**PHP CRUD DOCUMENTATION**

1. **PHP Login Script Documentation**

**Overview**

This document provides a detailed explanation of the PHP login script used in a CRUD (Create, Read, Update, Delete) application. The script handles user authentication by validating login credentials against a database and managing user sessions.

**File Structure**

The login script is part of a larger application with the following structure:

* login.php (current file) - Handles user authentication
* ../index.php - Main application page (redirected to after login)
* ../config/database.php - Database connection configuration
* register.php - User registration page
* ../css/custom.css - Custom styling
* ../database.sql - SQL file for database setup

**Script Flow**

**Session Management**

1. The script begins by initializing a PHP session using session\_start()
2. It checks if a user is already logged in by verifying session variables
3. If already logged in, the user is redirected to the main application page

**Form Processing**

The script processes form submissions through the following steps:

1. Validates that both username and password fields are not empty
2. Sanitizes input data using trim() to remove whitespace
3. Prepares and executes a secure database query using parameterized statements
4. Verifies credentials by checking:
   * If the username exists in the database
   * If the password matches the stored hash using password\_verify()

**Authentication Results**

Depending on the authentication result:

* **Success**: Creates session variables and redirects to the main page
* **Failure**: Displays a generic error message ("Invalid username or password")
* **System Error**: Shows a technical error message

**Security Features**

The script incorporates several security best practices:

1. **Parameterized Queries**: Prevents SQL injection via mysqli\_prepare() and bound parameters
2. **Password Hashing**: Uses PHP's secure password hashing functions
3. **Generic Error Messages**: Avoids revealing whether username or password was incorrect
4. **Input Sanitization**: Removes whitespace and potentially harmful characters
5. **Session Management**: Proper handling of session variables and authentication state

**User Interface**

The login page features:

* A responsive Bootstrap-based layout
* Form validation with visual feedback
* Clear error messages
* Navigation link to registration page
* Mobile-friendly design

**Code Sections**

**PHP Processing Section**

A computer screen shot of a program

AI-generated content may be incorrect.

**HTML Structure Section**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Implementation Requirements**

To implement this login system:

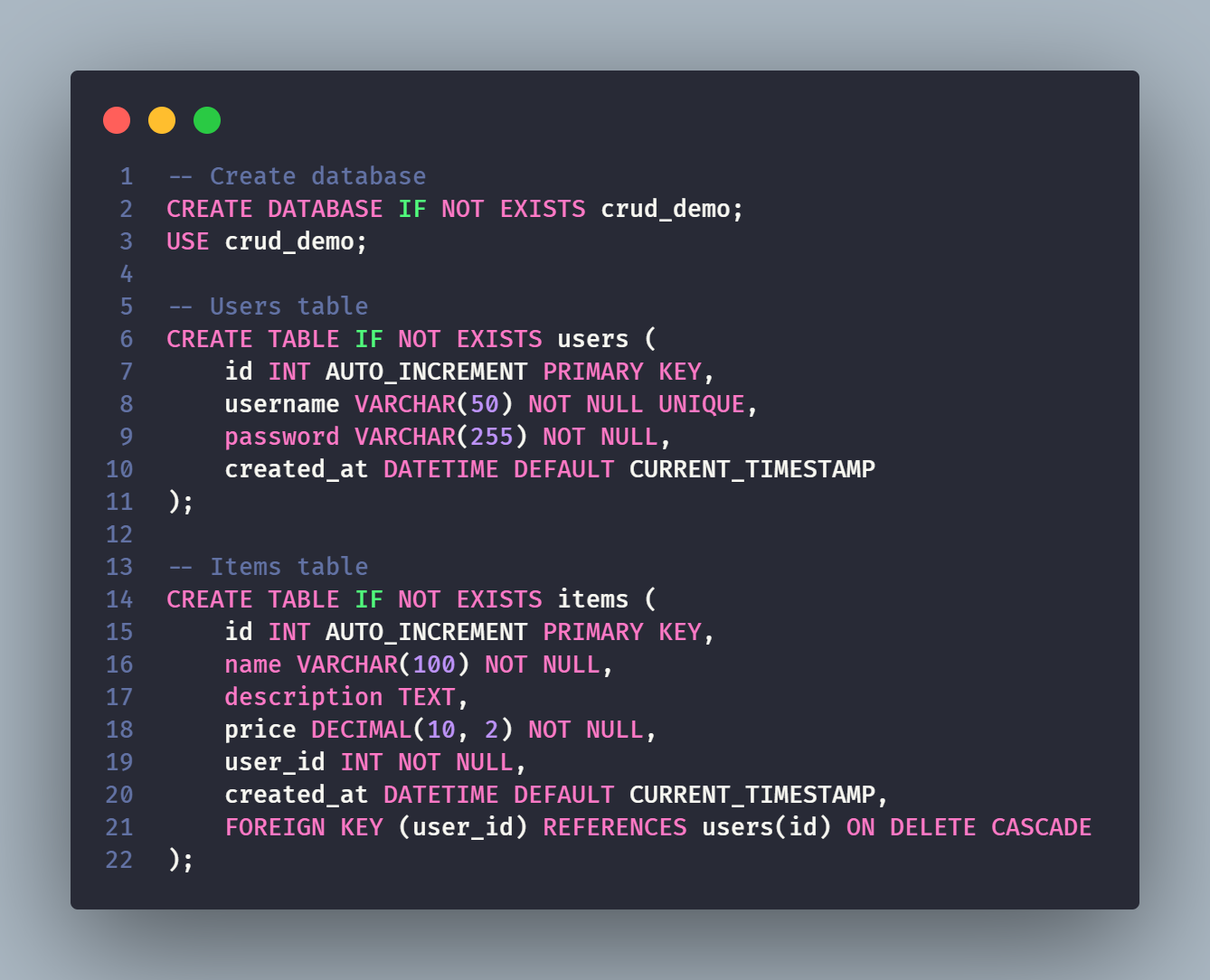
**Method 1:** Manual Database Setup

1. Create a MySQL database with:
   * users table for authentication
   * items table for CRUD operations
2. Configure database connection in ../config/database.php
3. Ensure proper file permissions
4. Configure session settings as needed

**Method 2:** Database Import **(Recommended)**

1. Start XAMPP Services:
   * Launch XAMPP Control Panel
   * Start both Apache and MySQL services
2. Access phpMyAdmin:
   * Open your browser and navigate to: http://localhost/phpmyadmin/
3. Create a New Database:
   * Click on "New" in the left sidebar
   * Enter "crud\_demo" as the database name
   * Click "Create"
4. Import the Database Structure:
   * Select the newly created "crud\_demo" database from the left sidebar
   * Click the "Import" tab at the top
   * Click "Choose File" and select the database.sql file
   * Leave the default settings as they are
   * Scroll down and click "Import" button

**Database SQL File**

****

The database structure includes:

1. users table - Stores user authentication information:
   * id: Unique identifier for each user
   * username: Unique username for login
   * password: Securely hashed password
   * created\_at: Timestamp of account creation
2. items table - Stores items for the CRUD application:
   * id: Unique identifier for each item
   * name: Item name
   * description: Detailed item description
   * price: Item price with decimal precision
   * user\_id: Foreign key linking to user who created the item
   * created\_at: Timestamp of item creation
   * Foreign key constraint ensuring that when a user is deleted, their items are also removed
3. **PHP Logout Script Documentation**

**Overview**

This document provides detailed documentation for the PHP logout script that properly terminates user sessions in a secure manner. The script is part of the user authentication system for the CRUD application and works in conjunction with the login script.

**File Purpose**

The logout.php script is responsible for:

* Terminating active user sessions
* Clearing session data
* Redirecting users to the login page
* Preventing unauthorized access to protected pages after logout

**Script Analysis**

**Code Breakdown**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Execution Flow**

1. **Session Initialization**: The script starts by initializing the PHP session with session\_start(). This is necessary to access and modify session variables.
2. **Session Variable Clearing**: All session variables are unset by setting the $\_SESSION superglobal to an empty array. This removes all user data from the session.
3. **Session Destruction**: The session\_destroy() function is called to completely remove the session from the server.
4. **Redirection**: The user is redirected to the login page using the header() function.
5. **Script Termination**: The exit statement ensures no further code is executed after the redirection header is sent.

**Security Considerations**

The logout script implements several security best practices:

1. **Complete Session Clearing**: Both unsets all session variables and destroys the session completely, ensuring no residual data remains.
2. **Proper Redirection**: Immediately redirects the user to the login page, preventing them from accessing protected content after logout.
3. **Script Termination**: Uses exit to prevent any further code execution after the redirect header is sent.

**Implementation Notes**

**Placement**

The logout script should be accessible from all protected pages, typically through a logout button or link.

1. **PHP Registration Script Documentation**

**Overview**

This document provides detailed documentation for the PHP registration script used in the CRUD application. The script handles user registration, including form validation, username availability checking, password security, and account creation.

**File Purpose**

The register.php script manages the user registration process by:

* Validating user input
* Checking for existing usernames
* Enforcing password requirements
* Securely hashing passwords
* Creating new user accounts in the database
* Providing feedback on validation errors
* Redirecting to login after successful registration

**Script Analysis**

**Code Structure**

The script is organized into two main sections:

1. PHP processing section for server-side validation and database operations
2. HTML section for rendering the registration form with Bootstrap styling

**Form Validation Process**

The registration script performs the following validation checks:

1. **Username Validation**:
   * Verifies the username is not empty
   * Checks database to ensure the username is unique
   * Sanitizes input using trim() to remove whitespace
2. **Password Validation**:
   * Ensures password is not empty
   * Enforces minimum length of 6 characters
   * Sanitizes input using trim()
3. **Password Confirmation**:
   * Verifies confirmation password is not empty
   * Matches confirmation against the original password

**Database Operations**

When validation succeeds, the script:

1. Prepares a parameterized SQL insert statement
2. Securely hashes the password using PHP's password\_hash() function with default algorithm
3. Executes the query to create a new user record
4. Redirects to the login page upon successful registration

**Security Features**

The script implements several security best practices:

1. **Parameterized Queries**: Prevents SQL injection using prepared statements
2. **Password Hashing**: Uses PHP's secure hashing functions
3. **Input Sanitization**: Trims and validates all user inputs
4. **Error Masking**: Displays user-friendly error messages without exposing system details
5. **XSS Prevention**: Uses htmlspecialchars() on form action to prevent cross-site scripting

**Implementation Details**

**PHP Processing Section**

**A computer screen shot of a computer code

AI-generated content may be incorrect.**

**HTML Form Section**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**User Interface**

The registration page features:

* Clean, responsive Bootstrap layout
* Form field validation with visual feedback
* Clear error messages
* Password confirmation field
* Link to login page for existing users
* Cancel button to abort registration

**Integration with Authentication System**

The registration script integrates with other components as follows:

1. **Database**: Creates user records in the same table used by the login script
2. **Login Script**: Redirects to login.php after successful registration
3. **Session Management**: No sessions are created until login occurs

**Using the Registration System**

When implementing this registration system:

1. **Database Requirements**:
   * Must have a users table with columns for:
     + id (auto-increment primary key)
     + username (unique)
     + password (for storing hash)
     + created\_at (optional timestamp)
2. **File Structure**:
   * Registration script should be in the same directory as login.php
   * Database configuration should be accessible via "../config/database.php"
3. **Password Policies**:
   * Current minimum requirement is 6 characters
   * To change password policy, modify the validation logic
4. **PHP CRUD Application: create.php Documentation**

**Overview**

The create.php file is a component of a PHP CRUD (Create, Read, Update, Delete) application that handles the creation of new items. This file combines server-side processing with a user interface form to add items to the database.

**Authentication & Session Management**

The script begins with session management and authentication checks:

A screen shot of a computer program

AI-generated content may be incorrect.

* The session is initialized using session\_start()
* Authentication is verified by checking the session variable $\_SESSION["loggedin"]
* Unauthenticated users are redirected to the login page

**Database Connection**

The script includes the database configuration file:

A computer screen shot of a computer code

AI-generated content may be incorrect.

* The connection to the database is established through the database.php file in the config directory

**Form Processing**

**Variables Initialization**

A screen shot of a computer code

AI-generated content may be incorrect.

* Variables for storing form data ($name, $description, $price)
* Variables for storing validation error messages

**Form Submission Handling**

The script processes form data when submitted via POST method:

A black rectangle with yellow text

AI-generated content may be incorrect.

**Input Validation**

The script validates each form field:

1. **Name Validation**:

A computer screen shot of a code

AI-generated content may be incorrect.

1. **Description Validation**:

A screen shot of a computer code

AI-generated content may be incorrect.

1. **Price Validation**:

A screen shot of a computer code

AI-generated content may be incorrect.

* Checks if price is empty
* Verifies that price is numeric and greater than zero

**Database Insertion**

If validation passes, the script inserts the new item into the database:

A screen shot of a computer

AI-generated content may be incorrect.

* Uses prepared statements to prevent SQL injection
* Associates the new item with the logged-in user via $\_SESSION["id"]
* Redirects to read.php upon successful insertion
* Displays an error message if the execution fails

**User Interface**

The user interface consists of an HTML form with Bootstrap styling:

**Page Structure**

* Navigation bar with links to view items, add items, and logout
* Form container with card styling
* Form fields for name, description, and price
* Submit and cancel buttons

**Form Elements**

A screenshot of a computer program

AI-generated content may be incorrect.

**Form Features**

* Form submits to itself using $\_SERVER["PHP\_SELF"]
* Input validation feedback using Bootstrap's validation classes
* Form maintains input values after submission if there are errors
* Error messages are displayed next to the corresponding fields

1. **PHP CRUD Application: read.php Documentation**

**Overview**

The read.php file is a core component of the PHP CRUD (Create, Read, Update, Delete) application that handles displaying items from the database and processing delete operations. This file combines both server-side processing and a user interface to display items in a tabular format.

**Authentication & Session Management**

The script begins with session management and authentication verification:

A screen shot of a computer program

AI-generated content may be incorrect.

* Session is initialized with session\_start()
* Authentication is verified by checking $\_SESSION["loggedin"]
* Unauthenticated users are redirected to the login page

**Database Connection**

The script includes the database configuration file:

A screen shot of a computer

AI-generated content may be incorrect.

* The connection to the database is established through the database.php file

**Delete Operation**

The script handles item deletion when a delete request is received:

A computer screen shot of a program

AI-generated content may be incorrect.

**Key features:**

* Checks for the presence of a delete parameter in the URL
* Uses prepared statements to prevent SQL injection
* Ensures users can only delete their own items by checking user\_id
* Redirects back to the read page after successful deletion
* Handles error cases with appropriate messages

**User Interface**

The user interface consists of an HTML page with Bootstrap styling:

**Page Structure**

* Navigation bar with links to view items, add items, and logout
* Card container for the item list
* Card header with title and "Add New Item" button
* Table for displaying items (when items exist)
* Information alert when no items are found

**Data Display**

The script fetches and displays items from the database:



**Key features:**

* Uses prepared statements for secure database queries
* Shows items in reverse chronological order (newest first)
* Only displays items belonging to the logged-in user
* Shows a friendly message with a link when no items are found
* Provides error information if the query fails

**Table Generation**

When items exist, the script generates a table to display them:



Table features:

* Responsive table with borders and striping for better readability
* Columns for ID, Name, Description, Price, Creation Date, and Actions
* Security measures with htmlspecialchars() for user-generated content
* Formatted price display with number\_format()
* Formatted date display with date()
* Edit and Delete action buttons for each item
* Confirmation dialog for delete action

1. **PHP CRUD Application: update.php Documentation**

**Overview**

The update.php file is a critical component of the PHP CRUD (Create, Read, Update, Delete) application that handles the updating of existing items in the database. This file manages both the retrieval of current item data and the processing of form submissions to update records.

**Authentication & Session Management**

The script begins with session initialization and authentication verification:

A screen shot of a computer program

AI-generated content may be incorrect.

* Session is initialized with session\_start()
* Authentication is verified by checking $\_SESSION["loggedin"]
* Unauthenticated users are redirected to the login page

**Database Connection**

The script includes the database configuration:

A screen shot of a computer

AI-generated content may be incorrect.

* The connection to the database is established through the database.php file

**Variables Initialization**

A screen shot of a computer code

AI-generated content may be incorrect.

* Variables for storing form data ($name, $description, $price)
* Variables for validation error messages

**Form Processing Logic**

The script has two main execution paths:

1. **POST Request Processing** - When form is submitted
2. **GET Request Processing** - When page is first loaded with an item ID

**POST Request Flow (Form Submission)**

**A computer screen shot of a computer code

AI-generated content may be incorrect.**

Key steps in the POST flow:

1. Validates that an ID was provided in the form submission
2. Performs validation on all form fields
3. Updates the database if validation passes
4. Redirects to read.php on success

**GET Request Flow (Initial Page Load)**

**A screen shot of a computer program

AI-generated content may be incorrect.**

Key steps in the GET flow:

1. Validates that an ID parameter exists in the URL
2. Retrieves the item data from the database
3. Populates the form with existing values
4. Redirects to error.php if ID is invalid or item doesn't exist

**Form Validation**

The script validates each form field when processing a form submission:

A screen shot of a computer program

AI-generated content may be incorrect.

Validation rules:

* Name and description must not be empty
* Price must not be empty, must be numeric, and must be greater than zero

**Database Operations**

**Retrieving Item Data**

When loading the page, the script retrieves the current item data:

A screenshot of a computer program

AI-generated content may be incorrect.

Key features:

* Uses prepared statements for security
* Ensures users can only edit their own items by checking user\_id
* Retrieves all fields of the item record

**Updating Item Data**

After validation, the script updates the item in the database:

A screenshot of a computer program

AI-generated content may be incorrect.

Key features:

* Uses prepared statements to prevent SQL injection
* Updates only name, description, and price fields
* Ensures users can only update their own items with the user\_id condition
* Redirects to read.php on successful update