PCA -2 ASSIGNMENT

Topic: Report on User Registration and Login System

For

[BCAC 591]: PHP WITH MYSQL LAB

Submitted by

[Pratyusha Mukherjee-23442723032
Hritick Ghosal-23442723024]

CO-ORDINATED BY:
Prof Majid Malik

PROGRAM – BCA (2023-2027)

THIRD YEAR
SEMESTER – 5th
In

NSHM College of Management and technology
124, 60, Basanta Lal Saha Rd, Tara Park, Behala, Kolkata, West Bengal-700053

Introduction

The **User Registration and Login System** is a fundamental component of modern web applications, providing secure access control and personalized user experiences. This system is built using **PHP** and **MySQL**, ensuring dynamic server-side processing and reliable database management. It allows users to register, authenticate, update their profiles, and manage their sessions securely. The project emphasizes security measures such as password hashing, SQL injection prevention, and session protection, making it both practical and secure for real-world deployment. The system's modular design also allows easy scalability and future feature integration.

Purpose

The primary purpose of this project is to **develop a secure, efficient, and user-friendly authentication system** that fulfills the fundamental requirements of any web-based platform needing user access control.

This system allows new users to register by entering basic information such as username, email, and password, which are validated both on the client and server side. Once registered, users can log in to the system, manage their profile details, and securely log out. Administrators have the additional ability to manage user data and perform essential administrative operations like viewing user lists, editing accounts, and approving or deactivating users.

The purpose can be summarized as follows:

- To ensure **data confidentiality and integrity** through secure password encryption and session management.
- To provide **controlled access** by distinguishing between general users and admin-level users.
- To demonstrate the **integration of PHP and MySQL** for dynamic database-driven applications.
- To apply **real-world security practices**, such as password hashing, SQL injection prevention, and input validation.
- To help students and developers understand how a full-fledged authentication system can be built using basic web technologies.

Overall, the purpose is to provide a practical, educational, and secure solution that can be easily adapted and expanded into real-world applications like e-commerce platforms, online portals, and management systems.

Software Requirements Specification (SRS)

1. Core Features & System Work Flow

1.1 User Features (Primary flows)

1. **Registration (Sign Up)**
    - Users can create an account by providing a unique username (or email), password, and optional profile details such as name.
    - Both client-side and server-side validations ensure all required fields are complete and password strength meets defined criteria.
    - Passwords are securely hashed using PHP's `password_hash()` function before being stored in the database.
    - On successful registration, user is redirected to the login page (or optionally logged in automatically).

2. **Login (Authentication)**
    - Users authenticate (log in) with their registered username (or email) and password.
    - Passwords are verified using PHP's `password_verify()` function to ensure secure authentication.
    - On successful login, a secure session is created (so the system safely remembers the user until they log out) and the user is redirected to their profile page.
    - Failed login attempts return a generic error message (no leakage of which field was wrong).

3. **Profile Page (Protected)**
    - Displays user-specific information (username, registration date, editable display name). Accessible only when a valid session is active (meaning the user is logged in and verified), ensuring secure and authenticated access.
    - Includes a logout option that securely terminates the active session.

4. **Change Password**
    - Logged-in users can change their password by submitting current password and new password.
    - The current password is verified before allowing the update; the new password is securely hashed (converted into an unreadable encrypted form to prevent misuse) before storage.

5. **Logout**
    - Securely ends the user session using PHP's session management functions (meaning the system clears all login data using `session_unset()`, `session_destroy()`, and cookie removal to prevent unauthorized access after logout).

1.2 Optional / Administrative Features

1. Admin Dashboard

- Provides a secure interface for administrators to view and manage registered users.
- Access is restricted to authorized admin credentials.

2. Account Edit & Deactivation/Deletion

- Allows users or administrators to edit and deactivate or permanently delete user accounts, ensuring flexibility and control over the account lifecycle.

3. Email Verification

- Introduces an email verification step during registration to confirm the user's email address, enhancing account authenticity and security.

---

2. Basic Security Features

1. **Password Hashing:**
    - Store passwords securely using PHP's password_hash() with the default bcrypt algorithm (it automatically converts passwords into a hidden, unreadable form).
    - Authenticate users with PHP's function password_verify() (it safely checks if the entered password matches the stored one without revealing the actual password).

2. **SQL Injection Prevention:**
    - Use MySQLi prepared statements for all database interactions (they keep user input separate from SQL commands, making data handling safer).
    - Avoid direct string concatenation in SQL queries (to prevent SQL injection — a hacking method where attackers insert malicious SQL code).

3. **Session Security:**

    - Use session_regenerate_id(true) on login (this gives each user a new session ID to stop session hijacking) and set secure cookie options like httponly, secure (to protect cookies from hackers).
    - End sessions properly using session_unset(), session_destroy(), and cookie deletion (this clears all login data).
    - These steps ensure PHP sessions safely track **logged-in** users and are fully erased after **logout** to prevent unauthorized reuse.

4. **Input Validation & Output Escaping:**

    - Apply server-side validation for all user inputs, checking for required fields, length limits, and allowed characters.
    - Escape user-generated content using PHP's function `htmlspecialchars()` (which converts special characters into safe symbols to prevent Cross-Site Scripting or XSS — a type of attack where hackers inject malicious code into web pages) when displaying data.

5. **Error Handling & Messaging:**

    - Provide generic error messages for authentication failures (avoid indicating whether username or password was wrong).
    - Log detailed server-side errors to a secure location for debugging, without exposing technical details to users.

6. **HTTPS Requirement (deployment):**

    - Use HTTPS protocol in production environments (so that all data sent between the user and the website is encrypted and protected from hackers) to keep user credentials safe from interception.

Database Schema and ER Diagram

Database Name: user_admin_system

user_admin_system

admins

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	int(11)	No				
name	varchar(255)	No				
email	varchar(255)	No				
password	varchar(255)	No				
role	varchar(50)	Yes	admin			
created_at	timestamp	No	current_timestamp()			
last_login	timestamp	Yes	*NULL*			
is_active	tinyint(1)	Yes	1			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	1	A	No	
email	BTREE	Yes	No	email	1	A	No	
idx_admin_email	BTREE	No	No	email	1	A	No	

password_resets

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	int(11)	No				
email	varchar(255)	No				
token	varchar(255)	No				
created_at	timestamp	No	current_timestamp()			

Indexes

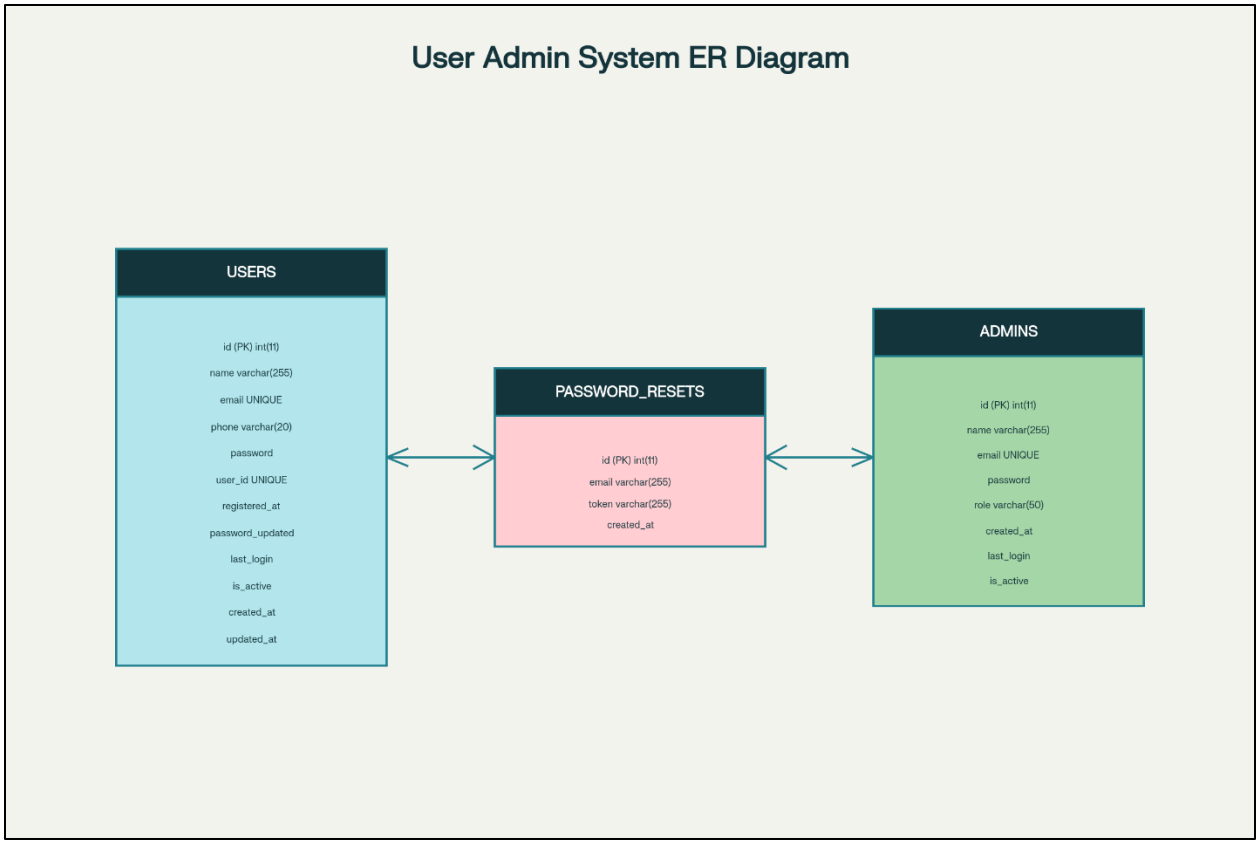
Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	0	A	No	
email	BTREE	No	No	email	0	A	No	
token	BTREE	No	No	token	0	A	No	

users

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	int(11)	No				
name	varchar(255)	No				
email	varchar(255)	No				
phone	varchar(20)	Yes	*NULL*			
password	varchar(255)	No				
user_id	varchar(50)	No				
registered_at	timestamp	No	current_timestamp()			
password_updated_at	timestamp	Yes	*NULL*			
last_login	timestamp	Yes	*NULL*			
is_active	tinyint(1)	Yes	1			
created_at	timestamp	No	current_timestamp()			
updated_at	timestamp	No	current_timestamp()			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	
email	BTREE	Yes	No	email	2	A	No	
user_id	BTREE	Yes	No	user_id	2	A	No	
idx_user_email	BTREE	No	No	email	2	A	No	



Relationships & Data Flow:

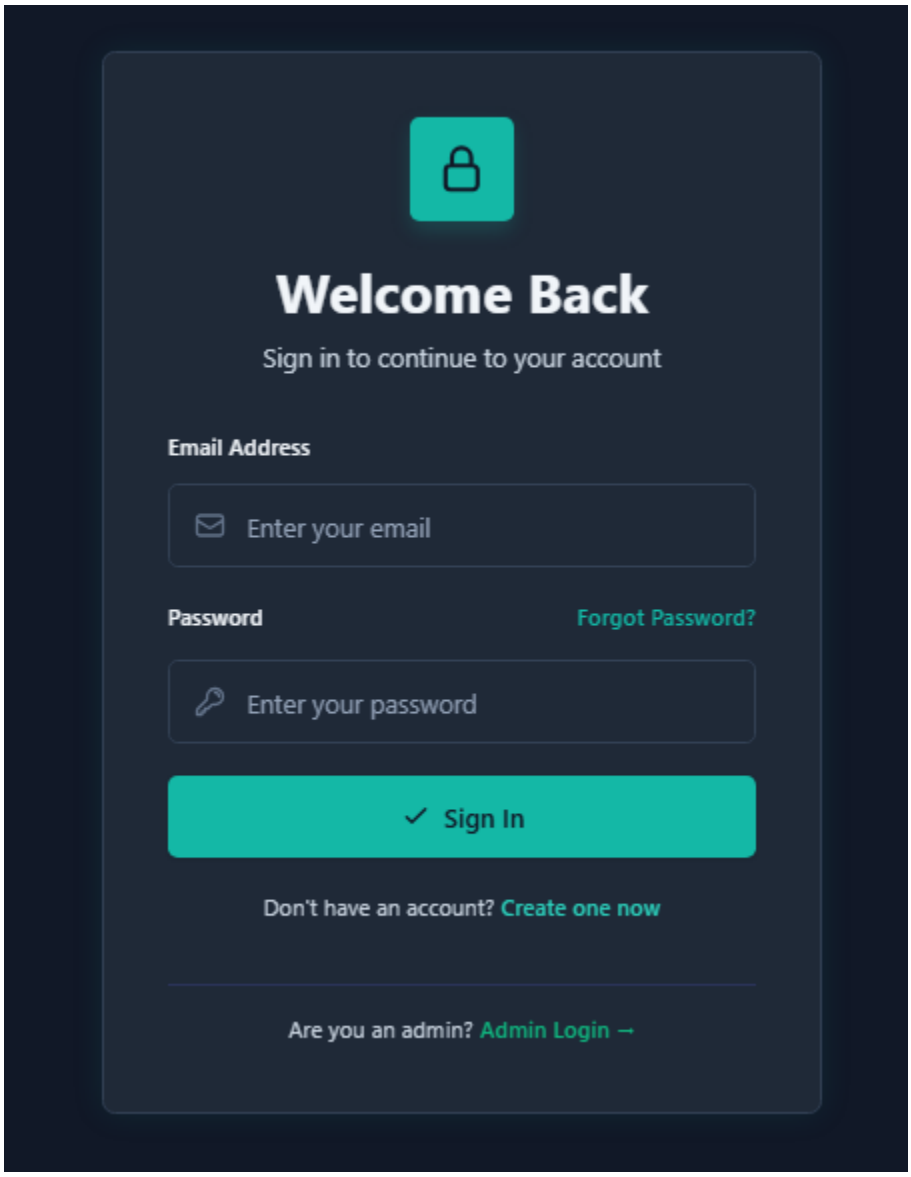
The database implements **implicit relationships** through the email field as a common attribute. The password_resets table serves as a junction point with **many-to-one** relationships to both admins and users tables:

- **password_resets.email → users.email**: enables regular users to request password resets
- **password_resets.email → admins.email**: allows administrators to recover their accounts

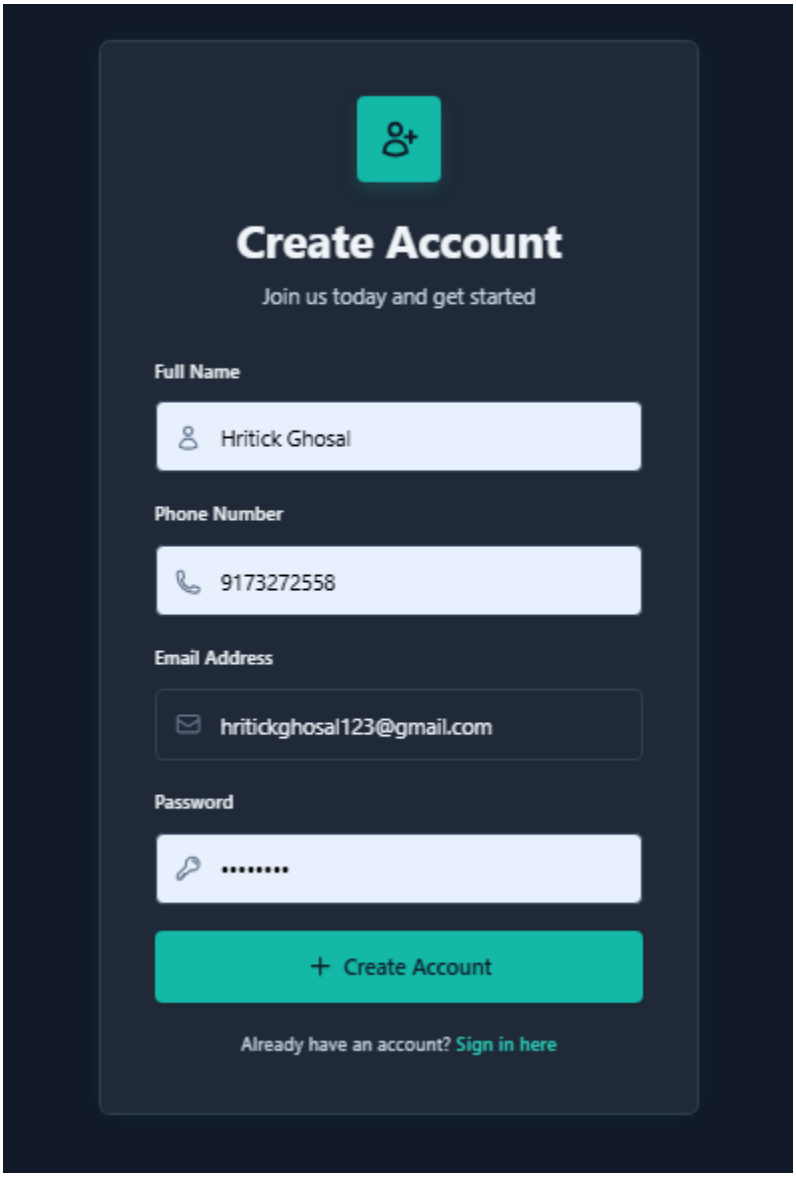
This design pattern allows a single password reset mechanism to serve both user types while maintaining separate authentication tables. The email field acts as a natural foreign key without explicit foreign key constraints, providing flexibility for the reset process.

UI / UX

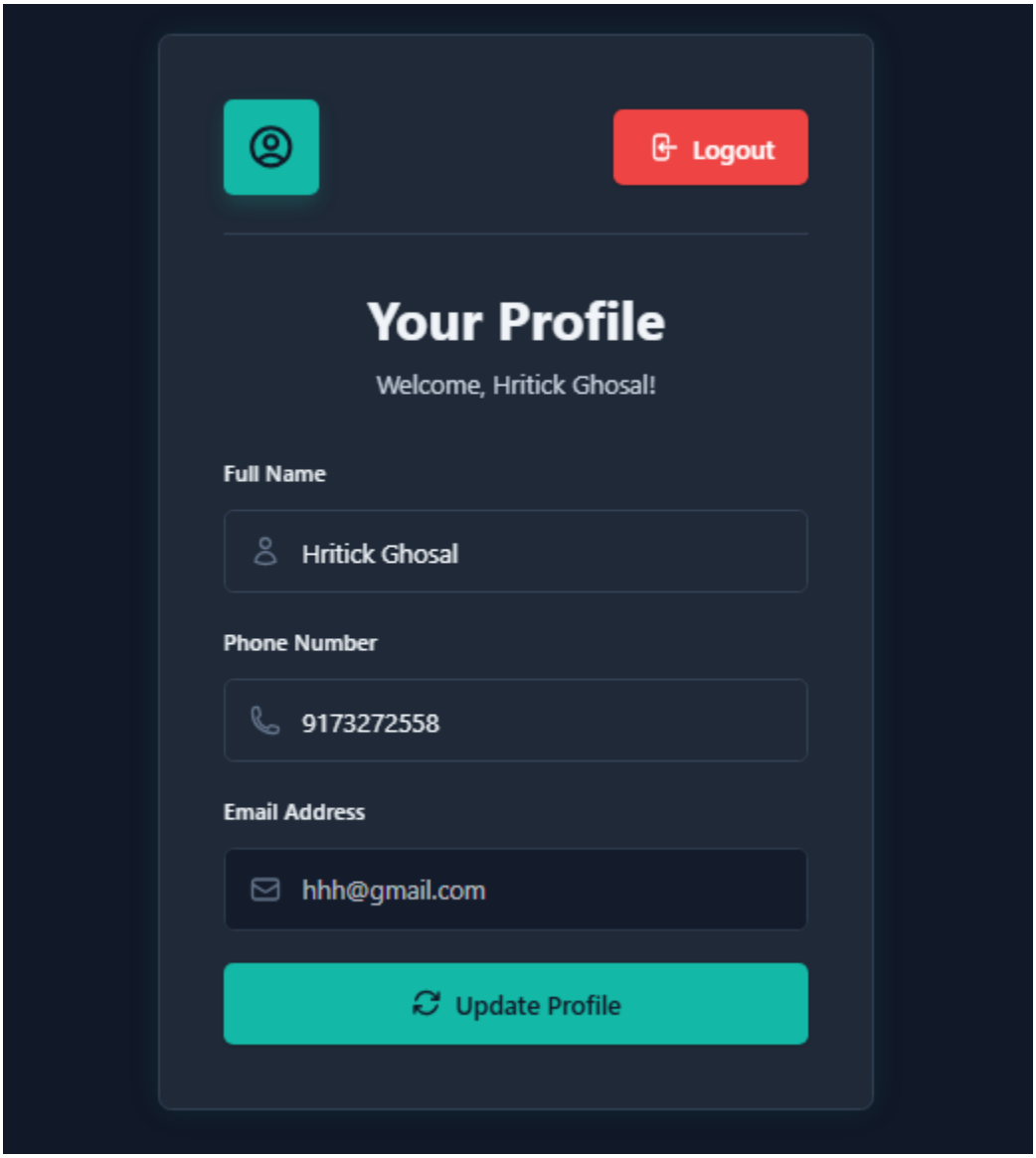
Home Page →



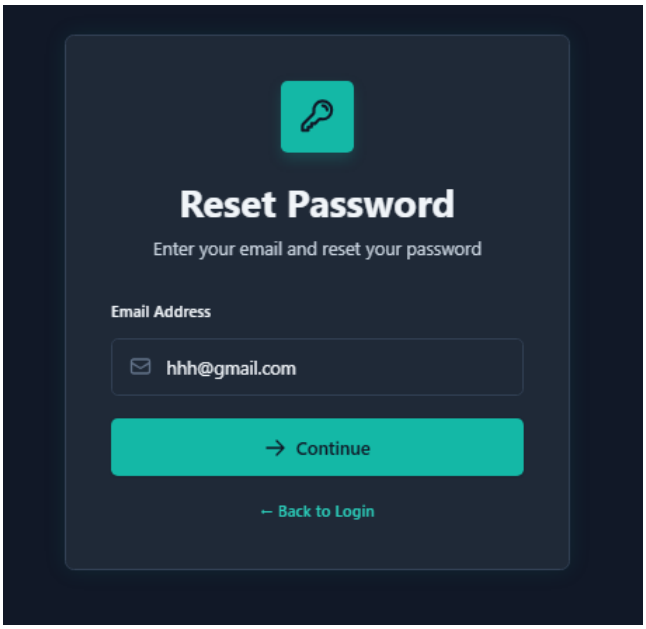
User Registration →



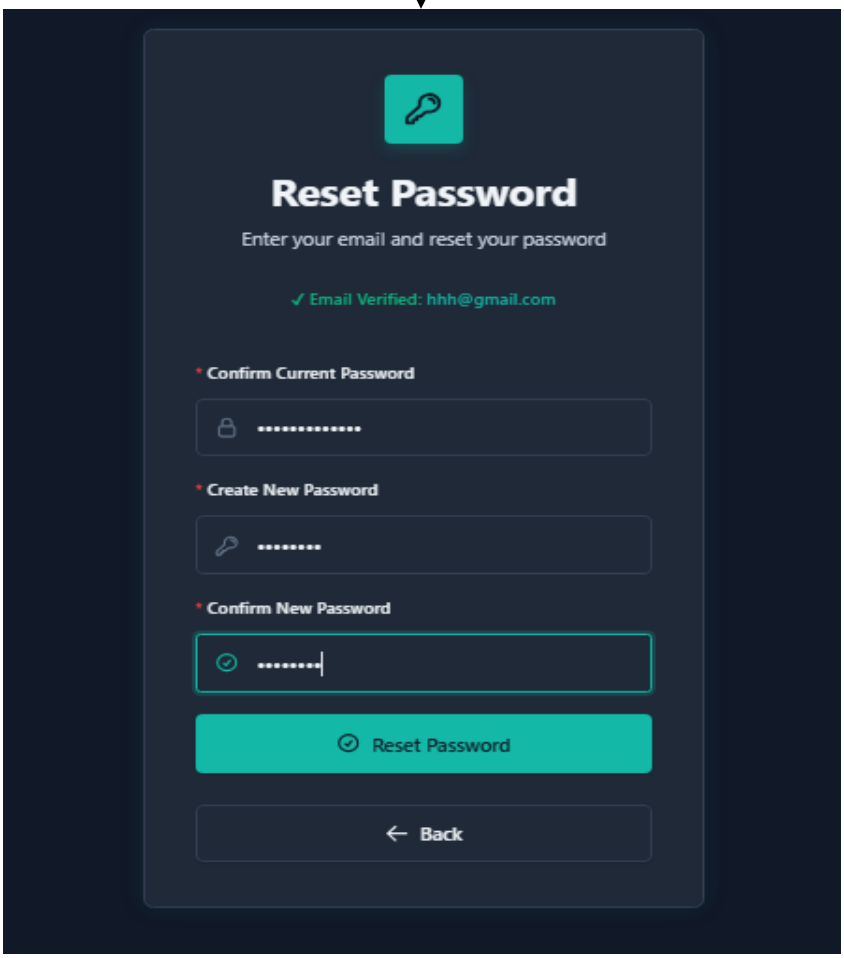
User Login and Update Profile →



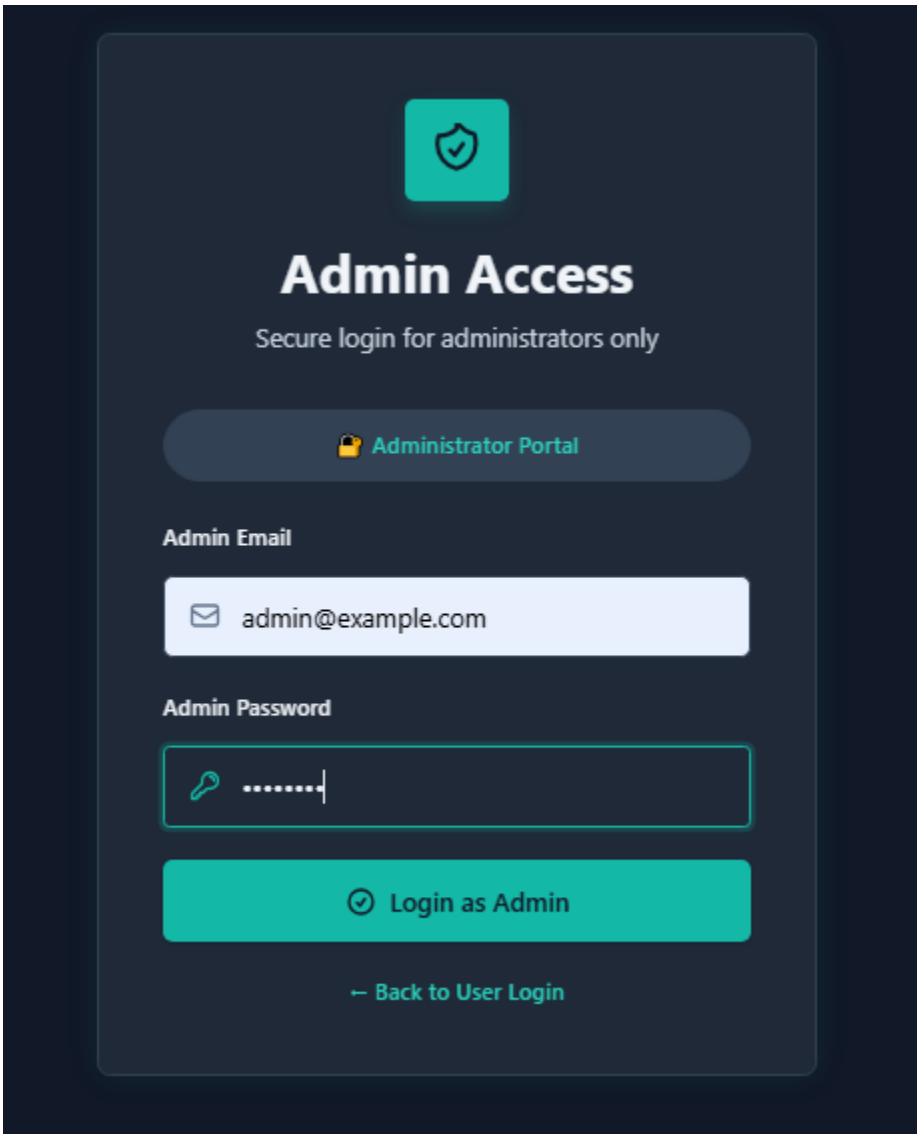
Password Reset Feature→



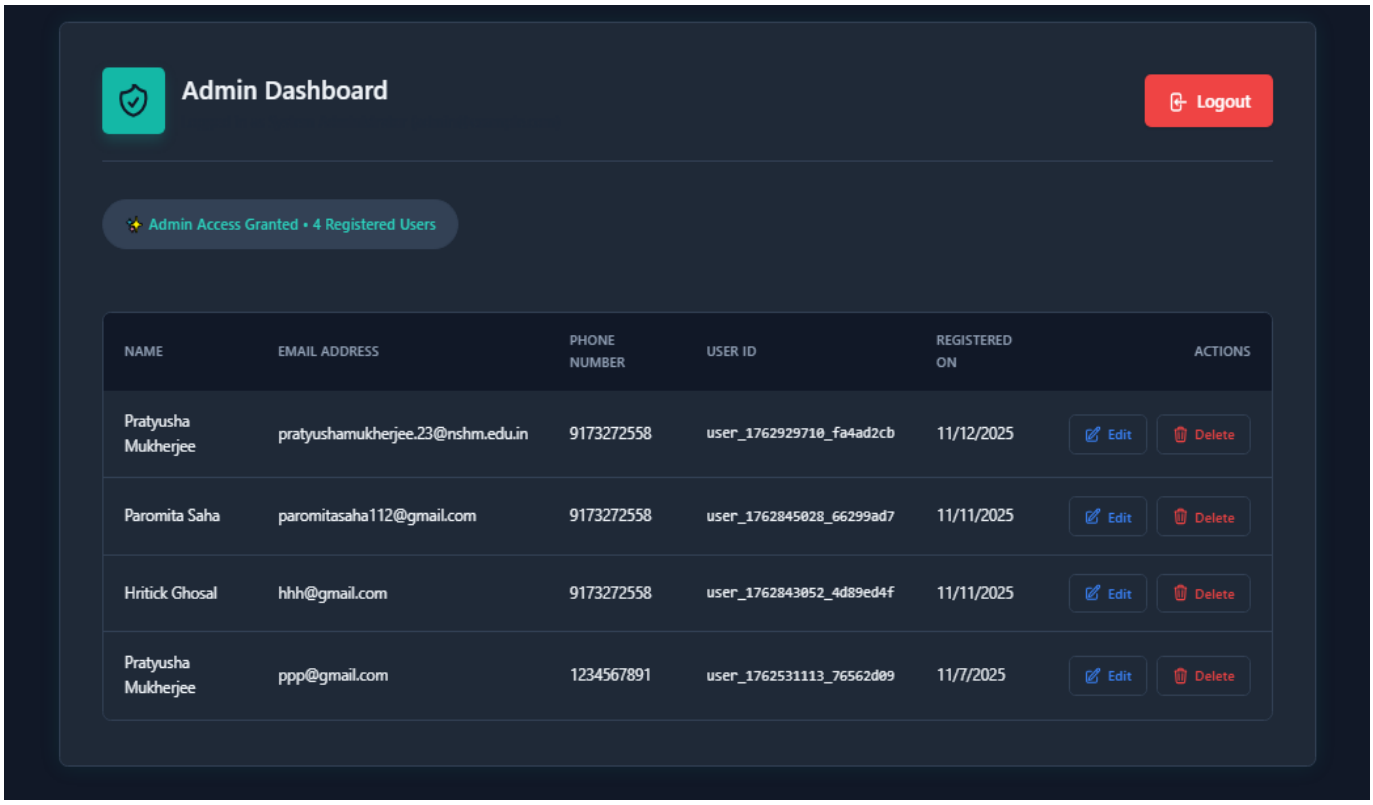
Next Step



Admin Login →


Admin Access
Secure login for administrators only
Administrator Portal
Admin Email
admin@example.com
Admin Password
Login as Admin
← Back to User Login

Admin Dashboard →



Future Implementation

While the current system efficiently handles registration, login, and user management, there are several enhancements that can be incorporated in the future to make it more powerful, secure, and scalable. Some of the proposed future implementations include:

1. **Two-Factor Authentication (2FA):**
   Adding an OTP-based verification system using email or SMS during login to provide an additional layer of security.
2. **Password Recovery via Email:**
   Allowing users to recover their forgotten passwords by receiving a password reset link on their registered email.
3. **Advanced User Roles and Permissions:**
   Introducing role-based access control where users can have roles like editor, moderator, or admin, each with specific privileges.

4. **Activity Logs:**
   Implementing an activity log feature to track all user and admin actions (login, logout, profile edits, etc.) with timestamps stored directly in the database.
5. **Captcha Verification:**
   Integrating Google reCAPTCHA to prevent automated bots from registering or attempting brute-force logins.
6. **Responsive UI Framework:**
   Enhancing the user interface using frameworks such as **Bootstrap** or **Tailwind CSS** for a better mobile and desktop experience.
7. **Email Notifications:**
   Sending automated email alerts for password changes, login attempts, and account updates to keep users informed about their account activities.
8. **API Integration:**
   Extending the system to provide secure APIs for mobile applications or third-party integrations using technologies like **JSON Web Tokens (JWT)**.

By implementing these improvements, the system can evolve into a robust, enterprise-level authentication and user management platform.

Conclusion

The **User Registration and Login System** built using **PHP and MySQL** effectively fulfills the core requirements of authentication and secure access management in web applications. The system not only allows users to register, log in, and manage their accounts but also incorporates strong security measures such as password hashing, SQL injection prevention, and safe session handling.

This project serves as a practical demonstration of backend development using PHP and database integration with MySQL. It highlights how secure coding standards and validation practices can protect sensitive user data from unauthorized access and attacks. The admin dashboard provides centralized control over all user activities, showcasing the importance of administrative monitoring in multi-user environments.

In conclusion, this project has successfully achieved its objectives by combining security, functionality, and usability. It can serve as a foundation for developing more complex systems such as **content management systems, e-commerce sites, or educational portals**. Future enhancements can further expand its scope, making it an even more versatile and secure solution for real-world applications.

References

1. PHP Official Documentation – https://www.php.net/manual/
2. MySQL Official Documentation – https://dev.mysql.com/doc/
3. W3Schools PHP & MySQL Tutorials – https://www.w3schools.com/php/
4. GeeksforGeeks – PHP with MySQL Projects and Tutorials – https://www.geeksforgeeks.org/
5. OWASP Foundation – Web Security Guidelines – https://owasp.org/
6. MDN Web Docs – Secure Authentication and Form Handling – https://developer.mozilla.org/
7. TutorialsPoint – PHP Security Best Practices – https://www.tutorialspoint.com/php/

GitHub Link for Pratyusha Mukherjee

https://github.com/iam-pratyusha/USER_LOGIN_PHP_MYSQL

GitHub Link for Hritick Ghosal

https://github.com/Hritick-Ghosal/Login-and-Registration-