Note Book Project

1. Directory Structure

```
notebook/
├── index.php
├── login.php
├── register.php
├── logout.php
├── notes/
│   ├── create.php
│   ├── edit.php
│   ├── delete.php
│   └── view.php
├── config/
│   └── db.php
└── templates/
    ├── header.php
    └── footer.php
```

2. Database Schema

Create a database named **notebook_db** and execute the following SQL to create tables:

```
CREATE DATABASE notebook_db;
USE notebook_db;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE notes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    title VARCHAR(100),
    content TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id)
  REFERENCES users(id) ON DELETE CASCADE
);
```

Add a test user (username: admin, password: admin):

```
INSERT INTO users (username, password)
VALUES ('admin', MD5('admin'));
```

Install:

1. Xampp
2. Visual Studio Code
3. GIT

INT AUTO_INCREMENT PRIMARY KEY

- `INT`:
  - Specifies that the column will store integer (whole number) values.
- `AUTO_INCREMENT`:
  - Automatically generates a unique sequential value for the column whenever a new row is inserted into the table.
  - Starts at 1 (by default) and increments by 1 for each new row.
  - Ensures that you don't need to manually specify values for this column when inserting rows.
- `PRIMARY KEY`:
  - Declares this column as the **primary key** of the table.
  - A primary key uniquely identifies each row in the table and enforces uniqueness.
  - It also creates an index on the column to speed up lookups.

FOREIGN KEY (user_id)
REFERENCES users(id) ON DELETE CASCADE

- If a user with `id = 1` is deleted from the `users` table, all posts in the notes table where `user_id = 1` will also be deleted automatically.
- This ensures **referential integrity** by preventing orphaned rows in the notes table.

MD5 (Message-Digest Algorithm 5) is a widely used cryptographic hash function that produces a **128-bit hash value**. It is commonly represented as a 32-character hexadecimal number. MD5 was designed to verify data integrity but is now considered insecure for cryptographic purposes due to vulnerabilities.

Characteristics of MD5:

1. **Input:** Can take any length of input data.
2. **Output:** Always produces a 128-bit (16-byte) fixed-length hash value, regardless of input size.
3. **One-Way Function:** MD5 is designed to be irreversible; you cannot retrieve the original input from the hash.
4. **Deterministic:** The same input will always generate the same hash.

3. Configuration

config/db.php

```
<?php
$servername = "localhost";
$username = "root";  // Replace with your database username
$password = "";  // Replace with your database password
$dbname = "notebook_db";  // Your database name

// Create connection
$conn = mysqli_connect($servername, $username, $password,
$dbname);

// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
?>
```

1. templates/header.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Notebook</title>
            <link      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
  <div class="container mt-4">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
      <a class="navbar-brand" href="#">Notebook</a>
      <div class="collapse navbar-collapse">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item">
            <a class="nav-link" href="index.php">Home</a>
          </li>
          <?php if (!isset($_SESSION['user_id'])): ?>
            <li class="nav-item">
              <a class="nav-link" href="login.php">Login</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="register.php">Register</a>
            </li>
          <?php else: ?>
            <li class="nav-item">
              <a class="nav-link" href="logout.php">Logout</a>
            </li>
          <?php endif; ?>
        </ul>
      </div>
    </nav>
```

2. Template: templates/footer.php

```
</div>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

3. Login: login.php

```
<?php
session_start();
require_once 'config/db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

```
    $username = $_POST['username'];
    $password = MD5($_POST['password']);

    $query = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
    $result = mysqli_query($conn, $query);
    $user = mysqli_fetch_assoc($result);

    if ($user) {
        $_SESSION['user_id'] = $user['id'];
        header('Location: index.php');
    } else {
        echo "Invalid credentials!";
    }
}
?>

<?php include 'templates/header.php'; ?>
<h2>Login</h2>
<form method="POST" class="mt-4">
    <div class="mb-3">
        <label for="username" class="form-label">Username</label>
        <input type="text" name="username" id="username" class="form-control" required>
    </div>
    <div class="mb-3">
        <label for="password" class="form-label">Password</label>
        <input type="password" name="password" id="password" class="form-control" required>
    </div>
    <button type="submit" class="btn btn-primary">Login</button>
</form>
<?php include 'templates/footer.php'; ?>
```

4. Register: register.php

```
<?php
session_start();
require_once 'config/db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
   $username = $_POST['username'];
   $password = MD5($_POST['password']);

   $query = "INSERT INTO users (username, password) VALUES ('$username', '$password')";
   if (mysqli_query($conn, $query)) {
      header('Location: login.php');
   } else {
      echo "Error: " . mysqli_error($conn);
   }
}
?>

<?php include 'templates/header.php'; ?>
<h2>Register</h2>
<form method="POST" class="mt-4">
   <div class="mb-3">
      <label for="username" class="form-label">Username</label>
      <input type="text" name="username" id="username" class="form-control" required>
   </div>
   <div class="mb-3">
      <label for="password" class="form-label">Password</label>
      <input type="password" name="password" id="password" class="form-control" required>
   </div>
   <button type="submit" class="btn btn-primary">Register</button>
</form>
<?php include 'templates/footer.php'; ?>
```

5. Logout: logout.php

```
<?php
session_start();
session_destroy();
header('Location: index.php');
exit;
?>
```

6. Create Notes: notes/create.php

```
<?php
session_start();
require_once '../config/db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
   $title = $_POST['title'];
   $content = $_POST['content'];
   $user_id = $_SESSION['user_id'];

   $query = "INSERT INTO notes (title, content, user_id) VALUES ('$title', '$content', '$user_id')";
   if (mysqli_query($conn, $query)) {
      header('Location: ../index.php');
   } else {
      echo "Error: " . mysqli_error($conn);
   }
}
?>

<?php include '../templates/header.php'; ?>
<h2>Create Note</h2>
<form method="POST" class="mt-4">
   <div class="mb-3">
      <label for="title" class="form-label">Title</label>
      <input type="text" name="title" id="title" class="form-control" required>
   </div>
   <div class="mb-3">
      <label for="content" class="form-label">Content</label>
      <textarea name="content" id="content" class="form-control" required></textarea>
   </div>
   <button type="submit" class="btn btn-primary">Save</button>
</form>
<?php include '../templates/footer.php'; ?>
```

7. Edit Notes: notes/edit.php

```
<?php
session_start();
require_once '../config/db.php';

$note_id = $_GET['id'];
$query = "SELECT * FROM notes WHERE id = '$note_id'";
$result = mysqli_query($conn, $query);
$note = mysqli_fetch_assoc($result);

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
   $title = $_POST['title'];
   $content = $_POST['content'];

   $query = "UPDATE notes SET title = '$title', content = '$content' WHERE id = '$note_id'";
   if (mysqli_query($conn, $query)) {
      header('Location: ../index.php');
   } else {
      echo "Error: " . mysqli_error($conn);
   }
}
?>

<?php include '../templates/header.php'; ?>
<h2>Edit Note</h2>
<form method="POST" class="mt-4">
   <div class="mb-3">
      <label for="title" class="form-label">Title</label>
         <input type="text" name="title" id="title" class="form-control" value="<?php echo $note['title']; ?>"
required>
   </div>
   <div class="mb-3">
      <label for="content" class="form-label">Content</label>
         <textarea name="content" id="content" class="form-control" required><?php echo $note['content'];
?></textarea>
   </div>
   <button type="submit" class="btn btn-primary">Update</button>
</form>
<?php include '../templates/footer.php'; ?>
```

8. Delete Notes: notes/delete.php

```
 <?php
session_start();
require_once '../config/db.php';

$note_id = $_GET['id'];
$query = "DELETE FROM notes WHERE id = '$note_id'";
if (mysqli_query($conn, $query)) {
   header('Location: ../index.php');
} else {
   echo "Error: " . mysqli_error($conn);
}
?>
```

View Notes: notes/view.php

```
<?php
session_start();
require_once '../config/db.php';

$note_id = $_GET['id'];
$query = "SELECT * FROM notes WHERE id = '$note_id'";
$result = mysqli_query($conn, $query);
$note = mysqli_fetch_assoc($result);
?>

<?php include '../templates/header.php'; ?>
<h2><?php echo $note['title']; ?></h2>
<p><?php echo nl2br($note['content']); ?></p>
<?php include '../templates/footer.php'; ?>
```

Index

```
<?php
session_start();
require_once 'config/db.php';
?>

<?php include 'templates/header.php'; ?>
<h1>Welcome to the Notebook</h1>

<?php if (isset($_SESSION['user_id'])): ?>
  <a href="notes/create.php" class="btn btn-primary">Create Note</a>
  <h2>Your Notes</h2>
  <ul class="list-group mt-3">
    <?php
    $user_id = $_SESSION['user_id'];
    $query = "SELECT * FROM notes WHERE user_id = '$user_id'";
    $result = mysqli_query($conn, $query);
    while ($note = mysqli_fetch_assoc($result)) {
      echo "<li class='list-group-item'>
            <a href='notes/view.php?id={$note['id']}'>{$note['title']}</a>
            <a href='notes/edit.php?id={$note['id']}' class='btn btn-warning btn-sm float-end ml-2'>Edit</a>
            <a href='notes/delete.php?id={$note['id']}' class='btn btn-danger btn-sm float-end'>Delete</a>
          </li>";
    }
    ?>
  </ul>
<?php else: ?>
  <p>Please <a href="login.php">login</a> to view your notes.</p>
<?php endif; ?>
<?php include 'templates/footer.php'; ?>
```

Notebook Application Documentation

Project Overview:

The **Notebook Application** is a simple PHP-based web application that allows users to register, log in, and manage their personal notes. It supports the following operations:

- **Create** a new note
- **Edit** an existing note
- **Delete** a note
- **View** a note's details

The app uses MySQLi Procedural for database interactions and Bootstrap for styling. It also includes user authentication with a login and register system.

Technologies Used:

- **PHP**: Server-side scripting language used for application logic.
- **MySQL**: Relational database management system for storing user and note data.
- **Bootstrap**: Front-end framework used for responsive design.
- **MySQLi (Procedural)**: Database connection and query execution.

Folder Structure:

```
notebook/
├── index.php          # Home page with user's notes
├── login.php          # Login page
├── register.php       # Register page
├── logout.php         # Logout logic
├── notes/
│   ├── create.php     # Page to create new notes
│   ├── edit.php       # Page to edit existing notes
│   ├── delete.php     # Logic to delete notes
│   └── view.php       # Page to view the note's details
├── config/
│   └── db.php         # Database connection settings
└── templates/
    ├── header.php     # Common header template
    └── footer.php     # Common footer template
```

Database Structure:

- **users** table:
  - `id` (INT) - Primary key, auto-incremented.
  - `username` (VARCHAR) - User's username.
  - `password` (VARCHAR) - User's password.
- **notes** table:
  - `id` (INT) - Primary key, auto-incremented.
  - `title` (VARCHAR) - Title of the note.
  - `content` (TEXT) - Content of the note.
  - `user_id` (INT) - Foreign key referencing the `users` table to associate notes with a user.

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE notes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    content TEXT NOT NULL,
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

User Stories:

1. User Registration:

- **As a user**, I can create a new account by entering my username and password.
- After registration, I can log in to the application.

2. User Login:

- **As a user**, I can log in using my credentials (username and password).
- Once logged in, I can create, view, edit, and delete notes.

3. Create Note:

- **As a logged-in user**, I can create a new note with a title and content.

4. View Notes:

- **As a logged-in user**, I can see a list of all my notes with the ability to view details.

5. Edit Note:

- **As a logged-in user**, I can edit the title and content of any of my notes.

6. Delete Note:

- **As a logged-in user**, I can delete any of my notes.

7. Logout:

- **As a user**, I can log out of the application.

System Flow:

1. **Login/Register Flow:**
   - User navigates to the login page and enters username and password.
   - If valid, they are redirected to the home page with their list of notes.
   - If user does not have an account, they can register via the register page.
   - After registration, the user can log in.
2. **Create Note Flow:**
   - User clicks "Create Note" on the home page.
   - User enters the title and content for the new note and submits.
   - The note is stored in the database under the user's account.
3. **View Note Flow:**
   - User clicks on a note's title to view its details.
   - The note's details (title, content) are displayed.
4. **Edit Note Flow:**
   - User clicks the "Edit" button next to a note they wish to edit.
   - User updates the title and content, and the note is updated in the database.
5. **Delete Note Flow:**
   - User clicks the "Delete" button next to a note they wish to delete.
   - The note is removed from the database.

Flowchart:

Below is a flowchart illustrating the main operations of the Notebook Application:

```
plaintext
Copy code
 +--------------------+                    +--------------------+
 |  User Registers    |                    |  User Logs In      |
 |                    |------> Login ------|                    |
 +--------------------+                    +--------------------+
           |                                          |
           v                                          v
 +--------------------+                    +--------------------+
 |   User Home Page   |<-------- View ------|  Notes List        |
 | (Create, View, Edit,|   or Edit/Delete   |  Create/Edit/Delete|
 |       Delete)      |                    +--------------------+
 +--------------------+                           |
           |                                       |
           v                                       v
 +--------------------+                +--------------------+
 |    Create Note     |                |   View/Edit/Delete  |
```

```
|  (Add Title, Content)|                |   Note (Update/Delete)|
+----------------------+                +----------------------+
```

Step-by-Step Operations:

1. **User Registration:**
   - **Input**: Username, password.
   - **Process**: User submits the form, and data is inserted into the `users` table.
   - **Output**: Redirect to login page.
2. **User Login:**
   - **Input**: Username, password.
   - **Process**: Validate credentials with the `users` table. If valid, create a session and redirect to home page.
   - **Output**: Home page with list of notes.
3. **Create Note:**
   - **Input**: Note title, content.
   - **Process**: Insert the note into the `notes` table associated with the logged-in user.
   - **Output**: New note appears on the home page.
4. **View/Edit/Delete Note:**
   - **Input**: Note ID.
   - **Process**: Query the `notes` table to fetch the note and display or allow editing.
   - **Output**: Updated note on the home page (for edit/delete) or a detailed view.
5. **User Logout:**
   - **Process**: Destroy the session and redirect to the login page.
   - **Output**: User is logged out and returned to login page.

Conclusion:

This Notebook application is a full-fledged PHP web application with basic user authentication, note management, and CRUD operations. It leverages MySQLi for database handling and Bootstrap for responsive design. The system architecture follows a simple flow from user registration and login to managing personal notes, making it easy for users to interact with their notes efficiently.