3/06/17 | 8pm 3/06/17 | Double check requirements before finalising |
| Database coding and development | 10am 4/06/17 | 5pm 4/06/17  *Actually committed 6:40 3/06/17* | Stop, pause, reflect, check requirements. |
| Form, css and CRUD pages.  Testing while developing. | 9am 5/06/17  (Also need to finish TM3 BIT692 in this time frame). | By end of day, 10/06/17 | Test driven development, will use a harness if need be.  Will need to multi task with finishing Object Orientated Java TM3 |
| Final testing, bug fixes and style tweaks. | 10am 11/06/17 | By end of day 11/06/17 | Must be completed and in before 11:30pm, 11/06/17 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Event** | **Impact** | **Mitigation Steps** | **Severity (1-5)** |
| Developer illness/injury | Progress will halt, and time frames may need to be pushed out. | Use generous time frames, contact John Jameson and Caroline Eade if need be. | 5 |
| Loss of internet/power | Could impact on time frames or delivery. | Low likelihood.  Have alternate charging sources, work on laptop from elsewhere if need be. Can also use mobile data if needed. | 3 |
| Object Orientated Java eating into time. | Impact on delivery | Need to prioritise work and manage time effectively so that projects can be juggled. | 4 |
| Computer issues | Impact on development and delivery | Low likelihood.  Back up to cloud at end of each section, and end of each day.  Keep SQL code elsewhere, in case database need to be rebuilt (UniServer has already corrupted once). | 4 |
| Code development takes longer than expected | Delivery | Added in extra time for development.  Can apply for an extension, if needed. May need to look at minimizing mark loss. | 5 |

b) Modified from code used in BIT695 TM2. (Williamson, 2017)

/\* The code to create the new tables for the data base, assignment 2.

\*/

CREATE TABLE players(

MemberID int(30) NOT NULL AUTO\_INCREMENT,

FirstName varchar(25),

FamilyName varchar(40),

Email varchar(40),

Phone varchar(20),

PRIMARY KEY(MemberID),

);

/\*effectively leaderboard \*/

CREATE TABLE board\_games(

MemberID int(30) NOT NULL,

Boardgame varchar(60) NOT NULL,

Position varchar(10) NOT NULL, /\* varchar to allow = sign \*/

Notes text,

Date date NOT NULL,

Event varchar NOT NULL(125),

PRIMARY KEY (MemberID, Boardgame),

FOREIGN KEY(MemberID) REFERENCES players(MemberID),

);

CREATE TABLE library (

MemberID int(30) NOT NULL,

FirstName varchar(25) NOT NULL,

Boardgame varchar(60) NOT NULL,

Available varchar(20) NOT NULL,

Notes text NOT NULL,

PRIMARY KEY (MemberID),

CREATE INDEX idx\_library ON library(Boardgame),

);

CREATE TABLE library\_borrowers (

MemberID int(30) NOT NULL,

FirstName varchar(30) NOT NULL,

borrowed varchar(125) NOT NULL,

PRIMARY Key (MemberID),

FOREIGN Key (MemberID) REFERENCES library(MemberID),

CREATE INDEX idx\_borrowers ON library\_borrowers(FirstName),

);

CREATE TABLE schedule (

Day varchar(10) NOT NULL,

Boardgame varchar(60) NOT NULL,

Venue varchar(125) NOT NULL,

EventType varchar(60) NOT NULL,

Primary key (Day),

CREATE INDEX idx\_schedule ON schedule(Boardgame),

);

c) Code modified from BIT695 TM2. (Williamson, 2017), (The Open Polytechnic, 2017), (Refsnes Data, 1999-2017), (Refsnes Data, 1999-2017).

**Table Board\_games CRUD code and form.**

**Board\_gamesForm.html**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Board\_games</title>

</head>

<body>

<form id="Board\_games"

method="post"

enctype="application/x-www-form-urlencoded"

action="board\_gamesSubmit.php">

<fieldset>

<legend> Board\_games Position Form </legend>

<ul>

<li>

<label>Member ID: </label>

<input name = "memberID" type="int" id="memberID" tabindex="1" title="memberID"

placeholder="Please enter member ID" required pattern="[0-9]+">

<span class="error">\* <?php echo $memberIDErr;?></span>

</li>

<li>

<label>Boardgame: </label>

<input name = "boardgame" type="text" id="boardgame" tabindex="2" title="boardgame"

placeholder="Please enter boardgame" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $boardgameErr;?></span>

</li>

<li>

<label>Position: </label>

<input name = "position" type="text" id="position" tabindex="3" Title="position"

placeholder="Please enter position" required pattern="[0-9]">

<span class="error">\* <?php echo $positionErr;?></span>

</li>

<li>

<label>Notes: </label>

<input name = "notes" type="text" id="notes" tabindex="4" title="notes"

placeholder="Notes" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $notesErr;?></span>

</li>

<label>Date: </label>

<input name = "date" type="date" id="date" tabindex="5" title="date"

placeholder="date" required pattern="[0-9 / -]+">

<span class="error">\* <?php echo $dateErr;?></span>

</li>

<label>Event: </label>

<input name = "event" type="text" id="event" tabindex="6" title="event"

placeholder="event" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $eventErr;?></span>

</li>

</ul>

<input type = "submit" value = "submit">

</fieldset>

</form>

</body>

</html>

<!--- Required pattern is reg expressions. The span tag allows for the fields to be

marked as required by an astrix.

The php echo statement is to display the errors from server side validation. --->

**board\_gamesRet.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>board\_games position List </title>

</head>

<body>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

/\*The code to retrieve the data. Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host,$uid,$pwd,$database);

function selectBoard\_games($conn) {

$sql = "select \* from `board\_games`";

$result = $conn -> query($sql);

return $result;

}

$result = selectBoard\_games($conn);

if ($result->num\_rows > 0) {

echo "<table>";

while ($row = $result->fetch\_assoc()) {

echo '<tr><td>' . $row["MemberID"];

echo '</td><td>' . $row["Boardgame"];

echo '</td><td>' . $row["Position"];

echo '</td><td>' . $row["Notes"];

echo '</td><td>' . $row["Date"];

echo '</td><td>' . $row["Event"];

echo '</td><td><a href="deleteBoard\_games.php?memberID=' . $row["MemberID"] . '">delete Board\_game</a>';

echo '</td><td><a href="editBoard\_games.php?memberID=' . $row["MemberID"] . '">Edit Board\_game</a>';

echo '</td></tr>';

}

echo '</table>';

} else echo '0 results';

//update is found in the

//file named editplayer.php Contains the form for editing players.

//delete is found in the

//file named deleteplayer.php.

?>

</body>

</html>

**board\_gamesSubmit.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Form Submission</title>

<body>

Thank you for submitting a position.

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database) { $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

} /\*Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host, $uid, $pwd, $database);

$tempMemberID=$\_POST["memberID"];

$tempBoardgame=$\_POST["boardgame"];

$tempPosition=$\_POST["position"];

$tempNotes=$\_POST["notes"];

$tempDate=$\_POST["date"];

$tempEvent=$\_POST["event"];

function test\_input($data){

$data=trim($data);

$data=stripslashes($data);

$data=htmlspecialchars($data);

return $data;

}

/\*Define the variables, and set to empty values

before starting validation.

Code patterns from W3Schools;

PHP form Validation,

https://www.w3schools.com/php/php\_form\_validation.asp

PHP form URL/Email,

https://www.w3schools.com/php/php\_form\_url\_email.asp

and PHP form Required,

https://www.w3schools.com/php/php\_form\_required.asp

retreived: April 17 2017 \*/

$conn=connect\_db($host, $uid, $pwd, $database);

function validate\_code()

{

$memberIDErr="";

$boardgameErr="";

$positionErr="";

$notesErr="";

$dateErr="";

$eventErr="";

$memberID="";

$boardgame="";

$position="";

$notes="";

$date="";

$event="";

if($\_SERVER["REQUEST\_METHOD"]=="POST")

{

if(empty($\_POST["memberID"]))

{

$memberIDErr="member ID is required";

}

else

{

$memberID=test\_input($\_POST["memberID"]);

if(!preg\_match("/^[0-9]\*$/",$memberID))

{

$memberIDErr="Please use only numbers";

}

}

if(empty($\_POST["boardgame"]))

{

$boardgameErr="Boardgame is required";

}

else

{

$boardgame=test\_input($\_POST["boardgame"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$boardgame))

{

$boardgameErr="Please use only letters and numbers";

}

}

if(empty($\_POST["position"]))

{

$positionErr="Position is required";

}

else

{

$position=test\_input($\_POST["position"]);

if(!preg\_match("/^[0-9]\*$/",$position))

{

$positionErr="Please use numbers and equals sign";

}

}

if(empty($\_POST["notes"]))

{

$notesErr="Notes is required";

}

else

{

$notes=test\_input($\_POST["notes"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$notes))

{

$notesErr="Please enter notes";

}

}

if(empty($\_POST["date"]))

{

$dateErr="date is required";

}

else

{

$date=test\_input($\_POST["date"]);

if(!preg\_match("/^[0-9 - \/]\*$/",$date))

{

$dateErr="Please enter date";

}

}

if(empty($\_POST["event"]))

{

$eventErr="Event is required";

}

else

{

$event=test\_input($\_POST["event"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$event))

{

$eventErr="Please enter event";

}

}

}

$valid=false;

if($memberIDErr=="" && $boardgameErr=="" && $positionErr=="" && $notesErr=="" && $dateErr=="" && $eventErr=="")

{

$valid=true;

}

return $valid;

}

function insert\_form($conn, $tempMemberID, $tempBoardgame, $tempPosition, $tempNotes, $tempDate, $tempEvent){

$sql ="INSERT INTO `board\_games` (`MemberID`, `Boardgame`, `Position`, `Notes`, `Date`, `Event`)

VALUES ('$tempMemberID', '$tempBoardgame', '$tempPosition', '$tempNotes', '$tempDate', '$tempEvent')";

$result = $conn -> query($sql);

return $result;

}

$valid=validate\_code();

//Checks that the form has passed server side validation before inserting the data.

if($valid==true){

$result=insert\_form($conn, $tempMemberID, $tempBoardgame, $tempPosition, $tempNotes, $tempDate, $tempEvent);

}

?>

</body>

</html>

**editBoard\_games.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Edit board\_games </title>

</head>

<body>

</body>

</html>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host,$uid,$pwd,$database);

$tempMemberID= $\_GET["memberID"];

//variable declaration to get form data from the database.

$formMemberID=$\_POST["memberID"];

$formBoardgame=$\_POST["boardgame"];

$formPosition=$\_POST["position"];

$formNotes=$\_POST["notes"];

$formDate=$\_POST["date"];

$formEvent=$\_POST["event"];

if($formMemberID != '')

{

$tempMemberID=$formMemberID;

}

//function to get the data from the database to display on the form.

function get\_data($conn, $tempMemberID){

$sql="SELECT \* FROM `board\_games` WHERE `board\_games`.`MemberID`=$tempMemberID";

$result= $conn -> query($sql);

return $result;

}

function updateBoardgames($conn, $tempMemberID, $formBoardgame, $formPosition, $formNotes, $formDate, $formEvent) {

$sql = "UPDATE `board\_games` SET `Boardgame`='$formBoardgame',

`Position`='$formPosition', `Notes`='$formNotes',

`Date`='$formDate', `Event`='$formEvent' WHERE `board\_games`.`MemberID`=$tempMemberID" ;

$result = $conn -> query($sql);

return $result;

}

//The if statement to update the database. Checks if the Member ID field still

//holds the member ID, which is the primary key, first.

if($formMemberID != ''){

$result=updateBoardgames($conn, $formMemberID, $formBoardgame, $formPosition, $formNotes, $formDate, $formEvent);

}

$result=get\_data($conn, $tempMemberID);

?>

<form id="board-games"

method="post"

enctype="application/x-www-form-urlencoded"

action="editBoard\_games.php">

<fieldset>

<legend> Edit Board\_games </legend>

<ul>

<?php $row = $result->fetch\_assoc() ?> <!--fetchs the data for the form-->

<li>

<label>Membership ID</label>

<input type="int" tabindex="1" name="memberID" value="<?php echo $row["MemberID"];?>">

</li>

<li>

<label>Boardgame:<br> </label>

<input type="text" tabindex="2" name="boardgame" value="<?php echo $row["Boardgame"]; ?>">

</li>

<li>

<label>Position:<br> </label>

<input type="text" tabindex="3" name="position" value="<?php echo $row["Position"];?>">

</li>

<li>

<label>Notes:<br> </label>

<input type="text" tabindex="4" name="notes" value="<?php echo $row["Notes"];?>">

</li>

<li>

<label>Date:<br> </label>

<input type="text" tabindex="5" name="date" value="<?php echo $row["Date"];?>">

</li>

<li>

<label>Event:<br> </label>

<input type="text" tabindex="5" name="event" value="<?php echo $row["Event"];?>">

</li>

</ul>

<input type="submit" value="Update">

</fieldset>

</form>

</body>

</html>

**deleteBoard\_games.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Delete Board\_game Position </title>

</head>

<body>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host, $uid, $pwd, $database);

$tempMemberID= $\_GET["memberID"];

//Temporary varible for holding the value of the parameter.

//Variable needs to be passed in whenever the variable is called.

function deletePosition($conn, $tempMemberID) {

$sql = "DELETE FROM `board\_games` WHERE `board\_games`.`MemberID`=$tempMemberID";

$result = $conn -> query($sql);

return $result;

}

$result = deletePosition($conn, $tempMemberID);

?>

</body>

</html>

**Table library CRUD pages and forms.**

**libraryForm.html**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Games Library</title>

</head>

<body>

<form id="Library"

method="post"

enctype="application/x-www-form-urlencoded"

action="librarySubmit.php">

<fieldset>

<legend> Add a game </legend>

<ul>

<li>

<label>Member ID: </label>

<input name = "memberID" type="int" id="memberID" tabindex="1" title="memberID"

placeholder="Please enter member ID" required pattern="[0-9]+">

<span class="error">\* <?php echo $memberIDErr;?></span>

</li>

<li>

<label>First name: </label>

<input name = "firstName" type="text" id="firstName" tabindex="2" title="firstName"

placeholder="Please enter first name" required pattern="[a-z A-Z]+">

<span class="error">\* <?php echo $fnameErr;?></span>

</li>

<li>

<label>Boardgame: </label>

<input name = "boardgame" type="text" id="boardgame" tabindex="3" title="boardgame"

placeholder="Please enter name of boardgame" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $boardgameErr;?></span>

</li>

<li>

<label>Available to borrow?: </label>

<input name = "available" type="text" id="available" tabindex="4" Title="available"

placeholder="Is it available" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $availableErr;?></span>

</li>

<li>

<label>Notes: </label>

<input name = "notes" type="text" id="notes" tabindex="5" title="notes"

placeholder="Brief information about game" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $notesErr;?></span>

</li>

</ul>

<input type = "submit" value = "submit">

</fieldset>

</form>

</body>

</html>

<!--- Required pattern is reg expressions. The span tag allows for the fields to be

marked as required by an astrix.

The php echo statement is to display the errors from server side validation. --->

**libraryRet.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Retrieve library items </title>

</head>

<body>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

/\*The code to retrieve the data. Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host,$uid,$pwd,$database);

function selectLibraryItems($conn) {

$sql = "select \* from `library`";

$result = $conn -> query($sql);

return $result;

}

$result = selectLibraryItems($conn);

if ($result->num\_rows > 0) {

echo "<table>";

while ($row = $result->fetch\_assoc()) {

echo '<tr><td>' . $row["MemberID"];

echo '</td><td>' . $row["FirstName"];

echo '</td><td>' . $row["Boardgame"];

echo '</td><td>' . $row["Available"];

echo '</td><td>' . $row["Notes"];

echo '</td><td><a href="deleteGame.php?memberID=' . $row["MemberID"] . '">delete Game</a>';

echo '</td><td><a href="editGame.php?memberID=' . $row["MemberID"] . '">Edit Game</a>';

echo '</td></tr>';

}

echo '</table>';

} else echo '0 results';

//update is found in the

//file named editGame.php Contains the form for editing players.

//delete is found in the

//file named deleteGame.php.

?>

</body>

</html>

**librarySubmit.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Libary Form Submission</title>

<body>

Thank you for adding a game to the library.

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database) { $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

/\*Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host, $uid, $pwd, $database);

$tempMemberID=$\_POST["memberID"];

$tempFirstName=$\_POST["firstName"];

$tempBoardgame=$\_POST["boardgame"];

$tempAvailable=$\_POST["available"];

$tempNotes =$\_POST["notes"];

function test\_input($data){

$data=trim($data);

$data=stripslashes($data);

$data=htmlspecialchars($data);

return $data;

}

/\*Define the variables, and set to empty values

before starting validation.

Code patterns from W3Schools;

PHP form Validation,

https://www.w3schools.com/php/php\_form\_validation.asp

PHP form URL/Email,

https://www.w3schools.com/php/php\_form\_url\_email.asp

and PHP form Required,

https://www.w3schools.com/php/php\_form\_required.asp

retreived: April 17 2017 \*/

$conn=connect\_db($host, $uid, $pwd, $database);

function validate\_code()

{

$memberIDErr="";

$firstNameErr="";

$boardgameErr="";

$availableErr="";

$notesErr="";

$memberID="";

$firstName="";

$boardgame="";

$available="";

$notes="";

if($\_SERVER["REQUEST\_METHOD"]=="POST")

{

if(empty($\_POST["memberID"]))

{

$memberIDErr="Member ID is required";

}

else

{

$memberID=test\_input($\_POST["memberID"]);

if(!preg\_match("/^[0-9]\*$/",$memberID))

{

$memberIDErr="Please use only numbers";

}

}

if(empty($\_POST["firstName"]))

{

$memberIDErr="First name is required";

}

else

{

$firstName=test\_input($\_POST["firstName"]);

if(!preg\_match("/^[a-z A-Z]\*$/",$firstName))

{

$firstNameErr="Please use only letters";

}

}

if(empty($\_POST["boardgame"]))

{

$boardgameErr="Boardgame is required";

}

else

{

$boardgame=test\_input($\_POST["boardgame"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$boardgame))

{

$boardgameErr="Please use only letters, numbers and spaces";

}

}

if(empty($\_POST["available"]))

{

$availableErr="Availabilty is required";

}

else

{

$available=test\_input($\_POST["available"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$available))

{

$availableErr="Please use only letters, numbers and spaces";

}

}

if(empty($\_POST["notes"]))

{

$notesErr="Type of Event is required";

}

else

{

$notes=test\_input($\_POST["notes"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$notes))

{

$notesErr="Please enter event type";

}

}

}

$valid=false;

if($memberIDErr=="" && $firstNameErr=="" && $boardgameErr=="" && $availableErr=="" && $notesErr=="")

{

$valid=true;

}

return $valid;

}

function insert\_form($conn, $tempMemberID, $tempFirstName, $tempBoardgame, $tempAvailable, $tempNotes)

{

$sql ="INSERT INTO `library` (`MemberID`, `FirstName`, `Boardgame`, `Available`, `Notes`)

VALUES ('$tempMemberID', '$tempFirstName', '$tempBoardgame', '$tempAvailable', '$tempNotes')";

$result = $conn -> query($sql);

return $result;

}

$valid=validate\_code();

//Checks that the form has passed server side validation before inserting the data.

if($valid==true){

$result=insert\_form($conn, $tempMemberID, $tempFirstName, $tempBoardgame, $tempAvailable, $tempNotes);

}

?>

</body>

</html>

**editGame.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Edit Game </title>

</head>

<body>

</body>

</html>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host,$uid,$pwd,$database);

$tempMemberID= $\_GET["memberID"];

//variable declaration to get form data from the database.

$formMemberID=$\_POST["frmID"];

$formFname=$\_POST["frmFname"];

$formBGame=$\_POST["frmBGame"];

$formAvailable=$\_POST["frmAvailable"];

$formNotes=$\_POST["frmNotes"];

if($formMemberID != '')

{

$tempMemberID=$formMemberID;

}

//function to get the data from the database to display on the form.

function get\_data($conn, $tempMemberID){

$sql="SELECT \* FROM `library` WHERE `library`.`MemberID`=$tempMemberID";

$result= $conn -> query($sql);

return $result;

}

function updateGame($conn, $tempMemberID, $formFname, $formBGame, $formAvailable, $formNotes)

{

$sql = "UPDATE `library` SET `FirstName`='$formFname',

`Boardgame`='$formBGame', `Available`='$formAvailable',

`Notes`='$formNotes' WHERE `library`.`MemberID`=$tempMemberID" ;

$result = $conn -> query($sql);

return $result;

}

//The if statement to update the database. Checks if the Member ID field still

//holds the member ID, which is the primary key, first.

if($formMemberID != ''){

$result=updateGame($conn, $formMemberID, $formFname, $formBGame, $formAvailable, $formNotes);

}

$result=get\_data($conn, $tempMemberID);

?>

<form id="EditGame"

method="post"

enctype="application/x-www-form-urlencoded"

action="editGame.php">

<fieldset>

<legend> Edit Game</legend>

<ul>

<?php $row = $result->fetch\_assoc() ?> <!--fetchs the data for the form-->

<li>

<label>Membership ID</label>

<input type="int" tabindex="1" name="frmID" value="<?php echo $row["MemberID"];?>">

</li>

<li>

<label>First name:<br> </label>

<input type="text" tabindex="2" name="frmFname" value="<?php echo $row["FirstName"]; ?>">

</li>

<li>

<label>Boardgame:<br> </label>

<input type="text" tabindex="3" name="frmBGame" value="<?php echo $row["Boardgame"];?>">

</li>

<li>

<label>Available:<br> </label>

<input type="text" tabindex="4" name="frmAvailable" value="<?php echo $row["Available"];?>">

</li>

<li>

<label>Notes:<br> </label>

<input type="tel" tabindex="5" name="frmNotes" value="<?php echo $row["Notes"];?>">

</li>

</ul>

<input type="submit" value="Update">

</fieldset>

</form>

</body>

</html>

**deleteGame.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Edit Game </title>

</head>

<body>

</body>

</html>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host,$uid,$pwd,$database);

$tempMemberID= $\_GET["memberID"];

//variable declaration to get form data from the database.

$formMemberID=$\_POST["frmID"];

$formFname=$\_POST["frmFname"];

$formBGame=$\_POST["frmBGame"];

$formAvailable=$\_POST["frmAvailable"];

$formNotes=$\_POST["frmNotes"];

if($formMemberID != '')

{

$tempMemberID=$formMemberID;

}

//function to get the data from the database to display on the form.

function get\_data($conn, $tempMemberID){

$sql="SELECT \* FROM `library` WHERE `library`.`MemberID`=$tempMemberID";

$result= $conn -> query($sql);

return $result;

}

function updateGame($conn, $tempMemberID, $formFname, $formBGame, $formAvailable, $formNotes)

{

$sql = "UPDATE `library` SET `FirstName`='$formFname',

`Boardgame`='$formBGame', `Available`='$formAvailable',

`Notes`='$formNotes' WHERE `library`.`MemberID`=$tempMemberID" ;

$result = $conn -> query($sql);

return $result;

}

//The if statement to update the database. Checks if the Member ID field still

//holds the member ID, which is the primary key, first.

if($formMemberID != ''){

$result=updateGame($conn, $formMemberID, $formFname, $formBGame, $formAvailable, $formNotes);

}

$result=get\_data($conn, $tempMemberID);

?>

<form id="EditGame"

method="post"

enctype="application/x-www-form-urlencoded"

action="editGame.php">

<fieldset>

<legend> Edit Game</legend>

<ul>

<?php $row = $result->fetch\_assoc() ?> <!--fetchs the data for the form-->

<li>

<label>Membership ID</label>

<input type="int" tabindex="1" name="frmID" value="<?php echo $row["MemberID"];?>">

</li>

<li>

<label>First name:<br> </label>

<input type="text" tabindex="2" name="frmFname" value="<?php echo $row["FirstName"]; ?>">

</li>

<li>

<label>Boardgame:<br> </label>

<input type="text" tabindex="3" name="frmBGame" value="<?php echo $row["Boardgame"];?>">

</li>

<li>

<label>Available:<br> </label>

<input type="text" tabindex="4" name="frmAvailable" value="<?php echo $row["Available"];?>">

</li>

<li>

<label>Notes:<br> </label>

<input type="tel" tabindex="5" name="frmNotes" value="<?php echo $row["Notes"];?>">

</li>

</ul>

<input type="submit" value="Update">

</fieldset>

</form>

</body>

</html>

**Table borrowers CRUD pages and form.**

**borrowersForm.html**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Games Borrowers</title>

</head>

<body>

<form id="Borrowers"

method="post"

enctype="application/x-www-form-urlencoded"

action="BorrowersSubmit.php">

<fieldset>

<legend> Add a game </legend>

<ul>

<li>

<label>Member ID: </label>

<input name = "memberID" type="int" id="memberID" tabindex="1" title="memberID"

placeholder="Please enter member ID" required pattern="[0-9]+">

<span class="error">\* <?php echo $memberIDErr;?></span>

</li>

<li>

<label>First name: </label>

<input name = "firstName" type="text" id="firstName" tabindex="2" title="firstName"

placeholder="Please enter first name" required pattern="[a-z A-Z]+">

<span class="error">\* <?php echo $firstnameErr;?></span>

</li>

<li>

<label>Boardgame borrowed: </label>

<input name = "borrowed" type="text" id="borrowed" tabindex="3" title="borrowed"

placeholder="Please enter name of boardgame" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $boardgameErr;?></span>

</li>

<li>

</ul>

<input type = "submit" value = "submit">

</fieldset>

</form>

</body>

</html>

**borrowersRet.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Retrieve Borrowed items </title>

</head>

<body>

Borrowed Items list

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

/\*The code to retrieve the data. Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host,$uid,$pwd,$database);

function selectBorrowedItems($conn) {

$sql = "select \* from `library\_borrowers`";

$result = $conn -> query($sql);

return $result;

}

$result = selectBorrowedItems($conn);

if ($result->num\_rows > 0) {

echo "<table>";

while ($row = $result->fetch\_assoc())

{

echo '<tr><td>' . $row["MemberID"];

echo '</td><td>' . $row["FirstName"];

echo '</td><td>' . $row["Borrowed"];

echo '</td><td><a href="deleteBorrower.php?memberID=' . $row["MemberID"] . '">delete Borrower</a>';

echo '</td><td><a href="editBorrower.php?memberID=' . $row["MemberID"] . '">Edit Borrower</a>';

echo '</td></tr>';

}

echo '</table>';

} else echo '0 results';

//update is found in the

//file named editBorrower.php Contains the form for editing players.

//delete is found in the

//file named deleteBorrower.php.

?>

</body>

</html>

**BorrowersSubmit.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Retrieve Borrowed items </title>

</head>

<body>

Borrowed Items list

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

/\*The code to retrieve the data. Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host,$uid,$pwd,$database);

function selectBorrowedItems($conn) {

$sql = "select \* from `library\_borrowers`";

$result = $conn -> query($sql);

return $result;

}

$result = selectBorrowedItems($conn);

if ($result->num\_rows > 0) {

echo "<table>";

while ($row = $result->fetch\_assoc())

{

echo '<tr><td>' . $row["MemberID"];

echo '</td><td>' . $row["FirstName"];

echo '</td><td>' . $row["Borrowed"];

echo '</td><td><a href="deleteBorrower.php?memberID=' . $row["MemberID"] . '">delete Borrower</a>';

echo '</td><td><a href="editBorrower.php?memberID=' . $row["MemberID"] . '">Edit Borrower</a>';

echo '</td></tr>';

}

echo '</table>';

} else echo '0 results';

//update is found in the

//file named editBorrower.php Contains the form for editing players.

//delete is found in the

//file named deleteBorrower.php.

?>

</body>

</html>

**editBorrower.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Edit Borrower </title>

</head>

<body>

</body>

</html>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host,$uid,$pwd,$database);

$tempMemberID= $\_GET["memberID"];

//variable declaration to get form data from the database.

$formMemberID=$\_POST["memberID"];

$formFName=$\_POST["firstName"];

$formBorrowed=$\_POST["borrowed"];

if($formMemberID != '')

{

$tempMemberID=$formMemberID;

}

//function to get the data from the database to display on the form.

function get\_data($conn, $tempMemberID){

$sql="SELECT \* FROM `library\_borrowers` WHERE `library\_borrowers`.`MemberID`=$tempMemberID";

$result= $conn -> query($sql);

return $result;

}

function updateGame($conn, $tempMemberID, $formFName, $formBorrowed)

{

$sql = "UPDATE `library\_borrowers` SET `FirstName`='$formFName',

`Borrowed`='$formBorrowed' WHERE `library\_borrowers`.`MemberID`=$tempMemberID";

$result = $conn -> query($sql);

return $result;

}

//The if statement to update the database. Checks if the Member ID field still

//holds the member ID, which is the primary key, first.

if($formMemberID != ''){

$result=updateGame($conn, $formMemberID, $formFName, $formBorrowed);

}

$result=get\_data($conn, $tempMemberID);

?>

<form id="Membership"

method="post"

enctype="application/x-www-form-urlencoded"

action="editBorrower.php">

<fieldset>

<legend> Edit Borrower </legend>

<ul>

<?php $row = $result->fetch\_assoc() ?> <!--fetchs the data for the form-->

<li>

<label>Membership ID</label>

<input type="int" tabindex="1" name="memberID" value="<?php echo $row["MemberID"];?>">

</li>

<li>

<label>First name:<br> </label>

<input type="text" tabindex="2" name="firstName" value="<?php echo $row["FirstName"]; ?>">

</li>

<li>

<label>Boardgame borrowed:<br> </label>

<input type="text" tabindex="3" name="borrowed" value="<?php echo $row["Borrowed"];?>">

</li>

<li>

</ul>

<input type="submit" value="Update">

</fieldset>

</form>

</body>

</html>

**deleteBorrower.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Delete Borrower </title>

</head>

<body>

Borrower record deleted.

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host, $uid, $pwd, $database);

$tempMemberID= $\_GET["memberID"];

//Temporary varible for holding the value of the parameter.

//Variable needs to be passed in whenever the variable is called.

function deleteBorrower($conn, $tempMemberID) {

$sql = "DELETE FROM `library\_borrowers` WHERE `library\_borrowers`.`MemberID`=$tempMemberID";

$result = $conn -> query($sql);

return $result;

}

$result = deleteBorrower($conn, $tempMemberID);

?>

</body>

</html>

**Table schedule CRUD pages and form.**

**schedule\_form.html**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Events</title>

</head>

<body>

<form id="Schedule"

method="post"

enctype="application/x-www-form-urlencoded"

action="schedule\_submit.php">

<fieldset>

<legend> Schedule of Events </legend>

<ul>

<li>

<label>Date: </label>

<input name = "date" type="date" id="date" tabindex="1" title="date"

placeholder="Please use date format" required pattern="[0-9 / -]+">

<span class="error">\* <?php echo $dateErr;?></span>

</li>

<li>

<label>Boardgame: </label>

<input name = "boardgame" type="text" id="boardgame" tabindex="2" title="boardgame"

placeholder="Please enter boardgame" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $boardgameErr;?></span>

</li>

<li>

<label>Venue: </label>

<input name = "venue" type="text" id="venue" tabindex="3" Title="venue"

placeholder="Please enter venue" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $venueErr;?></span>

</li>

<li>

<label>Type of Event: </label>

<input name = "eventType" type="text" id="eventType" tabindex="4" title="eventType"

placeholder="ie. Competitive, family friendly" required pattern="[a-z A-Z 0-9]+">

<span class="error">\* <?php echo $eventErr;?></span>

</li>

</ul>

<input type = "submit" value = "submit">

</fieldset>

</form>

</body>

</html>

<!--- Required pattern is reg expressions. The span tag allows for the fields to be

marked as required by an astrix.

The php echo statement is to display the errors from server side validation. --->

**Schedule\_ret.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Retrieve Schedule </title>

</head>

<body>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

/\*The code to retrieve the data. Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host,$uid,$pwd,$database);

function select\_schedule($conn) {

$sql = "select \* from `schedule`";

$result = $conn -> query($sql);

return $result;

}

$result = select\_schedule($conn);

if ($result->num\_rows > 0) {

echo "<table>";

while ($row = $result->fetch\_assoc()) {

echo '<tr><td>' . $row["Day"];

echo '</td><td>' . $row["Boardgame"];

echo '</td><td>' . $row["Venue"];

echo '</td><td>' . $row["EventType"];

echo '</td><td><a href="deleteEvent.php?day=' . $row["Day"] . '">delete event</a>';

echo '</td><td><a href="editEvent.php?day=' . $row["Day"] . '">Edit event</a>';

echo '</td></tr>';

}

echo '</table>';

} else echo '0 results';

//update is found in the

//file named editEvent.php Contains the form for editing events.

//delete is found in the

//file named deleteEvent.php.

?>

</body>

</html>

**schedule\_submit.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Form Submission</title>

<body>

Thank you for submitting an event.

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database) { $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

/\*Modified from code supplied

in BIT695 notes, The Open Polytechnic Companion Guide,

Block 2, Part 5:The server side(2), section 5 Querying a database.

Retrieved 10 April 2017.

Also used in BIT695\_TM2\_3431274 Question 1.

\*/

$conn=connect\_db($host, $uid, $pwd, $database);

$tempdate=$\_POST["date"];

$tempboardgame=$\_POST["boardgame"];

$tempvenue=$\_POST["venue"];

$tempeventType=$\_POST["eventType"];

function test\_input($data){

$data=trim($data);

$data=stripslashes($data);

$data=htmlspecialchars($data);

return $data;

}

/\*Define the variables, and set to empty values

before starting validation.

Code patterns from W3Schools;

PHP form Validation,

https://www.w3schools.com/php/php\_form\_validation.asp

PHP form URL/Email,

https://www.w3schools.com/php/php\_form\_url\_email.asp

and PHP form Required,

https://www.w3schools.com/php/php\_form\_required.asp

retreived: April 17 2017 \*/

$conn=connect\_db($host, $uid, $pwd, $database);

function validate\_code()

{

$dateErr="";

$boardgameErr="";

$venueErr="";

$eventTypeErr="";

$date="";

$boardgame="";

$venue="";

$eventType="";

if($\_SERVER["REQUEST\_METHOD"]=="POST")

{

if(empty($\_POST["date"]))

{

$dateErr="date is required";

}

else

{

$date=test\_input($\_POST["date"]);

if(!preg\_match("/^[0-9 \/ -]\*$/",$date))

{

$dateErr="Please use date format";

}

}

if(empty($\_POST["boardgame"]))

{

$boardgameErr="Boardgame is required";

}

else

{

$boardgame=test\_input($\_POST["boardgame"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$boardgame))

{

$boardgameErr="Please use only letters, numbers and spaces";

}

}

if(empty($\_POST["venue"]))

{

$venueErr="Venue address is required";

}

else

{

$venue=test\_input($\_POST["venue"]);

if(!preg\_match("/^[a-z A-Z 0-9 ']\*$/",$venue))

{

$venueErr="Please use only letters, numbers and spaces";

}

}

if(empty($\_POST["eventType"]))

{

$eventTypeErr="Type of Event is required";

}

else

{

$eventType=test\_input($\_POST["eventType"]);

if(!preg\_match("/^[a-z A-Z 0-9]\*$/",$eventType))

{

$eventTypeErr="Please enter event type";

}

}

}

$valid=false;

if($dateErr=="" && $boardgameErr=="" && $venueErr=="" && $eventTypeErr=="")

{

$valid=true;

}

return $valid;

}

function insert\_form($conn, $tempdate, $tempboardgame, $tempvenue, $tempeventType){

$sql ="INSERT INTO `schedule` (`Day`, `Boardgame`, `Venue`, `EventType`)

VALUES ('$tempdate', '$tempboardgame', '$tempvenue', '$tempeventType')";

$result = $conn -> query($sql);

return $result;

}

$valid=validate\_code();

//Checks that the form has passed server side validation before inserting the data.

if($valid==true){

$result=insert\_form($conn, $tempdate, $tempboardgame, $tempvenue, $tempeventType);

}

?>

</body>

</html>

**editEvent.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Edit Event </title>

</head>

<body>

Update event using this form.

</body>

</html>

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host,$uid,$pwd,$database);

$tempDay= $\_GET["day"];

//variable declaration to get form data from the database.

$formDay=$\_POST["day"];

$formBoardgame=$\_POST["boardgame"];

$formVenue=$\_POST["venue"];

$formEventType=$\_POST["eventType"];

if($formDay!= '')

{

$tempDay=$formDay;

}

//function to get the data from the database to display on the form.

function get\_data($conn, $tempDay){

$sql="SELECT \* FROM `schedule` WHERE `schedule`.`Day`='$tempDay'";

$result= $conn -> query($sql);

return $result;

}

function updateEvent($conn, $tempDay, $formBoardgame, $formVenue, $formEventType) {

$sql = "UPDATE `schedule` SET `Boardgame`='$formBoardgame',

`Venue`='$formVenue', `EventType`='$formEventType' WHERE `schedule`.`Day`='$tempDay'";

$result = $conn -> query($sql);

return $result;

}

//The if statement to update the database. Checks if the Day field still

//holds the date, which is the primary key, first.

if($formDay != ''){

$result=updateEvent($conn, $formDay, $formBoardgame, $formVenue, $formEventType);

}

$result=get\_data($conn, $tempDay);

?>

<form id="Schedule"

method="post"

enctype="application/x-www-form-urlencoded"

action="editEvent.php">

<fieldset>

<legend> Edit Event </legend>

<ul>

<?php $row = $result->fetch\_assoc() ?> <!--fetchs the data for the form-->

<li>

<label>Date:</label>

<input type="date" tabindex="1" name="day" value="<?php echo $row["Day"];?>">

</li>

<li>

<label>Boardgame:<br> </label>

<input type="text" tabindex="2" name="boardgame" value="<?php echo $row["Boardgame"];?>">

</li>

<li>

<label>Venue:<br> </label>

<input type="text" tabindex="3" name="venue" value="<?php echo $row["Venue"];?>">

</li>

<li>

<label>Event Type:<br> </label>

<input type="text" tabindex="4" name="eventType" value="<?php echo $row["EventType"];?>">

</li>

</ul>

<input type="submit" value="Update">

</fieldset>

</form>

</body>

</html>

**deleteEvent.php**

<!DOCTYPE html>

<html lang=en-GB>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link rel="StyleSheet" href="styles.css" type="text/css" media="screen" />

<title>Delete Event </title>

</head>

<body>

Event deleted.

<?php

$uid="root";

$pwd="root";

$database="assignment2";

$host = 'localhost';

function connect\_db($host, $uid, $pwd, $database)

{ $conn=mysqli\_connect($host, $uid, $pwd, $database)

or die('connection problem:' . mysqli\_connect\_error());

return $conn;

}

$conn=connect\_db($host, $uid, $pwd, $database);

$tempDay= $\_GET["day"];

//Temporary varible for holding the value of the parameter.

//Variable needs to be passed in whenever the variable is called.

function deleteEvent($conn, $tempDay) {

$sql = "DELETE FROM `schedule` WHERE `schedule`.`Day`=$tempDay";

$result = $conn -> query($sql);

return $result;

}

$result = deleteEvent($conn, $tempDay);

?>

</body>

</html>

# **References**

Ali Hodroj, E. C. (2015). *The 3-Second Rule of eCommerce: How retailers can scale for extreme holiday season traffic*. Retrieved June 1, 2017, from The In-Memory Computing Blog by GigaSpaces: https://blog.gigaspaces.com/the-3-second-rule-of-ecommerce-high-availability-scalability-inmemory/

Atlassian. (2017). *Homeage*. Retrieved June 02, 2017, from Atlassian Bitbucket: https://bitbucket.org/

Charlton, G. (2010, March 09). *The importance of uptime in e-commerce: survey*. Retrieved June 01, 2017, from Econsultancy: https://econsultancy.com/blog/5536-the-importance-of-uptime/

GitHub. (2017). *About*. Retrieved June 02, 2017, from GitHub: https://github.com/about

Henderson, A. (2017, March 25). *The CIA Triad: Confidentiality, Integrity, Availability*. Retrieved June 1, 2107, from The Panmore Institute: http://panmore.com/the-cia-triad-confidentiality-integrity-availability

Margaret Rouse, M. H. (1999 - 2017). *confidentiality, integrity, and availability (CIA triad)*. Retrieved June 01, 2017, from techTarget WhatIs.com: http://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA

Refsnes Data. (1999-2017). *PHP 5 Form Validation*. Retrieved from w3schools.com: https://www.w3schools.com/php/php\_form\_validation.asp

Refsnes Data. (1999-2017). *PHP 5 Forms - Required Fields*. Retrieved from w3schools.com: https://www.w3schools.com/php/php\_form\_required.asp

The Open Polytechnic. (2017). *Block 2, Part 5, Section 5: Querying databases*. Retrieved from BIT955: https://openpolytechnic.iqualify.com/course/-Kbra1QsqYA1eXXPjNoU/#/page/p262

The OWASP Foundation. (2014). Testing Guide 4.0 release.

The OWASP Foundation. (2017). OWASP Top 10 - 2017 rcl.

Williamson, P. A. (2017). *BIT695\_TM2\_3431274 Question 1.*