

Web Tech Report

Design and implementation of an interactive Kickstarter website

Module Name	Web Technologies
Module Code	ACCA7021
Lecturer	Gaynor Thomas
Student	Zhao Ning (1904564)

Table of Contents

Part 1. Front-end Programming	2
1.1 CSS Related	2
1.2 JS Related	2
Part 2. Backend Programming	3
2.1 Connection to the Database	3
2.2 Uploading Files to the Server	4
2.3 PHP Form Handling	4
Part 3. Web Security	4
3.1 Users and Permissions	4
3.2 Sessions Control	4
3.3 MySQL Prepared Statements	5
Part 4. A Brief Walkthrough of the Website	6
4.1 Normalized Database	6
4.2 The User Pages	7
4.2.1: "explore.php"	7
4.2.2: "mylist.php"	7
4.2.3: "create.php"	7
4.2.4: "edit.php"	7
4.3 The Admin Page	7
4.3.1: "jsgrid.php"	7
4.4 Shared Features	7

4.4.1: Navigation Bar - “navigation.php”	7
4.4.2: Paging - “res_paging.php”	7
4.5 Sign In / Sign Up Pages.....	7
4.5.1: Sign In - “signin.php”	7
4.5.2: Sign Up - “signup.php”	8
Part 5. Conclusion.....	8
5.1 Bugs Encountered	8
5.2 Works to Be Done	8
Appendix. References	9
• Tools and resources:	9

Part 1. Front-end Programming

1.1 CSS Related

The Sass language along with the CSS preprocessor that generates readable and easier-to-maintain CSS from its unique syntax:

- *Mix-ins*
- *Inheritance selectors*
- *Nesting selectors*

We apply this technology by first compiling Sass files into CSS files. We can also watch the source file so that it gets recompiled each time changes are saved:

```
sass --style compressed --watch scss/index/index.scss css/index.css
```

```
sass --style compressed --watch scss/signin/signin.scss css/signin.css
```

1.2 JS Related

The UglifyJS JavaScript optimizer is utilized to minify JS files (so as to reduce the amount of data transferred between server-side devices and client-side devices) and generate source maps:

```
uglifyjs js/signin.js -o js/signin.min.js -c --source-map "url='signin.min.js.map',  
filename='signin.min.js'"
```

1.3 Responsive Design

1.3.1: APPLY MEDIA QUERIES

Media queries enable us to create a responsive experience where specific styles are applied to small screens (of mobile phones), large screens(of PCs), and anywhere in between(of tablets). The media

query syntax allows for the creation of rules that can be applied depending on device characteristics¹.

1.3.2: THIRD-PARTY FRAMEWORKS AND LIBRARIES

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS². It enables developers to prototype or to build the entire app with Sass variables and mix-ins, responsive grid system, extensive prebuilt components such as date-pickers, customizable buttons and form inputs, and powerful plugins built on jQuery. Bootstrap 4 brings with it several highlights:

- a migration to Sass;
- the move to CSS's flexbox;
- **References:**
 - <https://sass-lang.com/guide>
 - <https://www.npmjs.com/package/uglify-js>
 - <https://developers.google.com/web/tools/chrome-devtools/inspect-styles/edit-styles>
 - <https://medium.com/@toolmantim/getting-started-with-css-sourcemaps-and-in-browser-sass-editing-b4daab987fb0>

Part 2. Backend Programming

2.1 Connecting to the Database

The following command opens a new connection to the MySQL server:

```
$link = mysqli_connect($strServer,$strUser,$strPwd,$strDatabase);
```

which is similar to:

```
$mysqli = new mysqli("localhost","my_user","my_password","my_db");
```

Besides, the CRUD (Create, Read, Update, Delete) functionality and its corresponding most-used commands with MySQL are also included:

- **CREATE DATABASE** IF NOT EXISTS `db` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
- **DROP DATABASE** IF EXISTS `db`;
- **USE** `db`;
- **CREATE TABLE** table (
 Col1 INT PRIMARY KEY AUTO_INCREMENT,
 Col2 VARCHAR(255),

¹ <https://developers.google.com/web/fundamentals/design-and-ux/responsive>

² <https://getbootstrap.com/>

Col3 INT

-);
- **INSERT INTO** `table` (`name`, `age`) VALUES ('Jane', 22);

2.2 Uploading Files to the Server

“**FormData**” provides the method **append(name, value)** and can be sent to the server using **XMLHttpRequest.send()** or the cross browser **ajax** framework.

2.3 Handling PHP Forms

- **\$_GET**: an array of variables passed to the current script through the visible URL parameters.
- **\$_POST**: an array of variables passed to the current script through the invisible HTTP POST method.

The **\$_GET** approach is used here for information filtering such as custom search functions, paging functions.

Part 3. Web Security

3.1 Users and Permissions

MySQL gives us various options to grant specific users permissions within databases and tables.

1. Create a new user named “testadmin”, with its password being “testadmin”:

```
CREATE USER 'testadmin'@'localhost' IDENTIFIED BY 'testadmin'
```

2. Provide the user “testadmin” with all privileges to the table “users”:
GRANT ALL PRIVILEGES ON Kickstarter.users TO 'testadmin'@'localhost';
3. Revoke the permission “UPDATE” from the user “testadmin” on the table “users”
REVOKE UPDATE ON Kickstarter.users FROM 'testadmin'@'localhost';
4. Verify the validity of the aforementioned operations.

3.2 Sessions Control

A server session manages the server side of client/server communications, providing shared resources, including a shared object cache and connection pools to a single data source.

A client session will be stored by the browser. There will be one session id for each user connected to the server.

Here we will use both these sessions for maintaining specific user states including persistent objects (like result sets returned from database queries) and authenticated user identities, among many other interactions³.

Code snippets:

```
If (!isset($_SESSION('name'))) {  
    echo "<script>  
        alert('Please sign in first.');        location.href = \"signin.php\";  
    </script>\";  
}
```

3.3 MySQL Prepared Statements

Compared to executing SQL statements directly, prepared statements have three main advantages⁴:

- They reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times);
- Their bound parameters minimize bandwidth required for data transmission as only the parameters instead of the whole queries are transferred;
- They are very useful against SQL injections, in that they avoid the potential risks of parameter values that are not correctly escaped by transmitting them later using a different protocol.

Code snippets:

```
$stmt = mysqli_stmt_init($link);  
  
$query = "insert into projects (name,goal,...) values (?,?,...)\";  
  
$mysqli_stmt_prepare($stmt, $query);  
  
$mysqli_stmt_bind_param(($stmt, «si...», $name, $goal, ...);  
  
$mysqli_stmt_execute($stmt);
```

³ <https://docs.oracle.com/cd/E19683-01/817-2172-10/dwsessn.html>

⁴ https://www.w3schools.com/php/php_mysql_prepared_statements.asp

Part 4. A Brief Walkthrough of the Website

4.1 Normalized Database

Three of the most commonly used normal forms are as follows:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)

First normal form (1NF)

- As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Second normal form (2NF)

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

Third normal form (3NF)

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute. In other words, any database that conforms to 3NF satisfies: a table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$ at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

There are in total 6 tables of the Kickstarter database that are involved in this website:

- 1) **“projects”**:
 - a. for a brief project information display;
 - b. includes such fields as “id” “name” “currency” “goal” “pledge” etc.;
- 2) **“projects_more”**:
 - a. for a detailed project information display;
 - b. includes such fields as “blurb” “county” “launched_date” etc.;
- 3) **“currencies” “categories” “cities” etc.**:
 - a. the product of database normalization;
 - b. includes such fields as “id” “currency_name” “category_name” etc.;
- 4) **“users”**:
 - a. for authentication and authorization;
 - b. includes such fields as “id” “name” “password” “active” etc.;

4.2 The User Pages

4.2.1: “EXPLORE.PHP”

This page displays all projects included in the Kickstarter database, including ongoing(live) ones, successful ones, and failed ones. A progress bar is attached to every project block. Fuzzy search criteria can be applied in the search bar.

4.2.2: “MYLIST.PHP”

This page displays all projects that belong to a specific signed-in user. Fuzzy search criteria can be applied in the search bar.

4.2.3: “CREATE.PHP”

This page allows a specific signed-in user to create a new project by filling out the project information form.

4.2.4: “EDIT.PHP”

This page allows a specific signed-in user to edit one of his or her existing projects by filling out the project information form, including uploading a new or replacing the old with a new project promotion image.

4.3 The Admin Page

4.3.1: “JSGRID.PHP”

This page allows only the super administrator to manage general users' accounts in a table-style UI, including activating currently disabled users' accounts, or disabling the currently active ones.

4.4 Shared Features

4.4.1: NAVIGATION BAR - “NAVIGATION.PHP”

This section is included in many pages as a common navigation feature, which encompasses the Kickstarter index page, the personal projects list page, sign in or sign out page, the categories drop-down list, and the search bar.

4.4.2: PAGING - “RES_PAGING.PHP”

This section is included in many pages that need dividing displayed results into multiple pages of results. Common page indicators like first, last, previous and next are present in the paging element when appropriate.

4.5 Sign In / Sign Up Pages

4.5.1: SIGN IN - “SIGNIN.PHP”

This page allows newly registered users log in to the website in order to access their own project records and to make a modification.

4.5.2: SIGN UP - “SIGNUP.PHP”

This page allows newcomers to the system to acquire their unique account (a finalized username and password combination) both for authentication and authorization purposes, and for access to subscription-only features (such as to start a new Kickstarter project).

Part 5. Conclusion

5.1 Bugs Encountered

- The carousel displayed in the **explore.php** page has not been implemented yet;
- The project creation (*insert* queries) isn't working properly;

5.2 Works to Be Done

- A more advanced search approach, incorporating words such as **OR**, **NOT**, and **AND**; (in uppercase letters) as search operators. These words can be used to tell the search function whether to broaden or to narrow the search range;
- Web scalability using containers; (containers are lightweight virtualization layers that can be started and terminated much quicker than VMs. They also do not consume as much resources;⁵)
 - The Nginx web application server have several features worth mentioning, including custom code modules, calling out to CGI scripts and load-balancing that can really boost the performance of a medium-sized to large-sized website.
 - Besides, The Node.js programming language and PHP differ in their approach to concurrency, with Node.js using a non-blocking event loop (running in a single process) and the standard PHP runtime using a blocking process⁶. Web servers like Apache and NGINX have ways to improve the PHP multi-tasking performance to some extent, but these are more limited than true language support for multi-tasking.
- Code Refactoring;

There are several rules of thumb for guiding developers through the refactoring process⁷:

- New functionality shouldn't be created during refactoring.
- Don't mix refactoring and direct development of new features. Try to separate these processes at least within the confines of individual commits.
- All existing tests must pass after refactoring.

There are two cases when application tests can be broken down after refactoring:

- An error was made while refactoring, which simply needs a general response;

⁵ <https://towardsdatascience.com/scalable-web-development-d57a46a7f349>

⁶ <https://www.hostingadvice.com/blog/comparing-node-js-vs-php-performance/>

⁷ <https://refactoring.guru/refactoring/how-to>

- Your tests were too low-level. For example, you were testing private methods of classes. Either we can refactor the tests themselves or write an entirely new set of higher-level tests. To write BDD-style tests would be a decent way to deal with this kind of situation.
- Transactions and Locking (To implement the project contribution or crowdfunding process (similar to the processing of shopping carts and the generation of orders, database transactions are often needed to ensure the ACID (atomicity, consistency, isolation, durability) properties to guarantee validity even in the event of errors, power failures⁸)
- Accessibility Evaluation; (Websites and web tools that are well designed and coded will not be difficult for people with disabilities to use them;)
- Making the web accessible benefits individuals, businesses, and society. International web standards define what is needed for accessibility.)

Appendix. References

[1] Luke Welling, Laura Thomson, PHP and MySQL Web Development, 4th Edition.

○ Tools and resources:

- 1) OS: macOS 10.13.2
- 2) IDE: IntelliJ IDEA 2019.1 (Ultimate Edition)
- 3) Database Server: MySQL 8.0.18;
- 4) MySQL Management: PhpMyAdmin 4.9.1;
- 5) Web Server: Apache/2.4.28 (Unix) PHP/7.1.7
- 6) Browser: Google Chrome 78.0.3904.108; Safari 11.0.2 (13604.4.7.1.6)

⁸ <https://en.wikipedia.org/wiki/ACID>