Projection plan: When it came to doing my project the base of objective was simple. To make a website like Spotify of YouTube but to be dedicated to Meditation. It was easy to imagine since there are many platforms that could be used like a mediation platform, personally I liked YouTube music. I decided to focus on a platform that does not require log in information. (I dislike websites like it does) But log-in and sing up's is something I plan to implement (I want to learn do things). I need to learn more about meditation since I don't meditate, I plan to add a few tracks at least the basic I order to help people regain focus or are able to sleep. I also plan on adding a feature of people being able to rate their meditation system. I am interested in a platform that collects data, builds and expands databases.

Objectives

- Create a web-based meditation platform similar to Spotify or YouTube but dedicated solely to meditation.
- Provide users with easy access to meditation tracks without requiring login information at first.
- Later, implement login and sign-up features to enhance personalization and learning experience.
- Include basic meditation tracks that help users regain focus, relax, or fall asleep.
- Add a rating feature for users to share feedback on each meditation track.
- Focus on learning how to collect and manage user data, expand databases, and improve backend functionality.

- Gain more understanding of meditation practices to build a meaningful and helpful platform.

1. Project Setup and Database Design

The first part of my project was to create the basic structure of what I wanted to build. I started with the **database schema**, which included the following tables:

- **Categories** – for organizing meditations by mood.
- **Users** – for storing user information.
- **Meditations** – for listing the available meditation tracks.
- **Sessions** – for storing each user's recorded sessions.

After setting up the database, I inserted **sample data** using generic information. I didn't focus much on this part because I wanted to prioritize learning and improving my understanding of **PHP** and **XAMPP**, which were the parts I was least familiar with.

---

2. XAMPP Configuration and Challenges

One of my main challenges was learning how to operate **XAMPP**, since it was completely new to me. I had issues with **port usage** — specifically, port **3306** was already being used by another program, which prevented MySQL from running properly for several days.

I spent my first few sessions learning how to configure ports, make changes, and troubleshoot the issue. Even after turning off the MySQL80 service, I was still unable to use XAMPP's MySQL, even after reviewing the recordings for guidance.

To fix the problem, I eventually **uninstalled and reinstalled XAMPP** with administrator permissions, restarted my computer, and reopened the software with admin privileges. After reconfiguring ports, updating the settings, and turning off MySQL80 completely, I was finally able to access **phpMyAdmin** and upload my database successfully.

---

3. Main Page – index.php

The next step was working on the PHP section, starting with the main page, **index.php**. This file serves as the **homepage** of the meditation website. It shows all the meditation tracks stored in the database and allows users to choose one to play.

This page also checks if a user is logged in or not. I used:

```
session_start();
```

This function keeps track of the user's session and remembers if they are logged in across different pages.

Then I used:

```
include('db_connect.php');
```

This connects the website to the MySQL database using the $conn variable.

After connecting, I added:

```
$sql = "SELECT * FROM meditations";

$result = mysqli_query($conn, $sql);
```

This selects all rows from the meditations table. The mysqli_query() function sends commands to the database and stores the results in $result.

The next part of the page mixes **HTML and PHP**. It displays the title, a welcome message, and a list of meditations.
An **if statement** checks if the user is logged in:

```
if (isset($_SESSION['username'])) {
  echo "<p>Welcome, " . $_SESSION['username'] . "! <a href='logout.php'>Logout</a></p>";
} else {
  echo "<a href='login.php'>Login</a> | <a href='register.php'>Register</a>";
}
```

If the user is logged in, it shows their name and a logout link. If not, it shows links to log in or register.

The next section loops through $result and displays each meditation **title**, **description**, and a **Play** button that links to another page. If no meditation is found, it displays a message for troubleshooting. Finally, the connection is closed using:

```
mysqli_close($conn);
```

---

4. Database Connection – db_connect.php

Before moving forward, it's important to explain db_connect.php, since it's the foundation of the website's database connection. This file links all other PHP files to the database (except **logout.php**, which doesn't need it).

It contains the following information:

```
$server = "localhost";
```

$user = "root";

$pass = "";

$db = "meditation_platform";

$conn = mysqli_connect($server, $user, $pass, $db);

- localhost – the local server (my computer).
- root – the default MySQL username for XAMPP.
- "" – the password field, which I left empty.
- meditation_platform – the name of the database.

If the connection fails, the following line shows an error:

```
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
```

This file is included in all other PHP scripts to make sure they can access the database.

---

5. Meditation Player – play.php

The next file is play.php, which plays a **selected meditation** from the database. When a user clicks on a meditation from the main page (index.php), this page opens and shows the **title**, **description**, **instructor**, and **audio player**.

It begins with:

```
session_start();

include('db_connect.php');
```

Then:

```
$id = $_GET['id'];

$sql = "SELECT * FROM meditations WHERE meditation_id = $id";

$result = mysqli_query($conn, $sql);

$row = mysqli_fetch_assoc($result);
```

This retrieves the meditation that matches the selected ID.
The information is displayed using:

```
<h1><?php echo $row['title']; ?></h1>

<p><?php echo $row['description']; ?></p>
```

Instructor: <?php echo $row['instructor']; ?>

and the audio is played with:

<audio controls>

<source src="<?php echo $row['media_url']; ?>" type="audio/mpeg">

</audio>

The media_url column points to the meditation file stored in the assets folder.

If the user is logged in, the page shows a form that allows them to record their session — entering duration, mood, and notes. The form sends this data to **record_session.php**. If not logged in, a message appears asking them to log in first. Finally, the connection is closed with mysqli_close($conn);.

---

6. Recording Sessions – record_session.php

This file records and saves the meditation session details for a logged-in user. It begins with:

session_start();

include('db_connect.php');

If the user is logged in:

if (isset($_SESSION['user_id'])) {

$user_id = $_SESSION['user_id'];

$meditation_id = $_POST['meditation_id'];

$duration = $_POST['duration_seconds'];

$mood = $_POST['mood'];

$notes = $_POST['notes'];

These values are taken from the form in play.php. Then the code adds a new record to the sessions table:

$sql = "INSERT INTO sessions (user_id, meditation_id, started_at, duration_seconds, mood, notes)

VALUES ('$user_id', '$meditation_id', NOW(), '$duration', '$mood', '$notes')";

If the data is inserted correctly, a message appears saying:

Session recorded successfully.

If something goes wrong, an error message appears.
If the user is not logged in, the page shows:

You must be logged in to record a session.

Finally, the connection to the database is closed.

---

7. User Registration – register.php

The register.php file allows new users to create an account. It starts by including the database connection and setting an empty message variable:

```
include('db_connect.php');

$message = "";
```

When the form is submitted:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  $username = $_POST['username'];
  $email = $_POST['email'];
  $password = $_POST['password'];
```

The user's information is then inserted into the users table. If registration is successful, a success message appears with a link to the login page. If there's an error, it displays an error message. The HTML form on the page provides a simple front-end for users to enter their data.

---

8. User Login – login.php

The **login.php** file allows existing users to sign into the meditation platform. It checks their email and password against the database.

It starts with:

```
session_start();

include('db_connect.php');

$message = "";
```

When the user submits the login form, the script runs:

```
$sql = "SELECT * FROM users WHERE email='$email' AND password_hash='$password'";

$result = mysqli_query($conn, $sql);
```

If one match is found:

```
$_SESSION['user_id'] = $row['user_id'];

$_SESSION['username'] = $row['username'];

header("Location: index.php");
```

If no match is found, it displays “Invalid email or password.”
Once logged in, the user is redirected to the homepage.

---

9. User Logout – logout.php

The **logout.php** file logs the user out of the meditation platform. It ends the session, clears the stored login information, and redirects the user back to the login page.

It works with three simple lines:

```
session_start();

session_destroy();

header('Location: login.php');

exit;
```

This ensures that the user’s session data is cleared and that the website remains secure.

---

10. CSS

Created a file type style.css and created a basic appealing layout and added to the .php

files: `<link rel="stylesheet" href="assets/style.css">`

This code is CSS to style the header:

```
header{  color: #1C4D10;}
```

This code is CSS to style the body:

```
body {  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;   background-color:

#3B3B3B; color: #FFFFFF; margin: 10;  padding: 0;}
```

```
header { background-color: #a7d7c5; padding: 20px; text-align: center;   color: #2e4a3f;
box-shadow: 0 2px 4px rgba(0,0,0,0.1);}
```

This code is CSS to style the sentences:

```
h1{ background-color: #1C4D10; text-align: center;}

h1, h2, h3 { margin: 10px 0;}

.container {max-width: 900px;   margin: 30px auto;background: white; border-radius: 12px;
box-shadow: 0 4px 10px rgba(0,0,0,0.05); padding: 25px;}
```

This code is CSS to style the play button:

```
.play-btn {display: inline-block; background-color: #4CAF50; color: white; padding: 10px
20px; text-decoration: none; border-radius: 8px;font-weight: bold; transition: background-
color 0.3s;}

.play-btn:hover {  background-color: #45a049;}
```

This code is CSS to style the link to other php files:

```
a { color: #4b8073; text-decoration: none;}

a:hover {  text-decoration: underline;}
```

This code is CSS to style the footer:

```
footer {text-align: center;padding: 15px;  background-color: #1C4D10; margin-top: 30px;
color: #FFFFFF;}
```