Project Title: Learning Path Creator

Completion date: 3/12/2023

Table of Contents

Contents

Introduction

Our learning path creator web application, called Learning Voyage provides a platform for anyone who wishes to spread accessible knowledge to the masses. It allows users to explore our curated pathways, discover new skills, and join a community passionate about education!

Setup and Installation

To setup and run the the application, one must have a server capable of interpreting PHP, either locally or on a separate domain. It is also required to have a connection to a MySQL server in order to store and manage data in Databases.
To deploy the application on another location, simply copy the entire contents of the project folder and paste them wherever you wish to run the application. Keep in mind, it is absolutely vital that **all the paths within the project folder remain unchained**! After doing this, setup your MySQL database with all the correct tables and then change the connection credentials within the app/core/Model.php so that the application is able to communicate with your database. After this the application will be ready to use!

Application Architecture

The application uses an Object-Oriented MVC design pattern to function. It contains a single index page into which all other content is rendered. The details of the page to be rendered are sent as actions into the GET request which are then used to call a function within the controller object. This function sets up the appropriate model for that action, queries the required data, passes it into a newly created view object, and then renders the view for the user to see. A consistent Header and Footer is visible across all pages so that the user is able to navigate through that application with ease. This application uses Bootstrap for front-end design and styling.

User Authentication and Authorization

User Authentication is done through the login page. The login page loads the Users model and has access to data of all registered users and it compares the user input to that information to check if the email address and password are valid or not. A user is not able to log into the system with invalid credentials. If the credentials are valid, the system creates a session with variables to tell all other pages within the system that a user is logged in and who the user is.

Some of the existing users within the system are:
Email: aaadibadola@gmail.com Password: aadibadola1
Email: iffat.nabila@gmail.com Password: iffat123
Email: johndoe@gmail.com Password: pass123

Database Interaction

The abstract Model class contains all necessary information to create a mysqli connections object and establish interaction with the database. This connection is inherited to all other model classes which contain the majority of the functions to query data.

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	category	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	category_has_topic	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	learning_path	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	resources	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	topic	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	topic_has_paths	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	users	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	32.0 KiB	-
	7 tables	Sum	44	InnoDB	utf8mb4_general_ci	192.0 KiB	0 B

Database Queries

All the queries are written within the model files. The queries used within the application are:
In Model.php:

- SELECT * FROM category;
- DELETE FROM topic_has_paths WHERE path_id = ?;
- DELETE FROM resources WHERE path_id = ?;
- DELETE FROM learning_path WHERE path_id = ?;
- SELECT * FROM users WHERE email = ?;

In Paths.php

- SELECT lp.*, u.name as author_name FROM learning_path lp JOIN users u ON lp.author_id = u.user_id
  WHERE lp.path_id in (SELECT lp.path_id
  FROM category c
  JOIN category_has_topic cht ON c.id = cht.category_id
  JOIN topic t ON cht.topic_id = t.id
  JOIN topic_has_paths thp ON t.id = thp.topic_id
  JOIN learning_path lp on thp.path_id = lp.path_id
  WHERE c.id = $categoryID);
- SELECT lp.*, u.name as author_name, u.email as author_email FROM learning_path lp JOIN users u ON lp.author_id = u.user_id WHERE path_id = $pathID;
- SELECT lp.*, u.name as author_name, u.email as author_email FROM learning_path lp JOIN users u ON lp.author_id = u.user_id
  WHERE lp.author_id = $authorID;
- SELECT lp.*, u.name as author_name FROM learning_path lp JOIN users u ON lp.author_id = u.user_id;
- SELECT * FROM resources WHERE path_id = $pathID;
- SELECT * FROM users;
- SELECT t.id,t.name FROM category c
  JOIN category_has_topic cht ON c.id = cht.category_id
  JOIN topic t ON cht.topic_id = t.id
  WHERE c.id = $categoryID;
- INSERT INTO learning_path (`title`, `author_id`, `last_updated`)
  VALUES (?, ?, ?);
- INSERT INTO topic_has_paths
  VALUES (?, $pathID);
- INSERT INTO resources (`step_num`, `path_id`, `name`, `link`, `type`, `description`)
  VALUES (?, ?, ?, ?, ?, $description);
- UPDATE resources SET name = ?, link = ?, type = ?, description = $description
  WHERE step_num = ? AND path_id = ?;
- INSERT INTO learning_path (`title`, `author_id`, `last_updated`, `reference`)
  SELECT CONCAT_WS(' ', title, '(clone) '), ?, CURDATE(), path_id FROM learning_path lp2
  WHERE lp2.path_id = ?;
- INSERT INTO topic_has_paths
  SELECT topic_id, ? FROM topic_has_paths
  WHERE path_id = ?;
- INSERT INTO resources
  SELECT `step_num`, ?, `name`, `link`, `type`, `description` FROM resources
  WHERE path_id = ?;

In Users.php

- INSERT INTO `users`(`name`, `email`, `password`) VALUES (?, ?, ?);
- SELECT `password` FROM `users` WHERE `email` = ?;
- SELECT lp.*, u.name as author_name FROM learning_path lp
  JOIN users u ON lp.author_id = u.user_id
  WHERE author_id =
  (SELECT user_id FROM users WHERE email = ?);
- UPDATE users SET name='$name', bio=$bio WHERE user_id=$userID;

Server-side scripting (PHP)

As mentioned previously, the application uses an OOMVC structure to function. PHP is used to access data from databases, populate pages dynamically and even for input validation. A lot of PHP functionalities were used to make this project possible. The include feature of PHP allowed for a consistent UI to be displayed across all pages. The isset() function was used in numerous functions to only execute a functionality based on the presence of a REQUEST variable. The spl_autoload functions were extremely useful as they allowed for creation of objects without the need to explicitly include the class files every time. PHP blocks in the middle of HTML pages were used to populate tags of similar structure but different content. Accessing GET and POST variables and using them for interactions with the database was used throughout the model files.

Responsive User Interface

A responsive UI was achieved by using the Bootstrap Library for CSS and JS. It streamlined the design process and resulted in a sleek application layout. CSS files were also meticulously created to enhance the beauty of the application.

Error Handling and Validation

Error Handling and Validation within the application is done using PHP. If statements are used to account for exceptional cases, especially when interacting with the database and all input fields which are not meant to be NULL or empty are checked for that condition. Email address validation checks if the entered email address contains an '@' symbol or not. Validation and Error Handling is done while uploading an image in the Edit Profile page to ensure that only a JPG image is uploaded to the server. Functions interacting with the database return responses which provide an appropriate output to the user in case of success or failure. In case a user manually enters an action which does not correspond to a functionality, the user is directed to a 404 error page.

Documentation and Code Quality

The workload for this project was divided in such a way that there would be no conflicts or anomalies within the application. Dividing the application in terms of Back-End, Front-End and Server-Side scripting allowed all members to follow consistent code regulations and nomenclature in their respective sections thus ensuring high code quality.
All code is organized in a systematic manner and properly indented to enhance readability.

Project-Specific Functionality (Innovative Feature)

Our application adds an additional functionality of Categories/Courses and Topics. We believe this improves the experience of a user coming into the application for the first time as they are met with a compact and concise home page through which they can easily navigate to find paths that pertain to any of their interests. In the case that a user visits this website without having a particular learning path in mind, this method of organization of all paths would allow the user to easily find a few paths that peak their interests.

Additional Notes

Notes to help use the learning path application:

- Selecting a dropdown from the Courses menu item allows you to view all paths for that course.
- Selecting the Path menu item will display the list of all learning paths
- Only a registered user is allowed to create or clone learning paths.
- Once logged in, users can create a new learning path by going to their profile.
- The creator of the path, if logged in is allowed to make additions to the path and edit existing path content.

Team Member Information and Contributions

Provide contact information for all group members, including names, student IDs, email addresses, URL to presentation video and URL project folder on member's GBLearn account.

All URLs must open in a new window.

Full Name: Aadi Badola Student ID: 101412150 GBC email: aadi.badola@georgebrown.ca	
Video URL	https://youtu.be/ghMCKXLuL2Q?si=JeLXqXcRME2EcFgM
GBLearn URL	https://f3412150.gblearn.com
Tasks completed by member	Responsible for Server Side scripting
	Designed PHP application based on MVC model
	Responsible for integrating back-end with front-end

Full Name: Iffat Amin Nabila Student ID: 101429832 GBC email: iffatamin.nabila@georgebrown.ca	
Video URL	https://youtu.be/2IEsJaX-_u4?si=PbYeDVqtQfQEag_E
GBLearn URL	https://f3429832.gblearn.com
Tasks completed by member	Responsible for creating all front-end views used to display data
	Responsible for designing the User Interface and guiding User Experience
	Created responsive layouts for all pages.

Full Name: Fernando Chavez Solares Student ID: 101423471 GBC email: Fernando.ChavezSolares@georgebrown.ca	
Video URL	
GBLearn URL	https://f3423471.gblearn.com
Tasks completed by member	Responsible for creation of schema and relations between tables
	Responsible for supplying appropriate SQL queries to be used in PHP