Multitool Workspace - 2BKI21
Fach - Programmieren

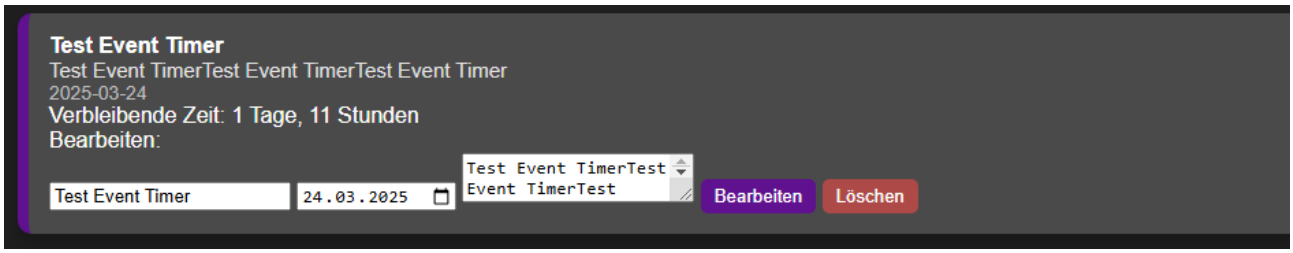


Schaubild 1: Screenshot aus unserem Projekt

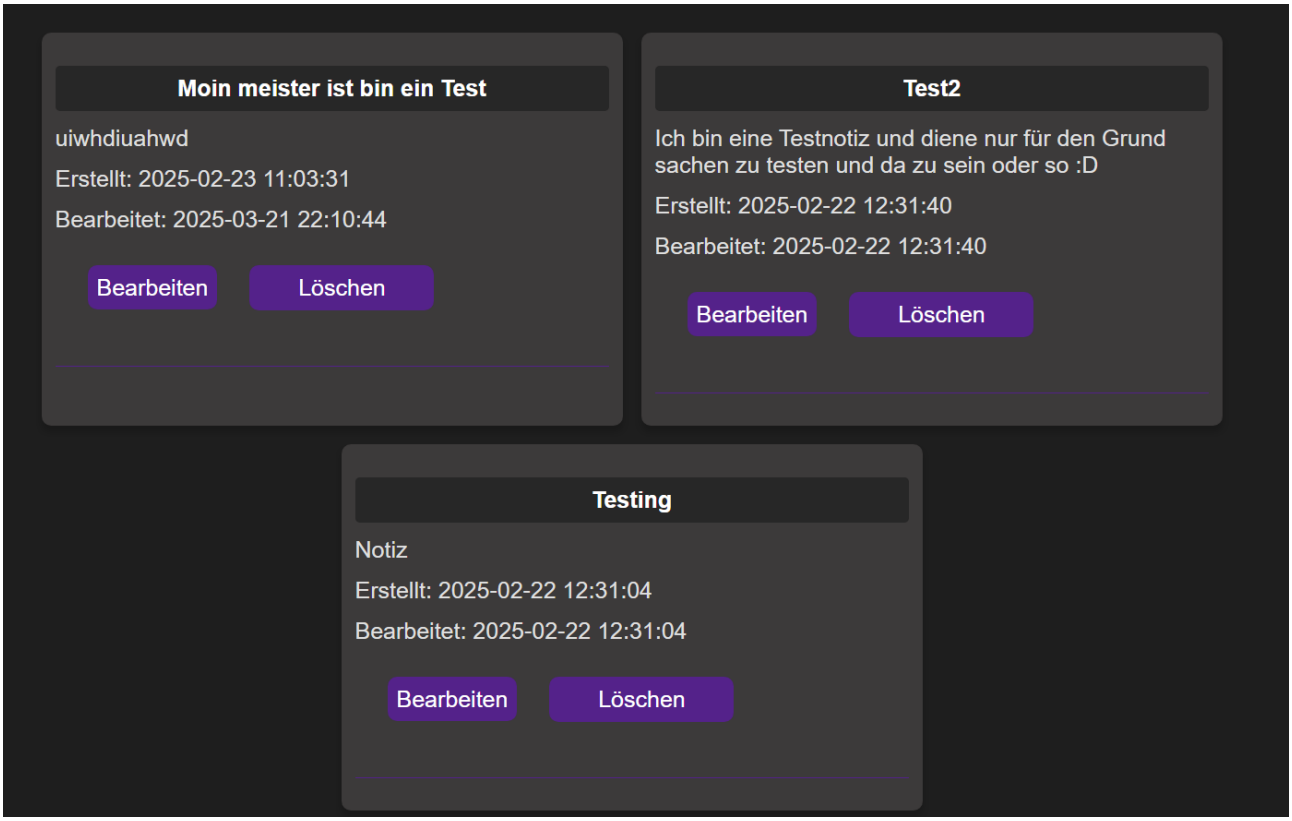


Schaubild 2: Screenshot aus unserem Projekt

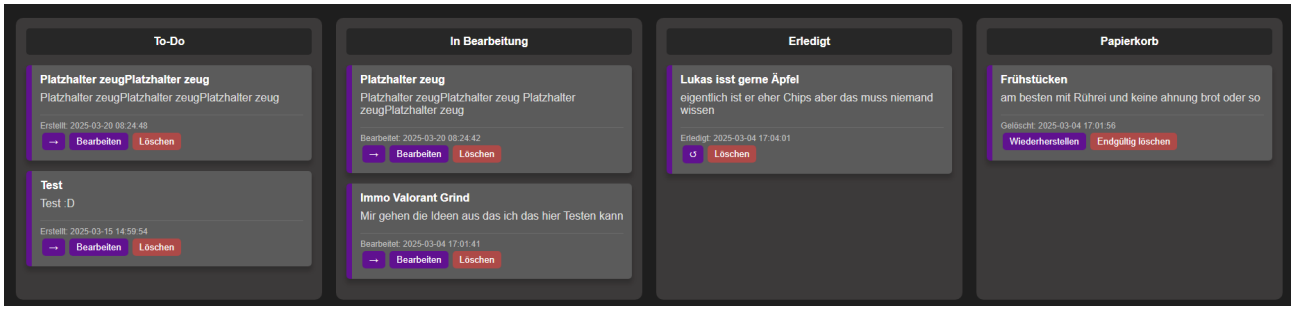


Schaubild 3: Screenshot aus unserem Projekt

Inhaltsverzeichnis

Abbildungsverzeichnis

Vorwort

Ideenfindung

Wir haben uns in den Ferien bereits zusammengeschlossen die Projektarbeit zusammen zu machen. Anfangs wollten wir noch etwas im Bereich der Elektrotechnik machen wie z.B. eine digitale Uhr aber als wir weiter überlegt haben und verschiedene unterschiedliche Ideen hatten haben wir uns überlegt was am Praxisnähesten wäre, was wir auch weiterhin benutzen würden nach dem Abschluss/während der Schule etc. Deswegen haben wir uns überlegt eine Website zu erstellen die für den täglichen Gebrauch nutzbar ist. Da sind uns dann sehr viele Ideen auf einmal in den Kopf gegangen, weshalb wir nicht dachten eine größere Website zu gestalten mit vielen unterschiedlichen Funktionen wie z.B. die To-Do Liste. Wir benutzen HTML, CSS und PHP um die Seite zu programmieren.

Planung

Wir haben uns zuerst für diese Funktionen der Seite entschieden (dies wird noch geändert)

Profilerstellung

Für unsere Seite haben wir uns überlegt, dass man sich ein eigenes Profil erstellen kann und dann dort alle Sachen in seinem eigenen Profil abgespeichert werden (in der Datenbank aber naja)

Kalender

Dadurch da wir wussten das wir Profile auf unserer Seite haben werden haben wir uns überlegt was man mit den Profilen alles erstellen könnte eines davon ist der Kalender wo man sich z.B. Termine reinschreiben kann und diese alle in unserer Datenbank gespeichert werden.

Wetter Abrufe

Wir wollten auch etwas implementieren wozu man etwas externes benötigt um Infos zu bekommen und haben uns dann dazu entschieden eine Wetter abrufe zu machen wo man dann das Wetter jederzeit abrufen kann und dies wollen wir auch auf die Homepage implementieren wo dann das Wetter von z.B. Mannheim dauerhaft angezeigt wird.

Passwort Generator/Manager

Dadurch da ich (Timo :D) Probleme hat seine Passwörter sich zu notieren sind wir auf die Idee gekommen einen Passwort Manager zu erstellen wo man sich Passwörter einspeichern kann, diese Gruppieren kann, löschen kann usw. Dazu passend dachten wir das es vielleicht keine schlechte Idee wäre diesen Passwort Manager noch mit einem Passwort Generator zu erstellen wo man sich dann auch Passwörter erstellen kann und selbst angeben kann wie viele Zeichen wie viel groß klein sein soll etc.

Bewertungsplattform

Wir dachten es wäre eine interessante Idee eine Funktion einzubauen wo man Serien, Filme etc. hochladen könnte und dazu seine eigene Meinung dazu geben kann, das wird dann alles in der Datenbank gespeichert und kann von jedem Benutzer angeschaut werden.

BMI-Rechner

Dadurch da wir schon im Prog Unterricht einen BMI-Rechner erstellt haben dachten wir uns das es keine Schlechte Idee wäre diesen in unsere Seite zu implementieren und zu verbessern

Notizen Speicher

Wir wollten noch eine Funktion erstellen die man auf der Homepage sich anzeigen lassen kann und da dachten wir uns das wir einen kleinen Notizen Speicher erstellen wollen wo man sich kleine Notizen aufschreiben kann und dann diese auf der Homepage angezeigt werden. Dies wird dann alles in der Datenbank gespeichert.

Konverter

Dadurch da es immer mal wieder blöd ist wenn eine Datei den falschen Dateityp hat oder man nur die Audio von einer Datei haben will dachten wir uns das es vielleicht praktisch wäre einen File Konverter zu erstellen.

To-Do Liste

Wir haben uns auch noch überlegt was zu einer Multi-Tool Website gehören könnte, neben den üblichen Sachen wie ein Kalender, Uhrzeit etc. haben wir uns auch noch überlegt eine To-Do Liste zu machen für Sachen die man erledigen muss.

Geld Umrechner

Wir haben lange überlegt was wir noch machen könnten was Menschen helfen könnte. Und dann kamen wir auf die Idee einen Geld Umrechner als Funktion zu implementieren, da Währungen andere Werte haben und es Menschen verwirren könnte.

Countdown

Wir dachten uns neben den ganzen Funktionen könnte man einen Countdown gebrauchen um sich bspw. eine Deadline für etwas bestimmtes wie eine Frist zu merken und um diese nicht zu überschreiten.

Elektrotechnik Rechner

Da wir im vorherigen Jahr Elektrotechnik hatten, dachten wir uns, dass man einen Elektrotechnik Rechner damit z.B. Spannungen, Widerstände usw. man ausrechnen kann.

Drehrad

Wir haben uns auch überlegt in Ehren zu Frau Piemontese einen Drehrad zu machen, weil sie diese schon sehr gern hat. Nämlich haben wir gesehen, dass sie gerne diese benutzt und wir uns dachten, dass man sowas gebrauchen könnte.

Farbpallette für die Seite

Wir haben uns im Vorfeld überlegt bevor wir irgendetwas Programmieren erstmal ein Theme für unsere Seite zu nehmen damit alles einheitlich gestaltet ist und nicht jede Seite anders aussieht.

Um die Farbpallette (Theme der Seite) herauszufinden haben wir die Seite [(https://coolors.co/)](https://coolors.co/) benutzt und uns mehre unterschiedliche Themes uns angeschaut bis wir eine ungefähre Idee hatten von Themes.

Theme 1:

Abbildung 1: Screenshot aus Coolors.co

Theme 2:

Abbildung 2: Screenshot aus Coolors.co

Theme 3:

Abbildung 3: Screenshot aus Coolors.co

Wir haben uns nach gemeinsamer Überlegung für etwas in Richtung Theme 3 entschieden, haben es noch etwas ausgeführt da wir mehr als 4 Farben brauchen und haben am Ende dann diese Farbpalette erstellt:

Abbildung 4: Screenshot aus Coolors.co

Für diese Farbpallette haben wir uns am Ende entschieden.

Der Schwarz und Weißton ist logischerweise für Schriftfarben und Hintergrundfarben gedacht. Während die Verschiedenen “Lila” Töne für die Knöpfe etc. sind.

Datenbank Auswahl

Dadurch da unsere Website viele Funktionen bietet die eine Art des Speicherns braucht haben wir uns überlegt eine Datenbank dafür zu nehmen.

Dadurch dass wir noch keine Erfahrung besitzen haben wir ChatGPT gefragt was es für Datenbanken gibt die wir benutzen könnten. (prompt: Gebe uns verschiedene Datenbanken für unsere Website die wir benutzen können um z.B. Sachen wie Profile mit Passwörtern und Kalendarien abzuspeichern wir benutzen HTML, PHP, CSS, JavaScript.)

ChatGPT hat uns folgende Datenbanken vorgeschlagen: MySQL/Mariadb, SQLite, PostgreSQL, MongoDB, Redis.

Dadurch da wir keine Flexible Struktur benötigen fällt MongoDB raus und da wir keine Temporären Daten wollen fällt Redis raus.

Als wir die engere Auswahl hatten zwischen den drei Letzen Datenbanken haben wir uns für MySQL/Mariadb entschieden, da diese mit PHP kompatibel ist und wir bereits durch das Berufskolleg Erfahrung mit PHP haben. Wir sind dann aber auf SQLlite umgestiegen da wir diese nicht auf einem Server hosten müssen aber trotzdem mit SQL arbeitet.

Projekt Umplanung

Nach dem ersten Milestone haben wir auf das Feedback unserer Lehrer gehört und wollten deshalb die Website einheitlicher machen mit den Funktionen, nachdem wir uns überlegt haben was wir machen wollen haben wir uns dazu entschieden eine Art „workspace“ zu machen mit folgenden Funktionen: Notizen speicher, To-Do Listen, Event Timer.

Projektverteilung

Wir hatten für unsere alte Planung bereits eine Notion Tabelle erstellt um einen Überblick zu bekommen aber nach der Umplanung haben wir uns nur grob überlegt wann wir was machen wollen mit Wochen unsere Planung sah dann am Ende ungefähr so aus:
SW 8 – Projektplanänderung – Geplante Bearbeitung von Timo & Lukas
SW 9-10 – Datenbank Aufsetzung – Geplante Bearbeitung von Timo & Lukas
SW 11-13 – Datenbank Testen in PHP & Profilverwaltung – Geplante Bearbeitung von Timo & Lukas
SW 11-13 – Homepage erstellen – Geplante Bearbeitung von Timo
SW 14-15 – Dokumentation Datenbank & Benutze Tools – Geplante Bearbeitung von Lukas
SW 13-16 – To-Do – Geplante Bearbeitung von Timo
SW 16-19 – Notizen – Geplante Bearbeitung von Timo
SW 16-18 – Event Timer – Geplante Bearbeitung von Lukas
SW 17-20 – Fehlerbehebung + Extra Zeit / Pausenzeit – Geplante Bearbeitung von Timo & Lukas
SW 21-24 – Dokumentation von Quellen, Hilfestellungen, Benutzen Tools und – Geplante Bearbeitung von Timo & Lukas
SW 25-Abgabe – Fertigstellung der Dokumentation, Fehlerbehebungen und nachholung falls etwas fehlerhaft ist – Aufgabe von Beiden.

Login/Register Design

Zuerst mussten wir eine Form(HTML tag) erstellen für die Eingabe Felder Benutzername und Passwort, hierbei gab es keinerlei Probleme da wir dies bereits im Unterricht gründlich gemacht haben.
Zudem haben wir noch einen <a> tag am Ende der Login gemacht sodass man auch zu der Registrierung gelangt falls man noch keinen Account besitzt.

Dadurch da das ganze noch nicht sehr Anschaulich aussieht haben wir eine .css Datei erstellt.
Doch bevor wir in dieser irgendetwas reinschreiben haben wir uns erstmals sehr grob gezeichnet wie die Seite aussehen soll:

Danach ging es zur .css, Wir haben damit angefangen eine neue Class zu erstellen die wir maincontainer genannt haben, dem Container die Größe 500x720 gegeben haben.
Wir haben in dieser .css Datei noch die font-family und die font-size für alle Elemente bestimmt.
Jedoch nachdem wir der Form die Class maincontainer gegeben habe war es noch nicht mittig weshalb wir verschiedenste Sachen probiert haben wie z.B. auf der linken Seite eine Margin geben aber dadurch da wir es nie richtig mittig hinbekommen haben haben wir recherchiert und herausgefunden, dass wenn man bei margin-left: auto; und margin-right: auto; es automatisch in die Mitte verschoben wird.
Jetzt fügen wir noch die Navigationsleiste hinzu indem wir das styleesheet noch verlinken und die Navigationsleiste kopieren und die links ändern da wir uns an einem anderen Ort befinden.
Dadurch da wir bereits in den Placeholdern stehen haben wo man seinen Benutzernamen und Passwort eingeben muss haben wir uns überlegt die Labels zu löschen damit es übersichtlicher aussieht. Danach haben wir noch die Abstände etwas umgeändert damit es nicht so geklemmt aussieht.

Register Css

Dadurch da jetzt die Login Seite fertig ist haben wir mit der Register Seite angefangen, wir haben zuerst die Navigationsleiste eingefügt und das stylesheet von der nav verknüpft mit dem <link> tag, wir haben alle classes von der Login Seite übernommen und der Klasse Maincontainer zu der <form> hinzugefügt wie wir es beim Login gemacht hatten, zudem haben wir Placeholder wieder geschrieben und die Labels entfernt da wir diese nicht brauchen.
Es kam danach zum ersten Fehler, dadurch da das PHP Skript nicht teil der <form> ist schreibt es die Ausgaben außerhalb des angelegten Kasten (class maincontainer), um dieses Problem zu lösen haben wir eine erstellt die auch das PHP Skript mit einbehält, dieser haben wir Maincontainer zugewiesen und haben aus der form die form entfernt. Nun hat alles einwandfrei funktioniert.

Navigationsleiste

Dadurch da wir auf jeder unserer Seiten eine Navigationsleiste haben wollen um auf der Seite zu Navigieren erstellen wir uns auf (erstmals) einer Seite den tag <nav> in diesen <a> tag fügen wir nun für jeden Link zu einer anderen Seite den <a> tag. Anfangs haben wir dies über den <ul> tag gemacht jedoch haben wir dann gesehen, dass HTML den <nav> tag bereits hat und haben dann lieber die vorgefertigte HTML Funktion genommen.

```
<nav>
    <a href="notes/notizen.php">Notizen</a>
    <a href="todo/todo.php">To-Do</a>
    <a href="event/event.php">Event-Countdown</a>
    <a href="user/login.php">Login</a>
</nav>
</body>
```

Notizen To-Do Event-Countdown Login

Man kann nun auf alle verschiedenen Knöpfe drücken und man wird dann auch zu den richtigen Seiten weitergeleitet jedoch sieht es noch nicht sehr anschaulich aus, weshalb wir eine .css Datei erstellen um das Aussehen und optischen Verhalten der Seite zu verändern.
Zuerst haben wir die Hintergrund Farbe auf ein dunkles Lila geändert, dann noch die Breite auf 100% gestellt damit die Breite sich über die ganze Seit erstreckt egal wie groß dein Gerät ist und dann noch die Höhe auf 100px.

Damit hatten wir nun eine Fläche wo wir nun unsere <a> tags noch vergrößern können und am Besten als Knöpfe darstellen lassen.
Dafür müssen wir in der .css Datei nun den tag <nav a> ändern, zuerst haben wir wieder die Hintergrundfarbe geändert und zwei Größen angegeben, dann haben wir auch noch die Schriftgröße und Farbe geändert.

Jedoch ergab sich dabei ein Problem, die Hintergrundfarbe wurde zwar geändert von den Elementen jedoch wurden sie nicht größer, dies kann jedoch relativ einfach mit der Hilfe von W3Schools gelöst werden, indem wir uns dort den Abschnitt CSS – Layout display: inline-block; angeschaut haben, dort steht beschrieben das im Gegensatz zu dem display: inline bei inline-block die Größe von Elementen geändert werden kann, deshalb müssen wir um das Problem zu lösen nur den display auf inline-block ändern.
Jetzt wollen wir auch noch den Text zentrieren, dafür haben wir dann einfach text-align: center; benutzt, doch das macht den Text zwar Vertikal Mittig nur noch nicht Horizontal, weswegen wir einen kleinen Trick benutzt haben das wenn man die line-height genauso groß einstellt wie die height des Elements das es dann sich zentriert.

Jetzt sieht die Navigationsleiste so aus.

Navigationsleiste: Elemente Positionieren

Dadurch da die Elemente alle noch nicht wirklich an guten Positionen stehen, haben wir damit angefangen die Elemente Vertikal mittig zu positionieren und das haben wir gemacht, indem wir einen Margin-top hinzugefügt haben, diesen Wert für das Margin-top war 10px da das <a> Element 80px groß war und die ganze Navigationsleiste 100px und wir unten oben den gleichen Abstand haben wollen. Für den Abstand rechts und links gab es jedoch ein Problem, da wir wollen dass die 4 Elemente mittig stehen, also müssen wir bei der nav das display auf flex stellen und justify-content auf center, somit sind jetzt die Elemente mittig angereiht.
Zuletzt mussten wir nun nur noch einen Margin links und rechts einstellen damit die Elemente nicht zusammengepresst aneinander stehen.

Navigationsleiste: Homebutton

Nachdem wir bereits angefangen hatten die Funktionen vom Login und Register zu programmieren ist uns aufgefallen, dass wir keinen Weg haben um zurück zur Hauptseite/Homepage zu kommen weshalb wir in die Navigationsleiste einen Button hinzufügen wollten.
Zuerst haben wir einen neuen <a> tag erstellt wo wir dann einen Emoji eingefügt haben von einem Haus um anzudeuten dass es ein Homebutton ist, diesen haben wir eine class gegeben die wir in der nav.css .homebutton genannt haben, wir haben die Größe auf 60x60 Pixel eingestellt, damit er kleiner ist als die anderen Pixel, der Rest war gleich außer dass wir Anpassungen machen mussten damit er wieder mittig ist, indem wir den Margin-top erhöht haben, zudem haben wir einen größeren border-radius eingestellt.

Jetzt gab es nur noch ein Problem, die Position des Buttons war noch zu sehr bei den Anderen Buttons was wir nicht wollten, wir haben probiert einen margin-right zu machen, welcher jedoch auch die anderen Buttons verschiebt, daher dass wir keine Lösung für das Problem gefunden haben, haben wir dann im Internet nach einer Lösung gesucht, auf w3schools.com/css/css_positioning.asp und developer.mozilla.org/en-US/docs/Web/CSS/position wurden wir dann fündig, denn indem wir die Position des Buttons auf absolute stellen war dieser unabhängig von den anderen Buttons.
Nachdem dieser unabhängig war müssten wir noch einen margin-right einstellen, sodass dieser nicht in der Mitte vom Bildschirm ist.
Nun funktioniert der Homebutton und war an der richtigen Position.

Navigationsleiste: Hover

Buttons ohne Hover

Nun sieht die Startseite des Projektes so aus zu Letzt wollen wir noch hinzufügen das wenn man mit der Maus über einem Element ist das es die Farbe ändert dies kann erfolgen indem man bei der .css Datei diesen Schritt macht:

```
32  nav a:hover {
33      background-color: #4E1E84;
34  }
```

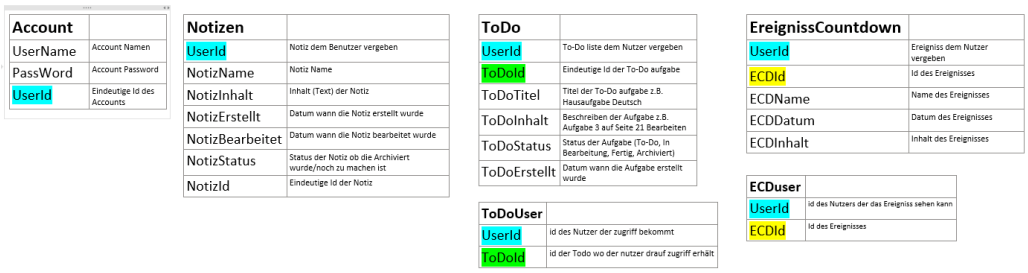
Jetzt wird die Farbe geändert wenn man über das Element „hovert“.

Buttons mit Hover

Damit wäre die Navigationsleiste vorerst fertig und braucht erst später eine Änderung um anzuzeigen auf was für einem Element man sich befindet.

Datenbank aufsetzen

Zuerst haben wir uns in OneNote einen Datenbank Prototypen gezeichnet wo wir alle unsere Tabellen einzeichnen für alle unsere Seiten und Funktionen, am Ende sah diese dann wie folgt aus:



Account	
UserName	Account Namen
PassWord	Account Password
UserId	Eindeutige Id des Accounts

Notizen	
UserId	Notiz dem Benutzer vergeben
NotizName	Notiz Name
NotizInhalt	Inhalt (Text) der Notiz
NotizErstellt	Datum wann die Notiz erstellt wurde
NotizBearbeitet	Datum wann die Notiz bearbeitet wurde
NotizStatus	Status der Notiz ob die Archiviert wurde/noch zu machen ist
NotizId	Eindeutige Id der Notiz

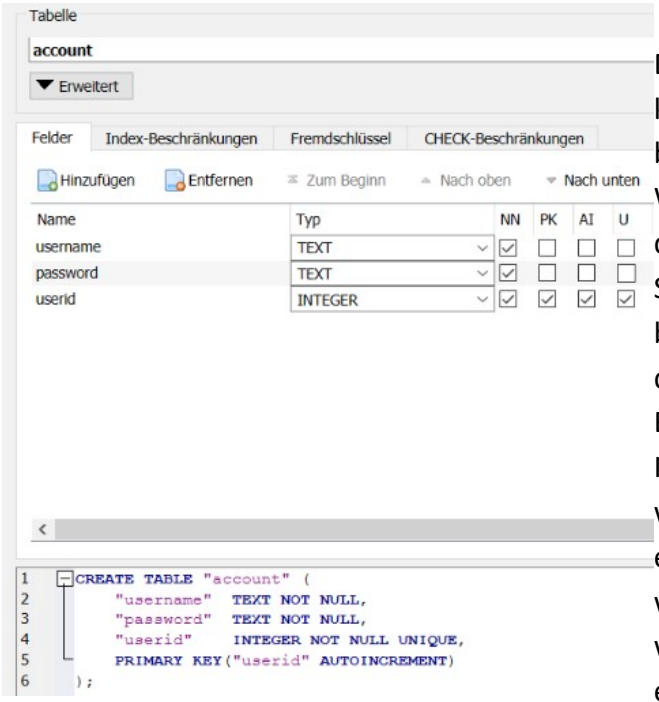
ToDo	
UserId	To-Do liste dem Nutzer vergeben
ToDoId	Eindeutige Id der To-Do aufgabe
ToDoTitel	Titel der To-Do aufgabe z.B. Hausaufgabe Deutsch
ToDoInhalt	Beschreiben der Aufgabe z.B. Aufgabe 3 auf Seite 21 Bearbeiten
ToDoStatus	Status der Aufgabe (To-Do, In Bearbeitung, Fertig, Archiviert)
ToDoErstellt	Datum wann die Aufgabe erstellt wurde

ToDoUser	
UserId	id des Nutzer der zugriff bekommt
ToDoId	id der Todo wo der nutzer drauf zugriff erhält

EreignissCountdown	
UserId	Ereigniss dem Nutzer vergeben
ECDId	Id des Ereignisses
ECDName	Name des Ereignisses
ECDDatum	Datum des Ereignisses
ECDInhalt	Inhalt des Ereignisses

ECDuser	
UserId	Id des Nutzers der das Ereigniss sehen kann
ECDId	Id des Ereignisses

Wir haben uns den DB Browser installiert um die Datenbank aufzusetzen wie wir sie bereits in OneNote haben.



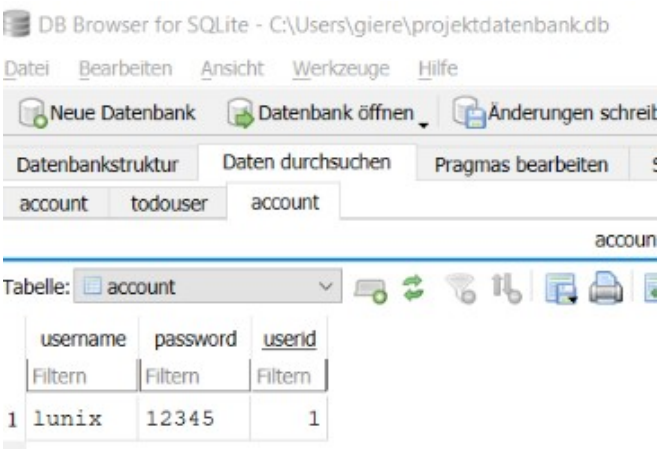

```
CREATE TABLE "account" (
    "username"   TEXT NOT NULL,
    "password"   TEXT NOT NULL,
    "userid"     INTEGER NOT NULL UNIQUE,
    PRIMARY KEY("userid" AUTOINCREMENT)
);
```

Das war jetzt unsere erste Tabelle die wir für uns kreieren, diese wird für die Benutzerverwaltung benutzt.
Wir haben bei username und password jeweils den Datentyp TEXT genommen, damit wir es als String später im Code haben, zudem haben wir bei allen NOT NULL aktiviert, das bewirkt dann das es nicht Leer sein kann.
Bei der userid haben wir zudem den Datentyp Integer angegeben, sodass die ID eine Zahl ist wie z.B. 63, wir haben auch Autoincrement eingestellt damit diese Automatisch vergeben wird, unique das jeder nur einmal vergeben werden kann und Primary key damit wir es eindeutig zuteilen können.

Diesen Schritt haben wir jetzt jeweils für alle unsere Tabellen gemacht nur das die userid kein Autoincrement, Primary key, unique da wir es bereits getan haben bei der Account erstellen und da die userid bereits erstellt ist.

Datenbank Werte einfügen

Jetzt haben wir unsere Datenbank vollständig aufgesetzt, jetzt wollen wir zum testen in die Tabelle Accounts einen Testnutzer erstellen, dazu mussten wir ein SQL Skript ausführen.
Zuerst müssten wir dafür INSERT INTO account einfügen, damit es in die Tabelle account hineinschreibt.
In der Zweiten Zeile dann die Werte die hinein gefügt werden die also für den Namen das Passwort und die ID.
Dies funktionierte wie man auf dem Abbild sieht,
Nun müssen wir dies noch mit einem PHP Skript verwirklichen.



Register Datenbank Werte Eintragen

Im PHP-Skript haben wir als erstes versucht eine Verbindung mit der Datenbank aufzubauen.

```php
// Verbindung zur SQLite-Datenbank herstellen
    $db = new PDO('sqlite:C:\xampp\htdocs\Projektarbeit\database\projektdatenbank.db');
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Dafür haben wir zuerst mit „new PDO“ eine Verbindung mit unserer PHP-Datei und einem Datenbankserver aufgebaut. Danach haben wir einen „setAttribute“ gemacht. Dies ist eine wichtige Einstellung, die bestimmt, wie PDO (PHP Data Objects) Fehler im Zusammenhang mit der Datenbankverbindung und SQL-Abfragen behandeln. Danach kam der kompliziertere Teil, nämlich die Variablen in einer try-catch-Exception verwenden. Aber als erstes haben wir die ID’s aus der HTML Zeile mit „trim($_REQUEST[ ])“ die Variablen deklariert. „trim“ ist dafür da um sogenannte „Whitespaces“ am Anfang oder am Ende eines Strings zu entfernen. Nach langer Recherche haben wir herausgefunden, dass wir in unserem PHP-Code die „try-“ und „catch-Exception“ bräuchten. Exceptions können in PHP „geworfen“ werden (throw) und „gefangen“ werden (catch). Um Sachen abzufangen sollte der Code von einem „try-Block“ umschlossen werden. Jeder „try-Block“ muss einen zugehörigen „catch-“ oder „finally-Block“ besitzen. Wir haben es beispielsweise mit einem „try-Block“ und „catch-Block“ gemacht. „catch-Block“ definiert, wie auf eine ausgelöste Exception reagiert werden soll. Wenn eine Exception ausgelöst wird und der aktuelle Funktionsbereich keinen „catch-Block“ hat, steigt die Exception im Aufrufstapel bis zur aufrufenden Funktion auf, bis sie einen passenden „catch-Block“ findet.

```php
$username = trim($_REQUEST['usrnm']);
$password = trim($_REQUEST['passwd']);

    try{
        // Benutzer in der Datenbank hinzufügen
        $statement = $db->prepare("INSERT INTO account (username, password) VALUES (:username, :password)");
        $statement->bindParam(':username', $username);
        $statement->bindParam(':password', $password);
        $statement->execute();

        echo "Registrierung war erfolgreich! Sie können sich jetzt anmelden.";
    } catch(PDOException $e){
        // Fehler behandeln, z.B. wenn der Benutzername schon existiert
        if ($e->getCode()=='23000'){
            echo "Benutzername ist bereits registriert.";
        } else{
            echo "Ein Fehler ist aufgetreten :/". $e->getMessage();
        }
    }
```

Try-Block und Catch-Block

Im try-Block soll ein Benutzer in der Datenbank hinzugefügt werden. Dafür wurde die Variable „$statement“ mit „prepare“ deklariert. Die Funktion prepare bereitet eine SQL-Anweisung zur Ausführung vor und liefert ein Anweisungsobjekt. In diesem Fall wurde die Anweisung „INSERT INTO account“ vorbereitet, damit der eingegebene Benutzername und das Passwort in die entsprechenden Spalten „username“ und „password“ der Datenbank eingefügt werden. Danach werden die Parameter „:username“ und „:password“ mit der Funktion bindParam an die Variable „$statement“ gebunden. Diese Funktion sorgt dafür, dass die Eingabedaten später korrekt an die SQL-Anweisung übergeben werden. Zum Abschluss des try-Blocks wird mit „execute“ die vorbereitete SQL-Anweisung ausgeführt. Bei erfolgreicher Ausführung gibt ein „echo“ die Nachricht aus, dass die Registrierung erfolgreich war.

Nach dem try-Block folgt der catch-Block, der zur Fehlerbehandlung dient. Hier wurde eine sogenannte PDOException eingebaut, die Fehler abfängt, die von PDO ausgelöst werden. Innerhalb des catch-Blocks wurde eine if-Abfrage implementiert. Diese Abfrage überprüft mithilfe der Exception-Methode getCode, ob der Fehlercode „23000“ vorliegt, was bedeutet, dass der Benutzername bereits in der Datenbank existiert. Wenn dies der Fall ist, wird eine entsprechende Fehlermeldung ausgegeben. Falls der Fehler nicht mit „23000“ übereinstimmt, wird ein „else“-Block ausgeführt, der die Nachricht „Ein Fehler ist aufgetreten :/“ ausgibt. Am Ende des catch-Blocks wurde getMessage verwendet, um die detaillierte Beschreibung der Exception anzuzeigen. Als der Code schließlich getestet wurde, trat eine Fehlermeldung auf:

„Fatal error: Uncaught PDOException: could not find driver in C:\xampp\htdocs\Projektarbeit\user\register.php:22 Stack trace: #0 C:\xampp\htdocs\Projektarbeit\user\register.php(22): PDO->__construct('C:\\xampp\\htdocs...') #1 {main} thrown in C:\xampp\htdocs\Projektarbeit\user\register.php on line 22“.

Zur Behebung des Fehlers wurde ChatGPT als Hilfsmittel genutzt. Nach Analyse des Codes stellte sich heraus, dass der Verbindungsstring für SQLite fehlerhaft war. Die Datei im Verbindungsstring war falsch angegeben. Der Fehler wurde behoben, indem „$db = new *enter* PDO('C:\xampp\htdocs\Projektarbeit\database\projektdatenbank.sqbpro');“ durch „$db = new (ohne enter) PDO('C:\xampp\htdocs\Projektarbeit\database\projektdatenbank.sqbpro');“ ersetzt wurde. Außerdem wurde die Dateiendung „projektdatenbank.sqbpro“ zu „projektdatenbank.db“ geändert. Nach diesen Änderungen lief das Programm fehlerfrei.

Login Funktionalität

Dadurch da wir jetzt einen funktionierenden Register haben, wollen wir natürlich noch einen Login machen, die ersten zwei Zeilen aus dem Code können wir übernehmen wie sie bereits sind, somit haben wir nun wieder eine Verbindung zur Datenbank, die Variablen bekommen wir auch genau gleich wie beim Register mit $_POST.
Nun machen wir wieder ein $statement wie beim Register nur diesmal machen wir kein INSERT, sondern ein SELECT da wir nichts in die Datenbank einfügen möchten, sondern wir Werte aus der Datenbank bekommen möchten, genauer genommen wollen wir alle Werte (*) von der Tabelle „account“ nehmen wo der eingegebene „username“ mit dem „username“ übereinstimmt der in der Datenbank steht, damit wir das eingegebene Passwort mit dem Datenbank Passwort übereinstimmt, am Ende sollte das $statement dann so aussehen:

```php
$statement = $db->prepare(query: "SELECT * FROM account WHERE username = :username");
```

Die SELECT haben wir über diese Seite herausgefunden
https://www.tutorialspoint.com/sqlite/sqlite_select_query.htm
Bevor wir den $statement ausführen, müssen wir noch den Parameter binden, damit es auch weiß woher der „username“ überhaupt genommen wird, jetzt erstellen wir eine neue Variabel,
in dieser Variabel machen wir nun noch ein statement mit einem fetch, der fetch gibt uns dann die Tabellenreihe als array aus, damit wir dieses array dann auch benutzen können, können wir zum Beispiel eine neue Variabel erstellen die wir nun $datapass nennen und ihr den Wert von der fetch das password ausgeben so ('password') und keine doppelten, da es ein Array ist und kein String!
Jetzt da wir das Passwort als Variabel haben, können wir nun eine If-Verzweigung erstellen die überprüft ob das eingegebene Passwort mit dem Passwort aus der Datenbank übereinstimmt.
Wenn dieses Passwort dann stimmt, wollen wir noch der Session die Werte für den Benutzernamen und der userid zuweisen, damit wir diesen dann auf Seiten wie dem Notizenspeicher benutzen können, damit er auf dem Benutzer gespeichert wird.
Wenn aber das Passwort falsch ist soll die Session abgebrochen werden.
Um zur Startseite wieder zurückgeleitet zu werden einfach die PHP-Funktion „header“ benutzen.

```php
$result = $statement->fetch(mode: PDO::FETCH_ASSOC); // Tabelle als Array ausgeben
if ($result) {
    $datapass = $result['password'];
    if ($datapass == $pass) {
        $_SESSION['sesuser'] = $result['username']; // Session werte vergeben
        $_SESSION['sesid'] = $result['userid']; // Session werte vergeben
        header(header: 'Location: ../index.html'); // Weiterleitung zur Startseite (hoffentlich auch mit Session werten)
    }
    else {
        session_abort(); // Session abbrechen wenn Passwort falsch
        echo"Irgendwas ist Falsch"; // Am besten eine richtige Fehlermeldung anzeigen lassen als Pop up
    }
}
```

Notizen Funktionalität

Notizen Funktionalität: Datenbank Attribute erklärt

Für unseren Notizenspeicher haben wir bereits die Datenbank Tabelle erstellt die haben wir „notizen“ genannt, in dieser Datenbank Tabelle gibt es mehrere Attribute:

- userid: Das ist die gleiche wie bei dem Login und Register. Wir benutzen sie dafür um herauszufinden für wen die Tabelle ist also wer sie einsehen kann.
- notizid: notizid ist ähnlich wie die userid nur das diese nur für die Notizen sind, damit man diese Eindeutig unterscheiden kann, weil wenn man dafür zum Beispiel den Notiznamen nehmen würde könnte es zu Problemen führen da man einer Notiz ja den gleichen Namen geben könnte. Die notizid wird automatisch vergeben sobald eine Notiz dann erstellt wird.
- notiz_inhalt: Dieser Attribut bestimmt darüber was in der Notiz steht z.B. 2 Liter Milch kaufen und 3 Eier.
- notiz_erstellt: Dieser Attribut ist dafür da um einzusehen wann die Notiz erstellt wurde, das wollen wir dann auch anzeigen, damit der Benutzer genau einsehen kann von wann die Notiz ist.
- notiz_bearbeitet: Dieser Attribut ist ähnlich wie notiz_erstellt nur das dieser dafür da ist um sehen wann die Notiz zuletzt bearbeitet wurde, dies wollen wir dem Benutzer nicht anzeigen aber wir wollen, dass die Notiz die zuletzt bearbeitet wurde als erstes angezeigt wird.
- notiz_status: Wir wollen den Attribut notiz_status benutzen um anzuzeigen, ob die Notiz gelöscht ist oder nicht, da wir uns überlegt hatten, dass man die Notizen wiederherstellen kann, deswegen ist wenn dieser Attribut auf 1 steht die Notiz aktiv und wenn der Wert auf Null ist soll er für den Benutzer nur über die Wiederherstellungsfunktion dann einzusehen sein.

Notizen Funktionalität: Datenbankverbindung + Session

Da wir jetzt genau wissen für was wir all unsere Datenbank Attribute verwenden wollen können wir uns an den Hauptcode des Notizenspeichers machen.
Zuerst beginnen wir wieder damit PHP zu initialisieren.
Jetzt wollen wir wieder eine Datenbankverbindung haben, das machen wir genau gleich wie bei den anderen Seiten mit dem try-catch-Block, also haben wir einfach den Code vom Login 1zu1 rüber kopiert.

Jetzt da wir eine Datenbankverbindung haben, wollen wir wieder die Funktion session_start machen und dann überprüfen ob der Nutzer eingeloggt ist, dies wollten wir dann wieder mit der Variabel „sesid“ machen die wir im Login vergeben haben, weil wenn der Benutzer nicht die Variabel „sesid“ hat, dann wissen wir dass er nicht eingeloggt ist.
Wir hatten zuerst probiert zu überprüfen ob die „sesid“ einen Wert in einem bestimmten Zahlenbereich hat z.B. so:
if ($_SESSION[‘sesid’] ==< 0)
Aber dann haben wir uns überlegt, dass es dafür einen besseren Weg geben muss. Und sind dann auf die Funktion „isset“ gestoßen (https://www.php.net/manual/de/function.isset.php).
Also dachten wir uns, dass wenn „sesid“ „isset“ ist, dass dann der Code dann normal ausgeführt wird und man sonst auf die Login Seite geführt wird.
Aber jetzt dachten wir uns, dass es vielleicht geschickter wäre zu überprüfen, ob die Variabel nicht nicht Null ist, also haben wir vor dem „isset“ noch ein ! gemacht dass es es negiert.
Jetzt wollten wir, dass wenn die Bedingung erfüllt ist noch machen, dass man zur Login Seite zurückgeführt wird, dies haben wir dann mit der Funktion „header“ (https://www.php.net/manual/en/function.header.php) gemacht.

Notizen Funktionalität: Notiz Erstellung

Jetzt wollen wir eine Funktion erstellen womit wir eine Notiz erstellen können. Das machen wir natürlich mit function und schreiben dort in die Klammer noch die Variabel $db, damit die Funktion auch mit der Datenbank dann kommunizieren kann.
Zuerst wird mit einer if-Abfrage überprüft, ob überhaupt ein Formular abgeschickt wurde, REQUEST_METHOD und wir wollen noch einen Knopf machen in HTML, wo man dann draufdrücken kann um die Notiz zu erstellen, deswegen haben wir dort noch als Bedingung reingemacht, dass neue_notiz gesetzt ist. Die if-Abfrage sieht jetzt so aus:
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['neue_notiz'])) { }.

Jetzt wollen wir die Werte(Attribute der Datenbank Tabelle) deklarieren die wir in die Datenbank einfügen wollen.

- userid: Die userid bekommen wir wieder durch die Session, indem wir aus der $_SESSION Variabel uns den Wert „sesid“ geben da diese mit der userid identisch ist.
- Zeit: Die Zeit für das Erstellungsdatum und wann es zuletzt bearbeitet wurde bekommen wir mit der date(https://www.w3schools.com/php/func_date_date.asp , https://www.php.net/manual/de/function.date.php) Funktion, wir haben es im Format Y-m-d H:i:s formatiert: Y=Jahr, m=Monat, d=Tag, H=Stunde (24 Stunden Metrik), i=Minute, s=Sekunde.
- Name & Inhalt: Den Namen und Inhalt werden wir dann wieder aus einem HTML Format entnehmen, wir haben die Werte erstmals notiz_name und notiz_inhalt genannt.
- Status: Wir wollten, dass der Status entweder 0 oder 1 ist um zu deklarieren ob die Notiz angezeigt wird, also haben wir einfach eine Variabel namens Status erstellt und setzen diese auf 1.

Jetzt wollen wir die Werte auch in die Datenbank eintragen, dies wollen wir wieder genau so machen wie beim Register mit prepare in einem try catch.
In die query setzen wir nun einen INSERT INTO notizen und als Werte haben wir dann die Attribute der Tabelle (userid, notiz_name etc.).

Die Parameter die wir in die Tabelle einsetzen, setzen wir nun auf unsere zuvor erstellten Variablen.
Falls dabei ein Fehler auftritt geben wir eine Fehlernachricht aus.

Notizen Funktionalität: Notizen Abrufen

Jetzt können wir Notizenspeichern, aber wir können sie uns noch nicht anschauen, wir wollten dies Anfangs auch in einer Funktion machen aber dachten uns dann, dass dies so oder so Sichtbar ist und wir uns deshalb die paar Zeilen Code sparen können.
Wir haben uns also zuerst die userid geholt von der Session, weil wir nur die Notizen von einem jeweiligen Nutzer haben möchten.
Dann haben wir es so wie beim Login(/Register) gemacht und haben einen try-catch-Block gemacht wo ein prepare drinnen steht, in der query steht aber diesmal SELECT, weil wir uns schließlich Werte ausgeben möchte und nicht einfügen möchten. Bei dem SELECT nehmen wir dann die Tabelle notizen natürlich, wir wollen dann mit dem WHERE Keyword überprüfen ob die userid und der Status 1 ist damit nur die richtigen Notizen angezeigt werden.
Jetzt kommt noch hinzu das wir das Keyword ORDER BY (https://www.w3schools.com/sql/sql_orderby.asp) benutzen und dann nach notiz_bearbeitet (descending) sortieren damit die neuesten oben sind.
Es gibt auch hier wieder ein catch mit einer Fehleranzeige falls etwas falsch läuft.

Notizen HTML

Jetzt ist der fast gesamte PHP Quellcode fertig, jetzt müssen wir nur so ziemlich nur noch das HTML & CSS machen.
Wir haben damit angefangen wieder die Navigationsleiste zu übernehmen.
Wir haben erstmals einen Button erstellt der dann dafür dienen soll, das Feld zu öffnen wo man dann eine neue Notiz mit erstellen kann.
Wir hatten uns überlegt wie wir es machen können das man das Feld nur sieht wenn man den Knopf drückt.
Wir haben dann eine Lösung mit einer Zeile Javascript gefunden, weil wenn man eine div erstellt und dieser div den style display None gibt und ihn dann mit Javascript verändert (z.B. display block) das man das Feld dann nur sieht wenn man den Knopf gedrückt sieht.

```
<button onclick="document.getElementById('notizForm').style.display='block';">Notiz Erstellen</button>
<div id="notizForm" style="display:none;">
    <form action="" method="post">
        <input type="text" name="notiz_name" placeholder="Notiz Titel">
        <br>
        <textarea name="notiz_inhalt" placeholder="Notiz Inhalt"></textarea>
        <br>
        <button type="submit" name="neue_notiz">Speichern</button>
    </form>
</div>
```

(https://www.w3schools.com/jsref/met_document_getelementbyid.asp)
Wir haben nun auch getestet einen Wert einzutragen, dies hat auch funktioniert im DB Browser wurde uns alles angezeigt:

	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	27	1	Testing	Notiz	2025-02-22 12:31:04	2025-02-22 12:31:04	1

Jetzt müssen wir noch einstellen, dass uns die Notizen angezeigt werden.
Wir haben die Notizen bereits im PHP-Code in einer Variabel gespeichert, also haben wir erstmals wieder PHP gestartet und die Variabel ausgegeben, das hat uns nur komische Sachen ausgegeben, da es als Array abgespeichert ist, also haben wir uns überlegt es mit einer foreach Schleife auszugeben, das hat nicht funktioniert und hat uns einfach nur nach langem laden angezeigt, dass die Datenbank gesperrt ist.
Aber dieser Fehler lag nicht an dem Code, sondern als wir den Fehlercode gegoogelt haben lag es daran das wir noch DB Browser offen hatten.
Jetzt wurde uns alles richtig angezeigt (ohne Design außer nav)(und ohne Löschknopf der kommt noch).

Notizen CSS

Für das CSS haben wir vor der foreach Schleife eine erstellt als Container wo dann alle Notizen reinkommen, dann haben wir in der foreach Schleife noch eine div erstellt die für jede Notiz erstellt wird neu erstellt wird, dann haben wir überlegt für alles Klassen zu erstellen aber dann kam uns die Idee das wir uns in PHP Code sparen können und nur die HTML Attribute verändern die in der sind.

Notizen Bearbeiten

Um Notizen zu bearbeiten hatten wir Schwierigkeiten herauszufinden wie man die Notizen bearbeiten kann, anfangs wollten wir das man in die Felder wo die Notizen drinstehen bearbeiten kann aber nachdem wir das probiert haben fanden wir das es nicht so aussah wie wir es wollten. Danach hatten wir die Idee irgendwie ein Feld zu öffnen wenn man auf bearbeiten drückt was aber kaum möglich war mit PHP weshalb wir uns überlegt hatten das wenn man auf bearbeiten geht man auf Notizen.edit.php weitergeleitet wird wo dann die Notizen bearbeitbar sind (mit der Anfangs Idee) und man dann das Formular schicken kann, das ist zwar nicht die Schönste Lösung aber es Funktioniert.

Notizen Löschen

Um Notizen zu Löschen erstellen wir im PHP Code eine Funktion die überprüft ob der Knopf löschen gedrückt ist (den erstellen wir dann gleich in der foreach).
Diese Funktion macht wieder ein try catch Block, dort wird dann wieder ein SQL query gemacht wo der Notiz_Status auf 0 gesetzt wird wo die Notizid und die Userid mit der vom html übereinstimmt.
Um die Funktion nun auszuführen erstellen wir einen Button mit type submit und geben ihm den Namen loeschen damit die Funktion ausgeführt wird wenn er gedrückt ist.

To-Do Planung

Für unsere To-Do Liste hatten wir uns überlegt oben ein Feld mit Textarea und normalem Input Feld zu machen wo man dann der Todo einen Titel und Inhalt geben kann, im Backend sollte dann wenn man das Formular abschickt die geschriebenen Inhalte in die Datenbank eingetragen werden, zudem auch noch die userid und die Zeit wo es erstellt wurde. Alle bereits in der Datenbank eingetragenen ToDos sollen in 4 Spalten aufgeteilt werden (To-Do, in Bearbeitung, Erledigt, Papierkorb). Alle To-Dos sollen standart wertig in der Spalte To-Do eingetragen werden und dann immer nach rechts verschoben werden können, außer beim Papierkorb, dafür wollten wir einen Knopf machen den man jederzeit drücken kann um die To-Dos dann in den Papierkorb zu schieben, im Papierkorb wollen wir dann 2 Knöpfe machen die dann zum Wiederherstellen und zum entgültigen Löschen sind. Man soll auch wenn die To-Dos nicht im Papierkorb oder erledigt ist sie bearbeiten können (Titel und Inhalt). Zudem wollen wir, dass die Notizen die zuletzt bearbeitet worden ganz oben sind.

To-Do Funktionalität: To-Do Erstellung

Für die To-Do Funktion erstellen wir eine neue function „todoSpeichern“ ähnlich wie bei den Notizen, damit wir die todos in der Datenbank speichern können. Als Parameter übergeben wir wieder die $db Variabel, damit wir in der Funktion Zugriff auf die Datenbank Verbindung haben. Genau wie bei den notizen haben wir eine if-Verzweigung erstellt wo überprüft wird, ob die REQUEST_METHOD gleich POST ist und ob der Knopf neue_todo angeglickt wurde (mit der Funktion isset), damit überprüfen wir ob das Formular abgeschickt wurde.
Nachdem wir das Formular gecheckt haben brauchen wir jetzt die Werte die wir in die Datenbanktabelle todo eintragen wollen:

- userid: Die userid haben wir wieder aus der Session genommen um zu wissen zu welchem Benutzer die To-Do gehört.
- Zeit: Für die Zeit haben wir genau wie bei den Notizen die Funktion date benutzt wo wir dann das Datum und die Uhrzeit bekommen in dem Datenbank Format.
- Titel & Inhalt: Den Titel und Inhalt bekommen wir aus den Input Feldern die wir später im html einfügen. Wir benutzen die Funktion trim (https://www.php.net/manual/de/function.trim.php) damit keine unnötigen Leerzeichen davor oder danach sind.
- Status: Wir haben den Status auf 1 gesetzt damit wir später überprüfen können in welcher Spalte die To-Do angezeigt werden soll, wie vorher erwähnt soll jede erstellte To-Do in der 1. Spalte stehen.

Dann haben wir wieder einen try-catch Block erstellt wie bei den Notizen. Im try block haben wir dann mit prepare einen INSERT INTO gemacht wo wir alle Werte in die todo Tabelle einfügen. Mit bindParam haben wir dann den Variabeln die Parameter übergeben. Falls dabei ein Fehler auftritt fängt der catch Block den Fehler und gibt eine Nachricht aus.

To-Do Funktionalität: To-Do Bearbeitung

Nachdem wir eine Funktion zum erstellen von To-Dos haben brauchen wir natürlich auch eine Funktion zum Bearbeiten. Dies machen wir mit der function todoBearbeiten wo wir auch wieder die $db Variable als Parameter übergeben, damit wir auf die Datenbank zugreifen können. Genau wie beim erstellen haben wir eine if-Verzweigung erstellt wo geprüft wird, ob die REQUEST_METHOD auf POST steht und ob der Knopf todo_bearbeiten gedrückt wurde, dadurch können wir überprüfen ob das Bearbeitungs-Formular abgesendet wurde. Wir holen uns wieder folgende Werte:

- userid: Die userid bekommen wir wieder aus der Session damit wir wissen zu welchem Benutzer die To-Do gehört.
- Zeit: Die Zeit wird mit der date Funktion wieder generiert damit wir wissen wann die To-Do bearbeitet wurde.
- Titel & Inhalt: Die neuen Werte für Titel und Inhalt holen wir aus den input Feldern vom Formular und benutzen trim um unnötige Leerzeichen zu entfernen. Danach erstellen wir wieder einen try-catch Block wie beim erstellen. Im try Teil haben wir diesmal aber einen UPDATE Befehl statt INSERT INTO benutzt, damit ändern wir die vorhandene To-Do in der Datenbank. Wir setzen den Titel, Inhalt und die aktuelle Zeit. Mit WHERE überprüfen wir, dass die To-Do zum richtigen Benutzer gehört und die richtige todoid hat. Mit bindParam haben wir dann wieder die Parameter an die Variabeln gebunden und mit execute den Befehl ausgeführt. Falls ein Fehler auftritt fängt der catch-Block diesen und zeigt eine Fehlermeldung an.

Das Bearbeiten funktioniert so nicht weil das wie bei Notizen anfangs, weil es keine Input felder sind also haben wir es wieder auf eine neue Seite gemacht die todo.edit.php heißt und haben dort alle Todos angezeigt (nicht nach Status sortiert) wo die Überschrift und der Inhalt der Input Felder ist und haben bei dem neuen Todo und noch einen Knopf hinzugefügt der uns dort hinleitet.

To-Do Funktionalität: To-Do Löschen (Status 4)

Nachdem wir nun die Funktionen zum Erstellen und Bearbeiten von To-Dos haben, benötigen wir auch eine Funktion zum Löschen. Diese Funktion erstellen wir mit dem Namen todoLoeschen und übergeben wieder die $db Variable, um auf die Datenbank zugreifen zu können. Wie bei den vorherigen Funktionen überprüfen wir zuerst mit einer if-Verzweigung, ob die REQUEST_METHOD auf POST gesetzt ist und ob der Knopf todo_loeschen gedrückt wurde. Dadurch stellen wir sicher, dass das Löschformular abgesendet wurde. Wir holen uns wieder folgende Werte:

- userid: Die userid entnehmen wir erneut der Session, um zu wissen, zu welchem Benutzer die To-Do gehört.
- todoid: Die todoid holen wir aus dem Formular, um zu wissen, welche To-Do gelöscht werden soll.

Dann erstellen wir einen try-catch-Block. Im try-Block verwenden wir diesmal einen UPDATE Befehl, um den todo_status auf 4 zu setzen. Dies bedeutet, dass die To-Do in den Papierkorb verschoben wird. Mit WHERE stellen wir sicher, dass die To-Do zum richtigen Benutzer gehört und die korrekte todoid hat. Wir binden die Parameter mit bindParam und führen das Statement mit execute aus. Falls ein Fehler auftritt, fängt der catch-Block diesen und gibt eine Fehlermeldung aus.

To-Do Funktionalität: To-Do Status ändern

Nachdem wir nun die Funktionen zum Erstellen, Bearbeiten und Löschen von To-Dos haben, benötigen wir auch eine Funktion zum Ändern des Status. Diese Funktion erstellen wir mit dem Namen todoStatus und übergeben wieder die $db Variable, um auf die Datenbank zugreifen zu können. Wie bei den vorherigen Funktionen überprüfen wir zuerst mit einer if-Verzweigung, ob die REQUEST_METHOD auf POST gesetzt ist und ob der Knopf todo_status gedrückt wurde. Dadurch stellen wir sicher, dass das Statusänderungs-Formular abgesendet wurde. Wir holen uns wieder folgende Werte:

- userid und todoid: Bei diesen Werten ist wieder das gleiche wie bei den Anderen Funktionen.
- Status: Den neuen Status holen wir ebenfalls aus dem Formular, um zu wissen, welcher Status gesetzt werden soll.

Dann erstellen wir wieder einen try-catch Block. Im try Block verwenden wir einen UPDATE Befehl, um den todo_status auf den neuen Wert zu setzen. Where benutzen wir wieder wie bei der anderen Funktion.

To-Do Funktionalität: To-Do Löschen (Status 5)

Um To-Do Elemente Permanent aus dem Papierkorb zu löschen haben wir uns überlegt einfach dessen Status auf 5 zu Setzen, dies machen wir einfach indem wir in das sql Statement den Status auf 5 Updaten somit wir die To-Do Elemente noch in der Datenbank haben aber sie nicht angezeigt werden weil wir nur vorhaben die Statuswerte 1-4 anzuzeigen in der “Tabelle“.

To-Do Anzeigen der To-Do Listen

Jetzt da wir alle PHP functions fertig haben können wir mit HTML CSS (und PHP um die Variablen auszugeben) Anfangen.
Zuerst starten wir HTML geben der Seite einen Titel und fügen die Navigationsleiste wieder ein.
Dann machen wir eine form, dort machen wir eine Überschrift ein Input Feld eine Textarea und ein Button um die form abzuschicken.

Event Timer Planung

Für unseren Event Timer haben wir uns überlegt was ähnliches wie die To-Do Liste zu machen d. h. ein Feld mit Textarea und normalen Input Feld zu machen wo man wie bei der To-Do List einen Titel und Inhalt angeben kann. Aber für unseren Event Timer haben wir uns überlegt noch einen Feld hinzuzufügen wo man die Deadline des Events angeben kann. Wir haben uns überlegt wie die Events geordnet sein sollen. Wir haben uns dann darauf beschlossen, dass das neueste Event ganz oben angezeigt werden soll. Wir wollten dann auch, dass man Events bearbeiten und löschen kann, weswegen wir 2 Knöpfe für das Bearbeiten und das Löschen eines Events hinzugefügt haben.

Event Timer Funktionalität: Event Timer Erstellung

Für den Event Timer haben wir keine Funktion erstellt. Wir haben für den Event Timer ganz einfach den try-catch-Block verwendet. Als Parameter übergeben wir wie bei der To-Do Liste und bei den Notizen die $db Variabel. Wir haben wieder eine if-Verzweigung erstellt wie bei der To-Do Liste. Man sieht, bei dem Event Timer müssten wir nicht wirklich vieles neu machen, da es sich von der To-Do Liste sehr ähnelt. Nachdem wir das Formular gecheckt haben brauchen wir jetzt die Werte die wir in die Datenbanktabelle „ereignisscountdown“ eintragen wollen:

- userid: Beim Hinzufügen, Bearbeiten oder Löschen eines Events wird die userid als Referenz genutzt, um sicherzustellen, dass das Event einem bestimmten Nutzer gehört.
- ecd_name: Beim Hinzufügen eines neuen Events wird ecd_name aus dem Formular entnommen und in die Datenbank eingefügt.
- ecd_datum: Beim Hinzufügen eines Events wird das eingegebene Datum in die Datenbank gespeichert.
- ecd_inhalt: Beim Hinzufügen eines Events wird die Beschreibung in die Datenbank gespeichert.

Wie in der To-Do Liste sowie in den Notizen wird im try-Block mit prepare einen INSERT INTO gemacht wo wir alle Werte in die ereignisscountdown Tabelle einfügen und mit Hilfe von bindParam verbinden wir die Platzhalter im SQL-Statement mit PHP-Variablen, um Eingaben sicher in die Datenbank einzufügen.

Event Timer Funktionalität: Event Timer Bearbeitung

Wir haben für das Bearbeiten wieder mit dem Try-Catch-Block gearbeitet und haben wieder $db als Variabel für Übergabeparameter verwendet. Wie beim Erstellen haben wir wieder eine if-Verzweigung erstellt. Damit überprüfen wir, ob das Formular zum Bearbeiten eines Events abgeschickt wurde. Für die Bearbeitung holen wir uns folgende Werte wieder:

- userid: Beim Hinzufügen, Bearbeiten oder Löschen eines Events wird die userid als Referenz genutzt, um sicherzustellen, dass das Event einem bestimmten Nutzer gehört.
- ecdid: Beim Bearbeiten eines Events wird die ecdid verwendet, um genau das richtige Event in der Datenbank zu aktualisieren.
- ecd_name: Beim Bearbeiten eines bestehenden Events wird ecd_name aktualisiert und das geänderte Event in der Datenbank gespeichert.
- ecd_datum: Beim Bearbeiten eines Events kann das Datum ebenfalls geändert werden.
- ecd_inhalt: Beim Bearbeiten eines Events kann die Beschreibung ebenfalls geändert werden.

Danach erstellen wir wieder ein Try-Catch-Block wie bei der Erstellung. Wir benutzen aber diesmal keinen INSERT INTO Befehl sondern einen UPDATE Befehl, so ändern wir den schon vorhandenen Event in der Datenbank. Dann setzen wir natürlich noch den Namen, Datum und Inhalt. Mit WHERE machen wir dasselbe wie bei der To-Do Liste, damit überprüfen wir, dass das Event zu dem richtigen Benutzer gehört und auch die richtige ecdid hat.

Event Timer Funktionalität: Event Timer Löschen

Nachdem wir alles für das Erstellen und Bearbeiten eines Events erledigt haben, brauchen wir noch das Löschen eines Events. Das haben wir wie bei der Bearbeitung und Erstellung mit einem Try-Catch-Block gemacht. Wie vorher überprüfen wir mit einer if-Verzweigung, ob das Formular zum Löschen eines Events abgeschickt wurde. Für das Löschen haben wir nur zwei Werte gebraucht und diese sind:

- ecdid: Beim Löschen eines Events wird die ecdid verwendet, um genau das richtige Event in der Datenbank zu löschen.
- userid: Beim Hinzufügen, Bearbeiten oder Löschen eines Events wird die userid als Referenz genutzt, um sicherzustellen, dass das Event einem bestimmten Nutzer gehört.

Wir erstellen dann wieder einen Try-Catch-Block. Wir benutzen für das Löschen die SQL-Abfrage DELETE FROM. Sie löscht das Event aus der Tabelle ereignisscountdown. Die ecdid-Bedingung stellt sicher, dass nur das bestimmte Event gelöscht wird, das der Nutzer ausgewählt hat. Die userid-Bedingung stellt sicher, dass nur das Event des eingeloggten Nutzer gelöscht wird.

Event Timer Funktionalität: Event Timer Anzeigen

Um jetzt alles Anzeigen zu lassen haben wir eine Datenbankabfrage machen müssen. Das haben wir diesmal nur mit einem Try-Catch-Block gemacht. Was wir hierfür gebraucht haben war nur ein Wert:

- userid: Beim Hinzufügen, Bearbeiten oder Löschen eines Events wird die userid als Referenz genutzt, um sicherzustellen, dass das Event einem bestimmten Nutzer gehört.

Jetzt erstellen wir wieder einen Try-Catch-Block. Wir vorbereiten wieder die SQL-Abfrage SELECT FROM. Dieser holt alle Events des Nutzers, dessen userid mit dem Platzhalter :userid übereinstimmt und sortiert sie nach dem Datum mit ecd_datum.

Event Timer Funktionalität: HTML

Jetzt da wir den Backend abgeschlossen haben, kommen wir zum Frontend. Wir haben mit dem übernehmen der Navigationsleiste angefangen. Danach haben wir die Input Felder für den Namen des Events, das Datum des Events und den Inhalt des Events hinzugefügt, und dabei noch einen „Hinzufügen“ Button um das Event hinzuzufügen. Die Events werden in einer ungeordneten Liste angezeigt. Danach überprüfen wir mit einer if-Verzweigung ob die Variable $events nicht leer ist. Ist die Variable nicht leer dann durchläuft die Schleife foreach jedes Event. Jedes Event wird in einer Liste dargestellt. Innerhalb jeder Liste werden die Details des Events dargestellt. Event-Name, Beschreibung und Datum werden direkt aus dem Array $event angezeigt, indem die entsprechenden Werte per PHP eingefügt werden. Dann brauchen wir noch eine Countdown-Berechnung. Dies berechnet die verbleibende Zeit bis zum Eventdatum. Das Eventdatum wird in ein DateTime-Objekt umgewandelt und mit dem aktuellen Datum verglichen. Dann noch eine Countdown-Anzeige welcher die verbleibende Zeit in Tagen und Stunden ausgibt. Dann haben wir ein Bearbeiten- und Lösch-Formular erstellt. Jedes Event enthält solch ein Formular. Das Formular zeigt vorab die aktuellen Eventwerte die der Nutzer ändern kann. Dabei kommen noch die Buttons Löschen und Bearbeiten.

Auslog Button

Wir haben am Ende unseres Projektes noch ein Auslogg button gemacht, diesen Knopf haben wir in die nav nach rechts gemacht und ihm ein Icon von icons8 gegeben, dieser Knopf führt wenn er gedrückt wird ein PHP code aus der die Session destroyed und somit den user ausloggt.

Zusammenfassung

Was hat geklappt

Die Sachen die funktioniert haben waren die Notizen, die To-Do Liste, der Event Timer, das Login und die Registrierung das Dokumentieren unserer Arbeit, den Auslog Knopf die Datenbank das Design Relativ Einheitlich zu halten.

Was hat nicht geklappt / Was könnte man besser machen

Die Einhaltung des Zeitplanes hat nicht richtig Funktioniert da aufgaben aufgeschoben wurden durch Private Angelegenheiten/Üben von Klassenarbeiten weshalb zwischen Milestone 2-3 nicht viel an der Projektarbeit gearbeitet wurde und dann vor Abgabe sehr viel Arbeit noch zu tun war.

Neue Ideen / Erweiterungen die nicht realisiert wurden

Es gibt keinen weg Notizen etc. mit anderen Benutzern zu Teilen. Die Seite ist nicht Responsive.

English Summary

The project we developed is called "Multitool Workspace," a versatile digital platform designed to help users stay organized and manage various aspects of their daily tasks. This workspace combines three essential productivity functions: a note storage system, a to-do list, and an event timer. By integrating these tools, the workspace offers a seamless and centralized solution for users to enhance their efficiency, manage their schedules, and keep track of important information. One of the key features of the Multitool Workspace is the note storage system, which allows users to create, store, and manage their notes easily. Whether it's jotting down quick reminders, brainstorming ideas, or saving important pieces of information, users can access their notes anytime they need. The notes are stored securely and can be organized for easy retrieval, ensuring that users never lose track of their thoughts and ideas. The second component is the to-do list, which plays a crucial role in task management. This feature enables users to create and manage task lists, helping them prioritize their daily activities. Users can add, edit, and delete tasks, set due dates, and mark tasks as complete once they have been accomplished. By having a clear overview of their pending and completed tasks, users can stay on top of their responsibilities and achieve their goals more effectively. The event timer is the third important functionality in the Multitool Workspace. This tool is designed to help users keep track of deadlines, appointments, and other time-sensitive events. Users can set timers or reminders for specific events, ensuring that they are notified when important moments or deadlines approach. This feature is especially useful for users who want to manage their time efficiently and avoid missing crucial events. In addition to these core functionalities, the Multitool Workspace includes an account system. Users can create their own accounts, log in securely, and connect to their personalized workspace. This feature provides each user with a private space to store their notes, tasks, and events, while also enabling them to access their information from different devices. The technical foundation of the Multitool Workspace includes a connection to an SQLite database. This database plays a vital role in storing user data, including notes, task lists, and event timers. By using SQLite, the workspace ensures that data is managed efficiently, securely, and in a structured manner. The database allows for quick data retrieval and updates, contributing to a smooth and responsive user experience. Overall, the Multitool Workspace aims to be a reliable and user-friendly platform for individuals who want to improve their productivity and organization. By bringing together note-taking, task management, and event tracking in a single digital space, the workspace simplifies everyday life and empowers users to manage their time and tasks with ease. Its intuitive interface, secure account system, and efficient database integration make it a practical tool for students, professionals, and anyone seeking to stay organized in today's fast-paced world.

Quellen Benutzer Software

Visual Studio Code - https://code.visualstudio.com/
xampp - https://www.apachefriends.org/
Sqlite Browser - https://sqlitebrowser.org/
Libre Office - https://www.libreoffice.org/
Coolors - https://coolors.co/
Homebutton Icon - https://icons8.de/icon/2797/home
Auslog Icon - https://icons8.de/icon/kl7ydc3xhWkZ/locked-outside

Quellen Hilfsmittel

W3schools - https://www.w3schools.com/
PHP Manual - https://www.php.net/manual/en/index.php
Chatgpt - https://chatgpt.com/
Youtube – https://youtube.de/
Angeschaute Videos:

Titel	**Link**
Kurz & Knapp #001: Try, Catch, Finally in PHP (deutsch)	https://youtu.be/7NLqdkpKVjk?si=cRK16ugZC2UFLDUG
SQL-Grundlagen: Die wichtigsten Befehle	https://www.youtube.com/watch?v=xMBz6i2Tv1Y
How to install DB Browser for SQLite on Windows 10/11 [2024 Update] Create Database, Table in SQL	https://www.youtube.com/watch?v=o8d38cGb7vY

Einverständniserklärung

Die vorliegende Projektarbeit haben wir selbständig und nur mit den abgegebenen Hilfsmitteln angefertigt. Alle Stellen, die dem Sinn oder dem Wortlaut nach anderen Werken entnommen sind, wurden durch Angabe der Quellen kenntlich gemacht. Alle wörtlich entnommenen Stellen sind als Zitate gekennzeichnet.

Mannheim den _____ Unterschrift Timo:_____

Mannheim den _____ Unterschrift Lukas:_____