* Database:
  + This part’s database is named bookmarks and contain two tables to help store user authentication information and their links.
  + The *users* table is formed of two columns, a *user\_name*, wich is also the primary key, and a *user\_pswrd,* allowing the user to verify their identity.
  + Given that the username is a primary key, it is included in the SignIn script to request for usernames to be unique.
  + Both columns are NOT NULL, this they both must the filled in order to create a new user.
  + The links table is composed of 5 diferent columns of which only two will be of the user’s concern, the other three are for backend processing.
  + *link\_id* is a numeric INT value, automatically incremented and set by the database. This number is primary key and allow us to identify every individual link, since diferent users could store the same URL and even under the same name.
  + *link\_adr* is the url address of the website to bookmark.
  + *link\_name* refers to the name the link is given by the user, this for easier browsing on the user’s side.
  + *link\_click* helps count how many times a link was clicked, latter on ussed to explain popularity
  + *user\_name* is a foreign key to the user table, ensuring every links instance is assigned to an user.
* SingIn & LogIn/Out:
  + For the login, I ussed as mix of Ajax, so that the user could send in new links and navigate without reloading, and php, to fetch the credentials.
  + The user credentials have been hashed with a simple hash, as to protect the user’s identity ([PHP: password\_hash - Manual](https://www.php.net/manual/en/function.password-hash.php)).
  + The user credentials serve to open a session, that will remain open as long as the user is on the website and will be closed once they leave, as a safety measure.
  + When the user Signs in, their password is automatically hashed (password\_hash($plaintextPassword, PASSWORD\_BCRYPT); ) and can the only be deciphered via password\_verify($password, $row['user\_pswrd']) during the log in.
* Adding a bookmark:
  + Simple SQL query passed through php allows to create a new bookmark
  + The bookmark entity is composed of a numeric primary key for storage purposes, a name, a username (foreign key to the user that created the link) and a click counter.
  + Both the link counter and the id are automatically generated, the username is fetched from the session in course.
  + The click counter, link\_click, serves to account the number of times a link was clicked by the user.
  + It is via the link counter that we use a SQL query to search the links with the most gross visits across all users.
* Browsing bookmarks:
  + To browse the bookmarks the user is expected to lookup the name they gave to the bookmark.
  + This, because it is more likely for the user to remember the name, they use than the link path.
  + The SQL query only looks through the links belonging to that specific user.
  + If no search is being executed, then all links are displayed. This, thanks to AJAX processing, to avoid constant re-loading. ([PHP - AJAX and PHP](https://www.w3schools.com/php/php_ajax_php.asp))
* Editing/Deleting:
  + As requested, the user can edit or delete a bookmark of their choice.
  + Both buttons related to these processes appear besides each bookmark, linking the action to the data to be altered.
  + The delete button simply deletes the link entity from the database, thus making it disappear form the users list.
  + A window prompt does appear before deletion, asking the user if they are certain of their decision. Upon approval, the bookmark is deleted.
  + For editing, we call upon a modal, allowing the user to edit either the link or the name assigned to the bookmark.
  + At any point the user may click on the x to close the window, and no changes will be made.
  + If the user wants to precede with the changes, they may press the save changes button, which will then call the php script (SQL query) to update the link instance in the link table of the database.
  + Let it be noted that all this process uses the link\_id as guiding thread for all the SQL queries, as they are unique to each link\_id and automatically generated by the system. ([MySQL PRIMARY KEY Constraint](https://www.w3schools.com/mysql/mysql_primarykey.asp))
* The page is divided in two main sections, the form to stored bookmarks and the browser.
* An additional home page (hero section) is added to greet the user and display the top 10 links that, as previously stated, are calculated based on total click across all the entities containing the same link.
* A navigation bar helps jump from one section to another and additionally allows to log out.
* The website cannot be accessed without first signing in.
* Please ensure proper database acces in in connection.php
* Database:
  + *Lessons* is the database for the homework’s second part. It is composed of three tables, serving to authenticate, store content and keep track of made progress.
  + The *user* table is identical to that of part 1, so you may refer to the database section of the documentation, under users.
  + The *xml\_storage* table, is composed of 3 columns and serves to store the quiz and lesson EMLs.
  + *id*  column assigns every xml file an autogenerate unique id that serves as primary key and will be used all through part 2 to identify the lesson and quiz EMLs.
  + *filename* is the name of the file uploaded, there is not much need for precision but in real world circumstances it is important to properly identify documents with significative names.
  + *xml\_content* contains the EMLs themselves, stored as LONGTEXT in the database to be called later on by the application.
  + The *progress\_record* table is made up of 4 diferent columns, are used for backend logic support.
  + *record\_id* is the entities primary key, automatically generated when a new instance is created.
  + *record\_status* serves to keep the users progress status for each lesson. When a user reads a lesson a new table instance is created, with a default status of “reading”. This changes to completed when the unit is finished (tested).
  + *user\_name* serves as foreign key to the user whose progress we are cording. All instances of *progress\_record* must have an assigned user.
* *lesson* foreign key to the unit for which we are recording progress. All instances of progress must have a lesson linked to them.
* Educational Markup Language (EML):
* Lessons:
* • <eml\_U1> - Root element containing the entire document.
* • <head> - Holds metadata-like elements:
* <title> - Main title of the document.
* <subtitle> - Describes the document’s content.
* • <lesson> - Represents different lessons:
* <title> - Lesson title.
* <paragraph> - Textual content explaining concepts.
* <list> - Contains lists of items.
* <list-paragraph> - Introductory text for a list.
* <item> - Individual list items.
* • <example-block> - Holds example sections.
* <example> - Represents an individual example.
* <title> - Title of the example.
* <paragraph> - Explanation of the example.
* <list> - Contains structured information (like lesson lists).
* <img> - References an image related to the example.
* Quizes:
* **<question>**: Represents an individual quiz question.
* **<text>**: Contains the question's text.
* **<choices>**: Groups the possible answer choices for a question.
* **<wrong>**: Represents an incorrect answer choice.
* **<correct>**: Represents the correct answer choice.
* Page functionality and documentation:
* Authentication:
  + This part employs the same authentication system as part 1. The user table in the database is private to the lessons database, but the logic is the same.
  + Please refer to the SingIn & LogIn/Out section of part 1.
* Parsers:
  + The php EML parser for the lessons and quizzes is divided in two files for each respective EML, thus generating cleaner code as you won’t have to unnecessarily call the code to lessons when your are only using the quiz ([PHP: xml\_parse - Manual](https://www.php.net/manual/en/function.xml-parse.php)).
  + Each parser takes their specific EML as parameter, there will be errors if the wrong type EML is submitted, as it won’t be able to find the tags to parse.
  + Lessons parser:
  + After setting the title, the php code iterates through the elements of the EML ussing foreach to address repeated elements. [PHP: Basic SimpleXML usage - Manual](https://www.php.net/manual/en/simplexml.examples-basic.php)
  + The way Examples and Lessons are addressed will have slight differences, as the examples include images and a slightly distinct formatting but, the core elements remain the same.
  + List elements and paragraphs elements will display similarly under both englobing tags.
  + Quiz parser:
  + All questions are all displayed the same, in a fieldset,but the information about the wrong/correct answers are maintained by assigning each new element a class attribute of the same name, as to allow corrections.
  + Other than that, it is straightforward looping through the questions, text and choices elements.
* Lesson grading:
  + A JS script (grading.js) oversees grading and outputting results, once the user submits their form.
  + The script counts the number of questions while also keeping record of the questions whose selected radio button is of class “correct”.
  + For each fieldset, if the selected answer was wrong, a “red” class attribute is added to it, if the choice was correct, it’s class becomes “green”.
  + At the end of the script, we calculate the success percentage and display the amount of correct choices counted over the total.
* Tracking logic:
  + With the *progress\_record* table, the website can keep record of the user’s progression to the course. (Grades are not kept since the point is self learning).
  + The *progress\_record* is for users to know where they were at in the website rather than to keep track of their activities, ergo why we display the information on the homepage.
  + To do this we use traking.php, the script iterates through *progress\_record* and identifies if any progress has been made (lessons have been opened/tested)
  + If there is material, either on “reading” or “tested” the script will print it for the user to see.
  + If nothing is found, the script suggests starting from Unit 1
  + To keep the records, two short php scripts have been implemented in unit.php and done.php.
  + In unit.php, the script checks whether there is any entity in the table containing the same *user\_id* & *lesson* pair. If there isn’t, it creates one, and this, every time a unit is reviewed for the first time.
  + In quiz.php the done script is only triggered if the user chooses to get their quiz graded. This gives the user the flexibility to look at the quiz and explore, without getting their lesson set as “tested”.
  + Once the user tries the quiz and gets a grade, the lesson’s status is set as “tested” and will appear as completed on the main page.
* Additional notes:
  + Same hashing rules from part 1 apply.
  + Home page addresses the user and shows them their progress.
  + A navigation bar facilitates going from one lesson to the next and back to the home page, allows logging out.
  + Website access requires signing in first.
  + The EMLs can be uploaded to the database via xmlUpload.php, one at a time, specifying proper file URL and name.
  + Please ensure the EML’s ids are numbered 1-6, from 1st to 3rd as units and 4th to 6th as quizzes. (Otherwise, you’d have to change the urls in the navigation bar)
  + Please ensure proper database acces in in connect.php