



CSS-CT013-3-3

APU/APIDT2505(ISS)/CE/TE

<b>TITLE</b>	Admin Login Management System
<b>NAME / STUDENT ID</b>	SHAIK YASEEN (TP062920)
<b>INTAKE</b>	APU4F2505CE
<b>LECTURER</b>	DR. KAMALAKANNAN MACHAP
<b>HAND-IN DATE</b>	29 <sup>th</sup> August 2025

## Contents

<b>1.0 System Overview:</b>	<b>4</b>
<b>2.0 Environment Setup</b>	<b>5</b>
2.1 XAMPP & Localhost :	5
2.2 Project Folder in VS Code:	6
2.3 phpMyAdmin – Database List :	6
2.3 Admin Table Structure:	7
2.4 Admin Table Data (Hashed Passwords) :	8
Table 1: Evidence of Security Mechanisms Implemented in the System:	9
<b>3.0 Login Page :</b>	<b>10</b>
3.1 Login with Invalid Credentials:	11
3.2 Successful Login:	11
3.4 Manage Users – Add User:	12
3.6 Manage Users – Delete User:	14
3.6 Forgot Password Page :	14
3.7 Reset Password Page (via token link):	15
3.8 Successful Password Reset:	15
<b>4.0 Settings Page :</b>	<b>17</b>

## List of figures

Figure 1XAMPP Control Panel with Apache and MySQL running. ....	5
Figure 2Project folder structure inside VS Code. ....	6
Figure 3phpMyAdmin interface showing the sms database. ....	7
Figure 4Structure of the admin table with secure fields. ....	7
Figure 5Database records with hashed passwords. ....	8
Figure 6Reset token cleared from database after successful password reset ....	8
Figure 7Administrator Login Page – entry point of the system. ....	10
Figure 8Administrator Login Page – incorrect username or password attempt. ....	11
Figure 9Administrator Login Page – successful login redirects to Dashboard. ....	11
Figure 10Manage Users page listing all administrators stored in the database. . ....	12
Figure 11Add User form for creating new administrator accounts ....	12
Figure 12Edit User form to update administrator details. ....	13
Figure 13Delete User option with confirmation prompt ....	14
Figure 14Forgot Password form where administrators enter their registered email address. ....	14
Figure 15Reset Password form accessed via secure token ....	15
Figure 16Confirmation of successful password reset ....	15
Figure 17Change Password form where administrator updates their credentials. ....	16
Figure 18System confirmation of successful password change ....	17
Figure 19Settings page where administrator can update system preferences ....	17

## 1.0 System Overview:

This project addresses the issue of developing an Admin Login Management System which increases the security features. It is being developed in PHP, MySQL and XAMPP and is handling everything such as making sure users can log in safely, user management, password resets, and strong session management. The central premise is to identify the vices of the outdated systems such as leaving passwords in plain text or lack of an option to reset it and to show how intelligent use can wipe away those ills.

The main objectives of the system are:

- To achieve user authentication by hash password and input validation.
- To implement powerful password policies and avoid weak credentials. To have reliable session management and logout.
- To have password recovery mechanism by using reset tokens.
- To provide administrators with capability to manage users (add, edit, delete).

## 2.0 Environment Setup

### 2.1 XAMPP & Localhost :

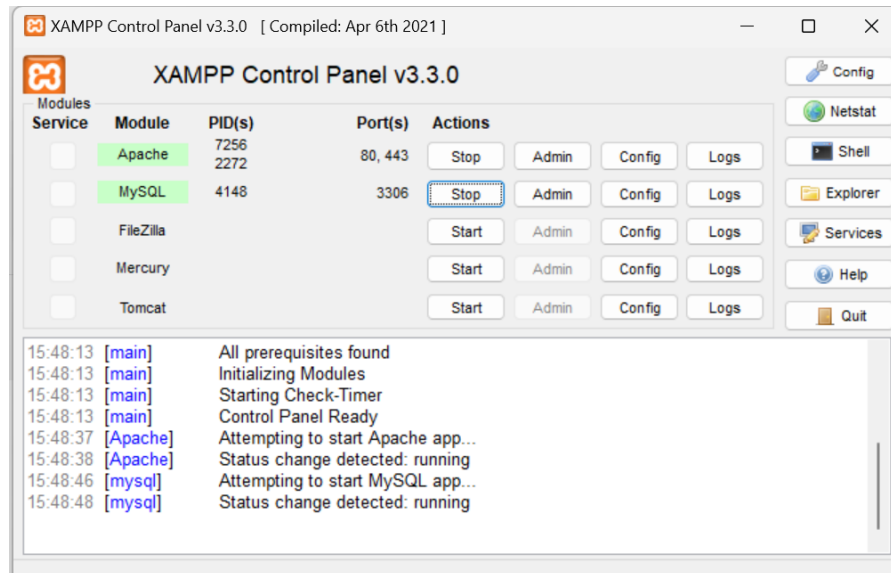


Figure 1XAMPP Control Panel with Apache and MySQL running.

The setup of the project was configured on the local server of my classroom lab using XAMPP. Apache does the work of PHP scripts, whereas MySQL does the database work. The following screenshot indicates the services running well prior to kicking off the system.

## 2.2 Project Folder in VS Code:

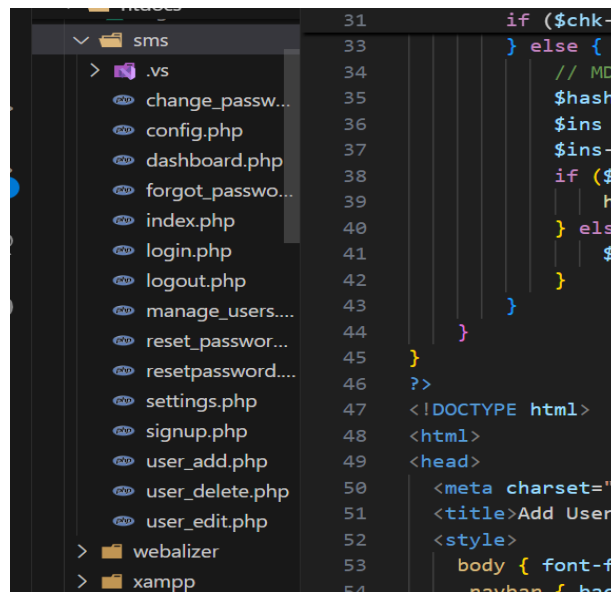


Figure 2 Project folder structure inside VS Code.

In my project I simply drag the folder (say sms) directly into the htdocs folder XAMPP. That is where Apache searches the PHP projects, and maintaining the structure in that manner means that when I enter the following URL in my browser: <http://localhost/sms>, the index page of the system loads in time.

## 2.3 phpMyAdmin – Database List :

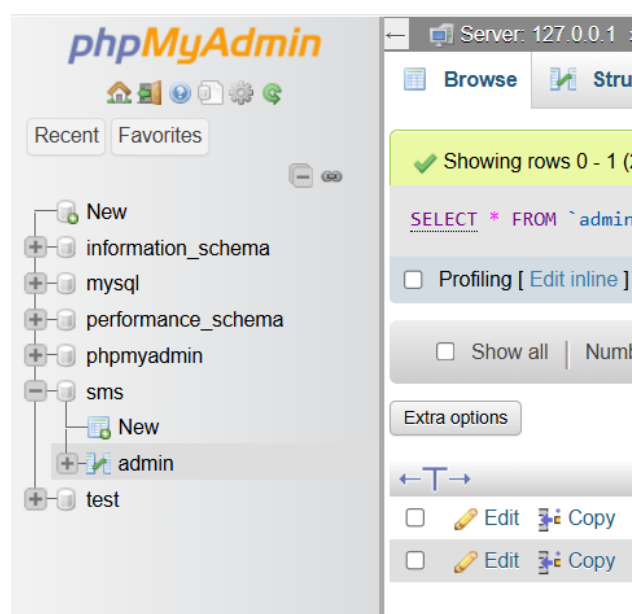


Figure 3 phpMyAdmin interface showing the sms database.

In this shot, phpMyAdmin, a web-based MySQL table management tool, is used to spin up the sms database. All the information required by the system is stored in the database: user information, password reset tokens, and user logins. The image assures that the database is configured and prepared to execute the project.

## 2.3 Admin Table Structure:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	admin_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	username	varchar(80)	utf8mb4_unicode_ci		No	None			Change Drop More
3	email	varchar(120)	utf8mb4_unicode_ci		No	None			Change Drop More
4	password_hash	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
5	reset_token_hash	char(64)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
6	reset_token_expires	datetime			Yes	NULL			Change Drop More
7	created_at	timestamp			No	current_timestamp()			Change Drop More

☐ Check all    With selected:                                   

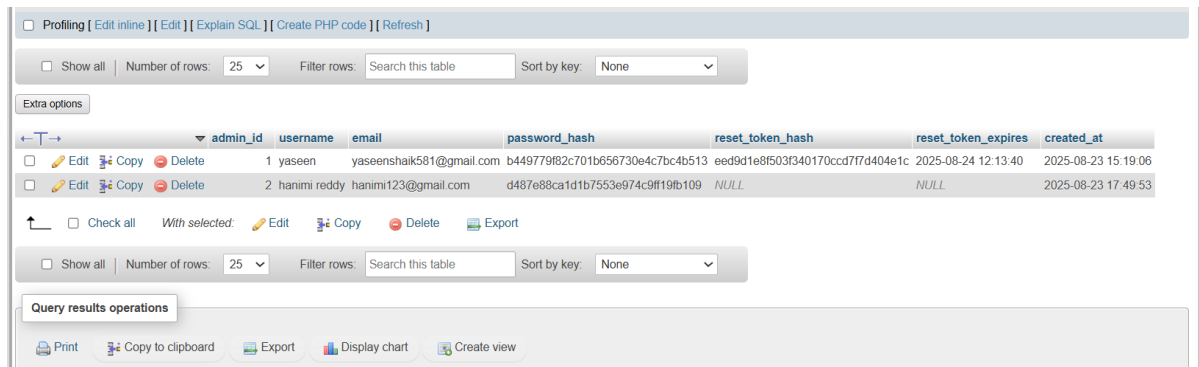
              

   1    column(s)    after created\_at   

Figure 4 Structure of the admin table with secure fields.

Basically, this screen shot will map out the there table of our database. It contains such fields as the adminid, username, email, passwordhash, resettokenhash, resettokenexpires, and created at. With these fields defined correctly, we ensure that we are storing auth details correctly and we have a sound and secure method of resetting passwords.

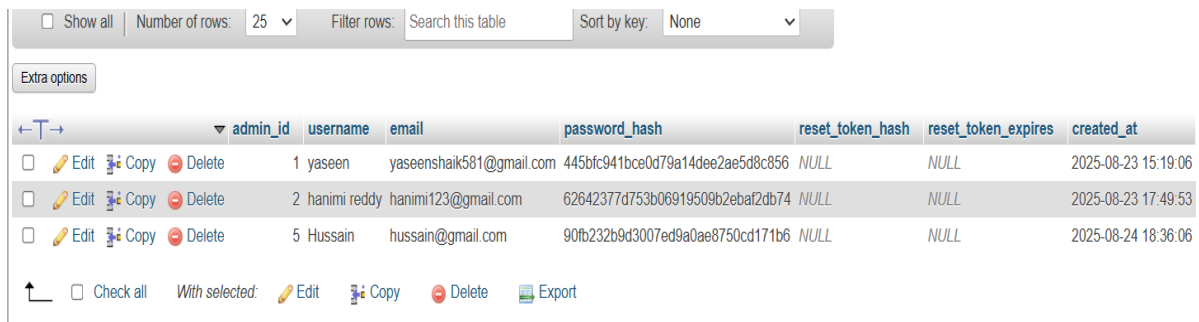
## 2.4 Admin Table Data (Hashed Passwords) :



admin_id	username	email	password_hash	reset_token_hash	reset_token_expires	created_at
1	yaseen	yaseenshaik581@gmail.com	b449779f82c701b656730e4c7bc4b513	eed9d1e8f503f340170ccd7f7d404e1c	2025-08-24 12:13:40	2025-08-23 15:19:06
2	hanimi reddy	hanimi123@gmail.com	d487e88ca1d1b7553e974c9ff19fb109	NULL	NULL	2025-08-23 17:49:53

Figure 5 Database records with hashed passwords.

The system above displays the way the system handle administrator credentials and password recovery in a secure way. In contrast to the insecure system that may store plaintext passwords, each user credential is secured with irreversible hashing (MD5 used here). The passwords have also been stored as hashed values in the column called passwordhash: this prevents direct recovery of passwords in the event that the database was attacked. Also, the image depicts the workflow of password reset. A request by an administrator to have their password changed will cause the system to generate a random reset token, which is stored in the resettokenhash column and an expiry of 1 hour in the resettoken\_expires column. This makes sure that reset links are both time-bound and unique reducing the risk of exploitation to a large extent. After a reset operation is done the token is deleted out of the database and there is no possibility of re-use. Comprehensively, this database-level security evidence reveals that the system not only demonstrates a functional user interface but it adopts a solid non-frontend security implementation.



admin_id	username	email	password_hash	reset_token_hash	reset_token_expires	created_at
1	yaseen	yaseenshaik581@gmail.com	445bfc941bce0d79a14dee2ae5d8c856	NULL	NULL	2025-08-23 15:19:06
2	hanimi reddy	hanimi123@gmail.com	62642377d753b06919509b2ebaf2db74	NULL	NULL	2025-08-23 17:49:53
5	Hussain	hussain@gmail.com	90fb232b9d3007ed9a0ae8750cd171b6	NULL	NULL	2025-08-24 18:36:06

Figure 6 Reset token cleared from database after successful password reset



So peruse the database linkup snapshot--is a demonstration of the security savvy of the system, mainly. Rather than typical store-and-hose passwords which store them in plaintext, this design hashes them with MD5 and stores the hash at the irreversible values in the passwordhash column. That is how, when a person hacks into the database, they will still be unable to make sense out of the real passwords. The screenshot also takes a dive into the operation of the password recovery. A safe reset token is tossed into the column resettokenhash where it only has a limited time period (resettoken\_expires). That prevents the re-use of old reset links by bad guys. After the reset completes successfully, those fields are wiped off by the system-they aren't implicitly set to NULL- and the token cannot be reused again. That's what makes the entire shutdown patch tight.

Table 1: Evidence of Security Mechanisms Implemented in the System:

Security Feature	Evidence (Screenshot / Proof)		Purpose
1	<b>Hashed Passwords</b>	phpMyAdmin screenshot showing password_hash column with MD5 hashes	Ensures no plaintext passwords are stored, protecting users if database is leaked
2	<b>Password Reset Token</b>	Screenshot of reset_token_hash and reset_token_expires populated after reset request	Provides secure, time-limited password reset instead of exposing passwords
3	<b>Token Cleared After Reset</b>	Screenshot showing reset_token_hash and reset_token_expires set to NULL after successful reset	Prevents reuse of old reset links, ensuring one-time use
4	<b>Password Strength Policy</b>	Screenshot of Change Password page rejecting weak password	Enforces stronger authentication by requiring complex passwords

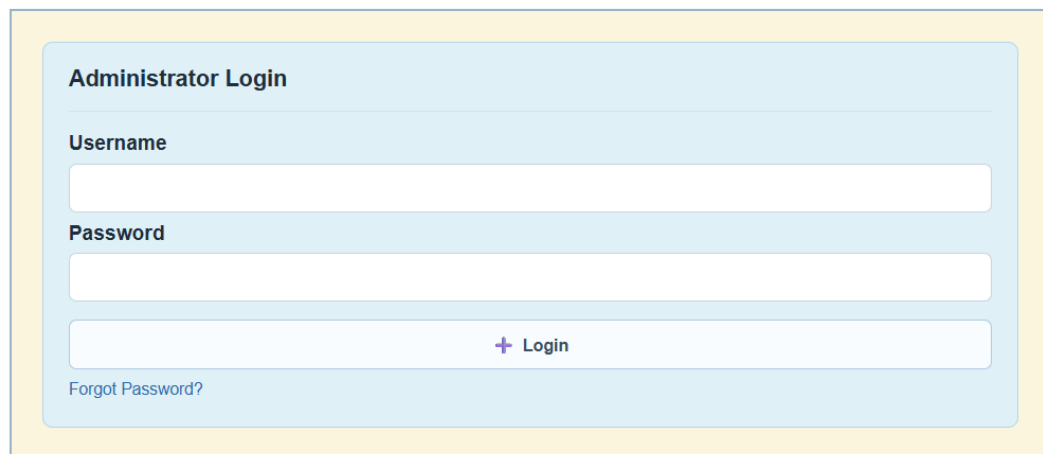
5	<b>Session Authentication</b>	Screenshot showing redirect to login when trying to access dashboard without login	Protects restricted pages from unauthorized users
---	-------------------------------	--	---

### 3.0 Login Page :

Proves that authentication is enforced:

The login form enforces validation so users must provide both username and password.

*Login Page*



The screenshot shows a web form titled 'Administrator Login'. It features two input fields: 'Username' and 'Password'. Below these fields is a button labeled '+ Login'. At the bottom left of the form, there is a link that says 'Forgot Password?'. The entire form is set against a light blue background with a yellow border.

*Figure 7 Administrator Login Page – entry point of the system*

The Admin Login Management System provides entry to the login page. On access to the dashboard, you will be required to provide a valid username and password before access is allowed. This configuration provides authentication where only approved administrators can access the site. There is also a Forget password button which can assist in case you forget the basic information.

### 3.1 Login with Invalid Credentials:

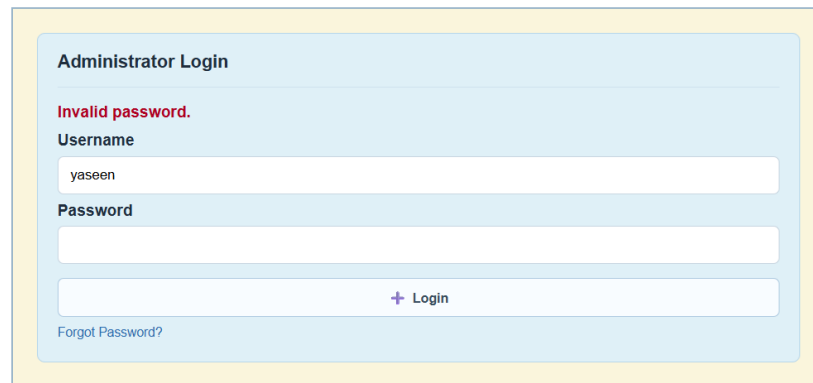
A screenshot of a web application's administrator login page. The page has a light blue header with the text "Administrator Login". Below the header, there is a red error message that says "Invalid password." in bold. Underneath the error message, there are two input fields: "Username" and "Password". The "Username" field contains the text "yaseen". Below the "Password" field, there is a "Login" button with a purple plus icon. At the bottom left of the login form, there is a link that says "Forgot Password?".

Figure 8 Administrator Login Page – incorrect username or password attempt.

Its system gives an error message indicating that credentials is not valid whenever the user gets it wrong with his user name or password. It's a fast method in telling you what did go wrong, and it pushes you into trying it again with the correct details. All these prevent unauthorized persons to enter into the thing without necessary authorization and ensure that the process is user-friendly. The system, in the background checks, the credentials with the database through a secure verification step before authorization.

### 3.2 Successful Login:

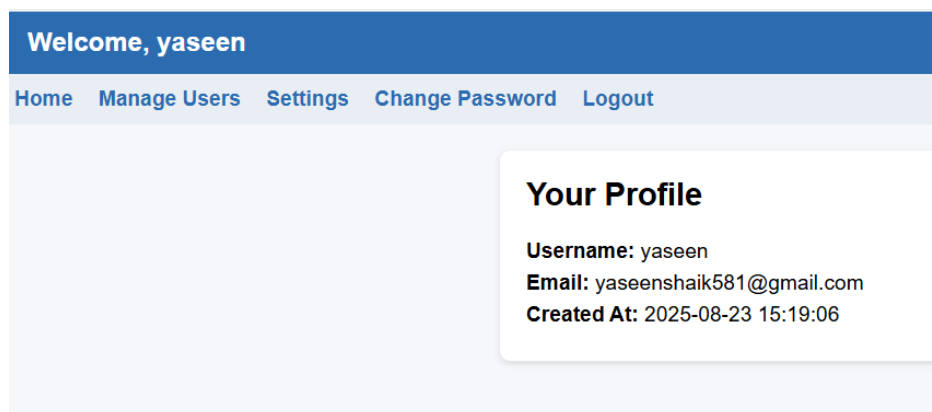
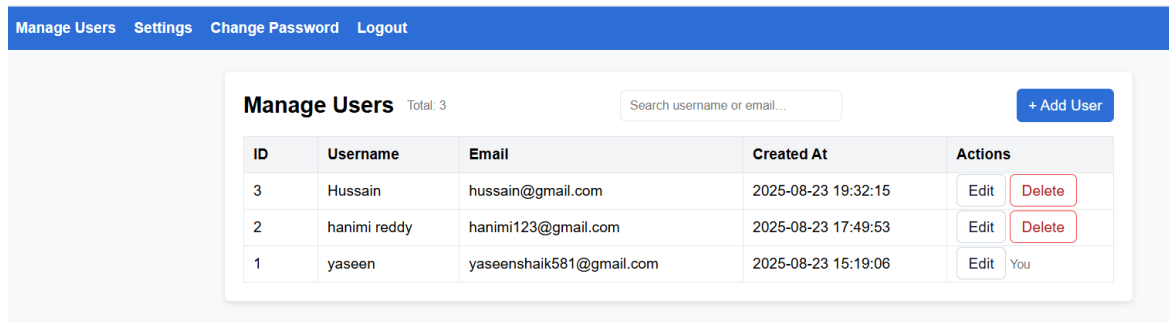
A screenshot of a web application's administrator dashboard. The dashboard has a dark blue header with the text "Welcome, yaseen". Below the header, there is a navigation bar with links: "Home", "Manage Users", "Settings", "Change Password", and "Logout". The main content area is light blue. On the right side of the main content area, there is a white box titled "Your Profile". Inside this box, there is user information: "Username: yaseen", "Email: yaseenshaik581@gmail.com", and "Created At: 2025-08-23 15:19:06".

Figure 9 Administrator Login Page – successful login redirects to Dashboard.

Upon typing the right login details the admin will have access, and the main dashboard will appear. That is when the session of the admin actually begins. This is enforced upon login so that only professionals with the appropriate credentials can have access to sensitive features such as user management and settings.

### 3.3 Manage Users – User List:



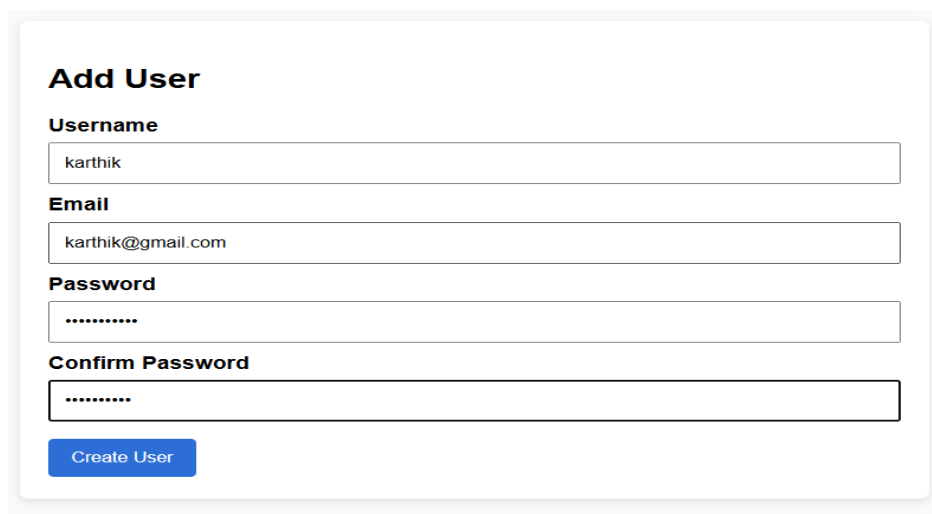
The screenshot shows a web interface for managing users. At the top, there is a blue navigation bar with links: "Manage Users", "Settings", "Change Password", and "Logout". Below this, the "Manage Users" section is displayed. It includes a header "Manage Users" with a "Total: 3" count, a search bar "Search username or email...", and a "+ Add User" button. A table lists the administrators:

ID	Username	Email	Created At	Actions
3	Hussain	hussain@gmail.com	2025-08-23 19:32:15	Edit Delete
2	hanimi reddy	hanimi123@gmail.com	2025-08-23 17:49:53	Edit Delete
1	yaseen	yaseenshaik581@gmail.com	2025-08-23 15:19:06	Edit You

Figure 10Manage Users page listing all administrators stored in the database.

Hey, there is a table on this page containing all the administrator accounts, the IDs, usernames, emails, dates they were signed up. Information It is drawn up in real time out of our database. Using this, admins are able to update or delete user accounts when required. It shows the Read of CRUD, in essence.

### 3.4 Manage Users – Add User:

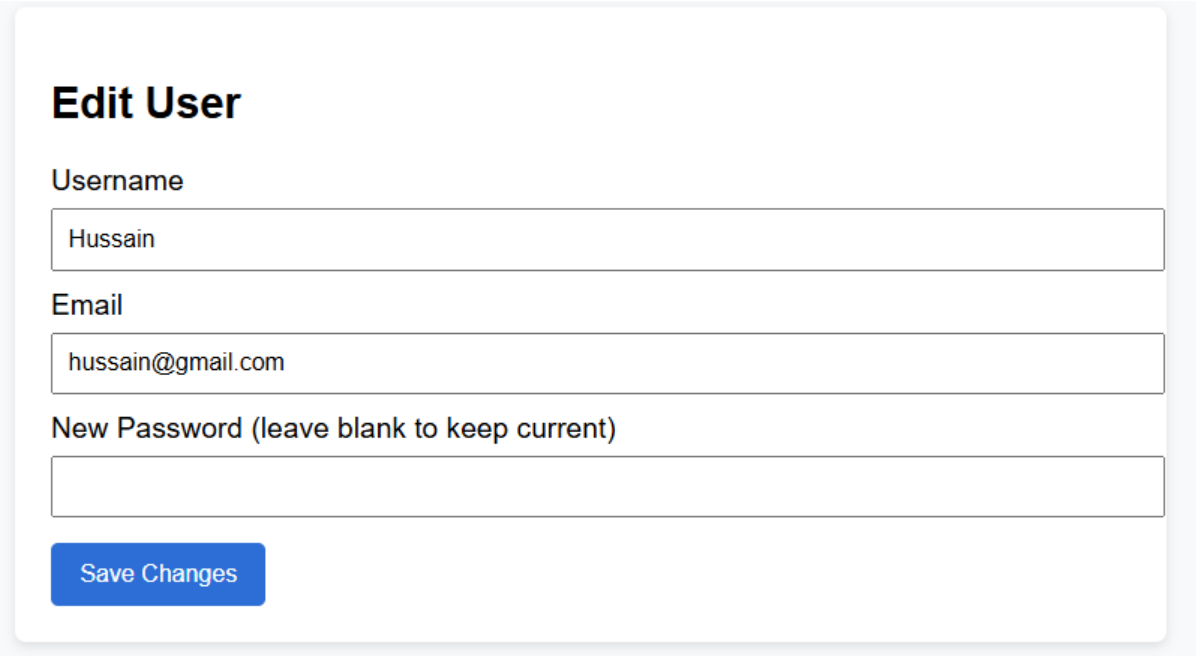


The screenshot shows the "Add User" form. It has a title "Add User" and four input fields: "Username" (containing "karthik"), "Email" (containing "karthik@gmail.com"), "Password" (containing "....."), and "Confirm Password" (containing "....."). A blue "Create User" button is at the bottom.

Figure 11Add User form for creating new administrator accounts

Essentially, admins are able to create new users by scribbling on a registration kind of form. The system checks twice on matters such as the format of your email address and your password, that it is actually a strong password and then whacks the data into the database. That's the Create part of CRUD, then.

### 3.5 Manage Users – Edit User:



**Edit User**

Username  
Hussain

Email  
hussain@gmail.com

New Password (leave blank to keep current)

Save Changes

*Figure 12 Edit User form to update administrator details.*

The Edit feature serves the basic purpose of allow us to make changes to user details, such as usernames or email, without ever feeling involved in any way. It provides us with the ability

to have accounts running smoothly and the data up to date. That is only the Update operation in CRUD terms.

### 3.6 Manage Users – Delete User:

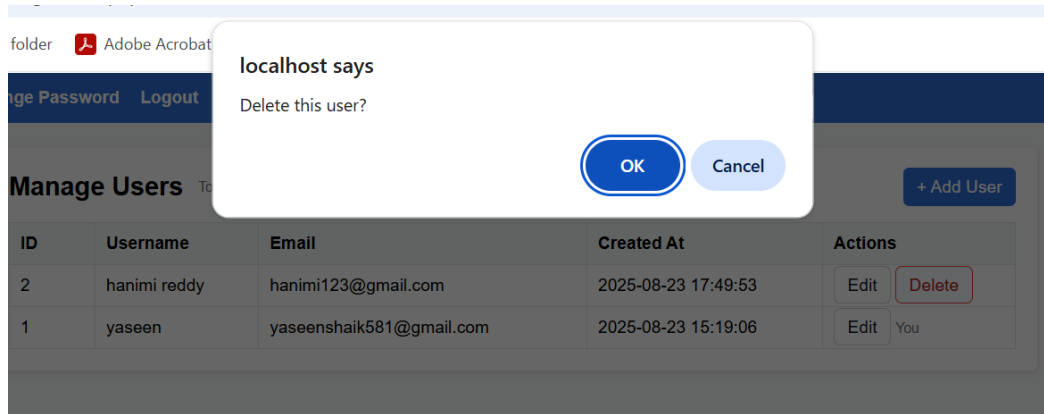


Figure 13Delete User option with confirmation prompt

The Delete button ensues the authority upon which the admins drop users off the system. A confirmation pop up is poisoned to ensure that there is no accidental spoil of the data although before they make the final move a confirmation screen emerges. And that is how the Delete element of CRUD is bundled up.

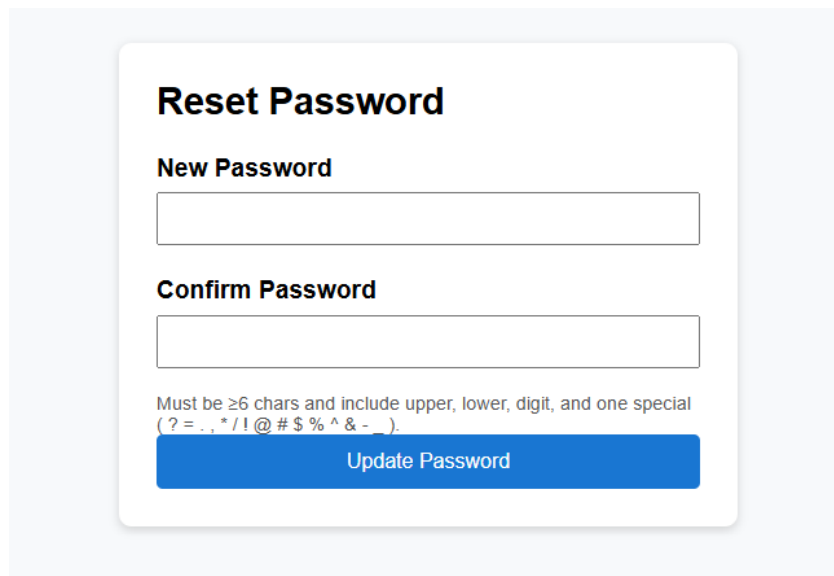
### 3.6 Forgot Password Page :

The screenshot shows a 'Forgot Password' form. It has a title 'Forgot Password' in bold. Below the title is a label 'Email Address' followed by a text input field containing 'admin@example.com'. At the bottom of the form is a blue button labeled 'Send Reset Link'.

Figure 14Forgot Password form where administrators enter their registered email address.

In case an administrator loses their login password, he/she can simply ask to get a reset and type the registered email address. The system will verify the email and create secure reset token. The system does not display the password immediately and chooses to send a password reset link, instead, which enhances the level of security given. This assists in repelling unauthorized access.

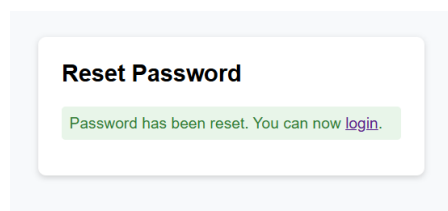
### 3.7 Reset Password Page (via token link):

A screenshot of a web form titled "Reset Password". It contains two input fields: "New Password" and "Confirm Password". Below the "Confirm Password" field, there is a text requirement: "Must be ≥6 chars and include upper, lower, digit, and one special (? = . , \* / ! @ # \$ % ^ & - \_ )". At the bottom of the form is a blue button labeled "Update Password".

*Figure 15 Reset Password form accessed via secure token*

A special token is given to redirect the admin to the reset page. The database stores the token, expiry time to prevent being used after expiry. The other password can be safely set by the admin on this page.

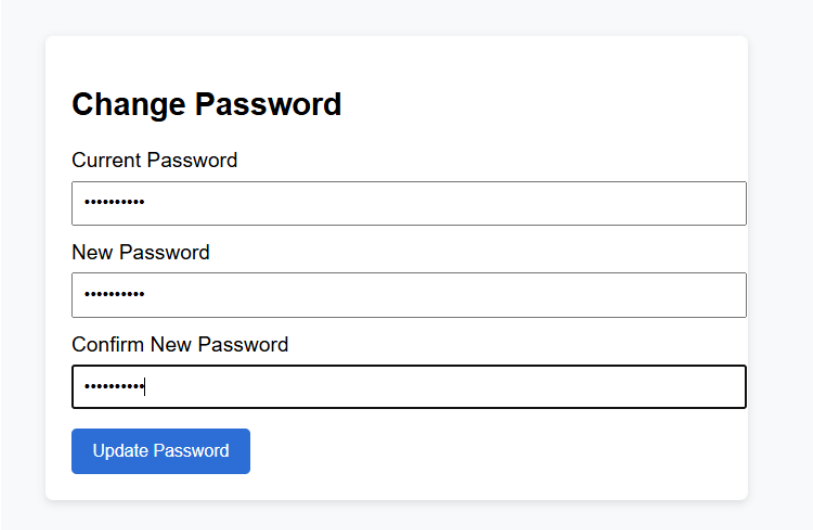
### 3.8 Successful Password Reset:

A screenshot of a confirmation message box titled "Reset Password". It contains a green message: "Password has been reset. You can now [login](#)."

*Figure 16 Confirmation of successful password reset*

Then when the new password actually satisfies the strength verification and matches the confirmation the system merely updates the database with the new hash. Thereafter, the administrator is only expected to log in directly using the new password. A lot of security and end-user friendliness win.

### 3.9 Change Password Page:

A screenshot of a web form titled "Change Password". The form is white with a light gray border and is set against a light gray background. It contains three input fields: "Current Password", "New Password", and "Confirm New Password". Each field has a placeholder of seven dots. Below the fields is a blue button with the text "Update Password".

**Change Password**

Current Password

.....

New Password

.....

Confirm New Password

.....

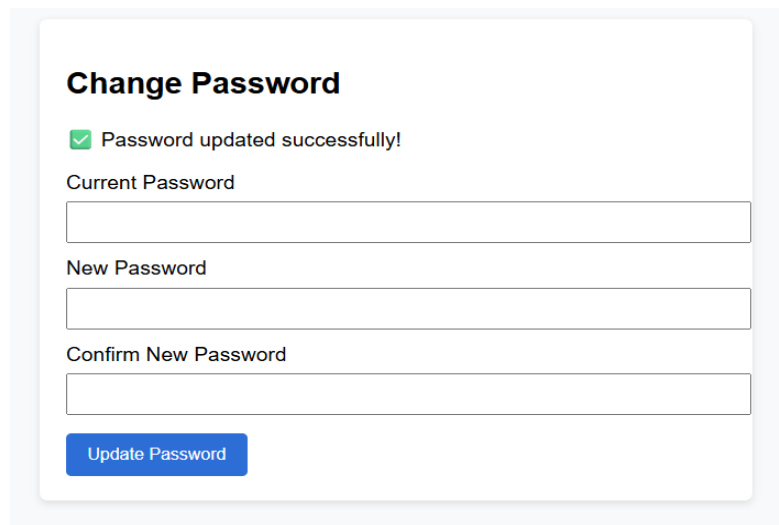
Update Password

*Figure 17 Change Password form where administrator updates their credentials.*

Change Password page allows admins with the logged in status to change their passwords in a fast and secure way. You put in your current password, then in a new password and validate it, this way unauthorised changes cannot be made. The system, also, ensures the new password conforms to the policy of password strength: at least six characters, an uppercase letter, a lowercase letter, a figure, a special character. When it has a too weak password or can not be matched, it will be rejected and hence you get a truly strong password.

### 3.10 Successful Change Password Message:





**Change Password**

✓ Password updated successfully!

Current Password

New Password

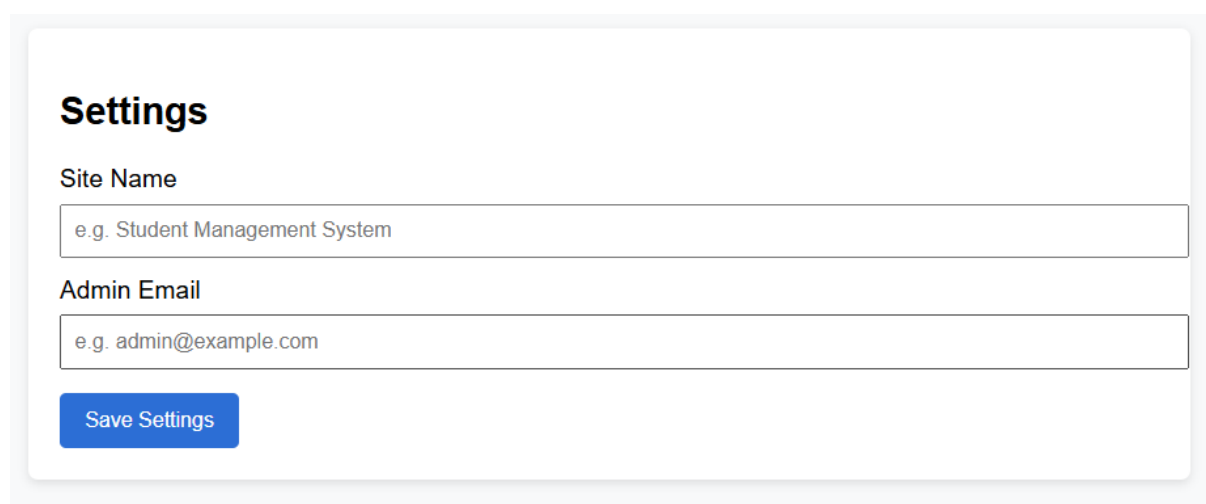
Confirm New Password

Update Password

Figure 18 System confirmation of successful password change

Once the admin provides the correct current password and a strong new password the system replaces the current password with a new password hash in the database. There is the appearance of a success message complying an individual with the fact that all is fine. Since that time, the administrator should be using the newer password every time they log-in. No old creds of a out of reach. That is in fact a feature that raises the security of the system since it challenges admins to refresh their passwords on a frequent basis.

#### 4.0 Settings Page :



**Settings**

Site Name

Admin Email

Save Settings

Figure 19 Settings page where administrator can update system preferences

Security Feature	Evidence (Screenshot / Proof)		Purpose
1	<b>Hashed Passwords</b>	phpMyAdmin screenshot showing password_hash column with MD5 hashes	Ensures no plaintext passwords are stored, protecting users if database is leaked
2	<b>Password Reset Token</b>	Screenshot of reset_token_hash and reset_token_expires populated after reset request	Provides secure, time-limited password reset instead of exposing passwords
3	<b>Token Cleared After Reset</b>	Screenshot showing reset_token_hash and reset_token_expires set to NULL after successful reset	Prevents reuse of old reset links, ensuring one-time use
4	<b>Password Strength Policy</b>	Screenshot of Change Password page rejecting weak password	Enforces stronger authentication by requiring complex passwords
5	<b>Session Authentication</b>	Screenshot showing redirect to login when trying to access dashboard without login	Protects restricted pages from unauthorized users



