APEX PLANET SOFTWARE PVT LTD

Security Enhancements Documentation :

This application has been enhanced to protect against common web vulnerabilities by implementing:

- Prepared statements for all database queries
- Server-side and client-side form validation
- User roles and role-based access control

1. Prepared Statements

What:

All SQL queries use PDO prepared statements.

Why:

Prevents SQL injection attacks by separating SQL logic from user input.

Where:

Used in all files that interact with the database, such as:

- `login.php
- `register.php
- `create.php
- `edit.php
- `delete.php

Example:

```
<?php
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ?");
$stmt->execute([$username])
```

2. Form Validation:

Server-side Validation:

What:

All forms are validated in PHP before processing.

Why:

Ensures data integrity and prevents malicious or invalid data from being stored.

Where:

Implemented in all form-handling scripts, e.g.:

- `register.php`
- `login.php`
- `create.php`
- `edit.php`

Example:

```
<?php
if (empty($title)) {
   $errors[] = "Title is required.";
} elseif (mb_strlen($title) > 255) {
   $errors[] = "Title must be less than 255 characters.";
}
```

Client-side Validation

What:

HTML5 validation attributes and JavaScript are used in all forms.

Why:

Provides instant feedback to users and improves user experience.

Where:

All forms in the application.

Example:

html

```
<input type="text" name="title" required minlength="3" maxlength="255">
```

js

```
document.getElementById('formId').addEventListener('submit', function(e) {
   if (!this.checkValidity()) {
      e.preventDefault();
      alert('Please fill out the form correctly.');
   }
});
```

3. User Roles and Permissions

User Table Extension:

What:

A role column was added to the `users` table.

Why:

Enables role-based access control (e.g., admin, editor, user).

How:

sql

```
ALTER TABLE users ADD COLUMN role ENUM('admin', 'editor', 'user') NOT NULL
DEFAULT 'user';
```

Role-based Access Control

What:

Helper function `requireRole()` checks user roles before allowing access to protected pages.

Why:

Restricts sensitive actions (like creating, editing, or deleting posts) to authorized users only.

Where:

At the top of protected files, such as:

- create.php
- `edit.php`
- `delete.php`

Example:

```
<?php
requireRole(['admin', 'editor']);
```

\\Helper Function (in `utils/session.php`):\\

```
<?php
function requireRole($roles) {
    if (!isset($_SESSION['user_role']) || !in_array($_SESSION['user_role'], (array)$roles)) {
        header('HTTP/1.1 403 Forbidden');
        exit('Access denied');
    }
}
```

4. Summary of Security Measures

- **Prepared Statements:** Used everywhere to prevent SQL injection.
- **Server-side Validation**: All forms are validated in PHP.
- **Client-side Validation:** All forms use HTML5 and JS validation.
- **User Roles**: users table has a role column.
- **Access Control:** Only users with the correct role can access protected pages.

Prepared by :

TALADA SURYA MADHAV.