School of Information Technology Education
BACHELOR OF SCIENCE IN COMPUTER SCIENCE

Personal Portfolio with PHP File Handling (DOCUMENTATION)

ANGELES, DENMARC FRANCIS HARRY P.

21-BSCS-01

ROCHELLE S. LANTO

INSTRUCTOR

DECEMBER 13, 2024

Table of Contents

Introduction

This project creates a dynamic personal portfolio website, leveraging PHP file handling for efficient content management. The website is designed to be user-friendly and flexible, catering to both guest and administrator users. The key advantage of using PHP file handling is its simplicity and ease of implementation for a project of this scale, avoiding the overhead and complexity of a database system. This approach makes the website easily maintainable and adaptable to future changes.

The core functionality revolves around a clear separation of user roles and permissions. Guest users have read-only access to the portfolio's content, providing a straightforward viewing experience. Administrators, on the other hand, enjoy full control over the website's content, enabling them to easily update and maintain the portfolio's information. This dynamic content management system allows for frequent updates and ensures the portfolio always reflects the user's current skills and projects. The system's architecture is designed to be scalable, allowing for future expansion without significant changes to the core functionality. The use of a structured file system further enhances maintainability and organization.

Project Objective:

The primary objective is to create a highly dynamic and easily manageable personal portfolio. This involves:

1. **Enhanced Dynamism:** Allowing administrators to seamlessly add, edit, and delete content across various sections of the portfolio (e.g., About Me, Skills, Projects). This ensures the portfolio remains current and accurately reflects the user's skills and experience.

2. **Flexible Content Management:** Implementing a system that simplifies content updates, making it easy for the administrator to maintain the website without requiring extensive technical knowledge. The use of text files managed by PHP provides a straightforward solution.

3. **Clear User Roles:** Distinguishing between guest and administrator access levels, ensuring that only authorized users can modify the website's content. This maintains the integrity of the portfolio and prevents unauthorized changes.

4. **Simplified Technology:** Utilizing PHP file handling as the core technology for content management, avoiding the complexity and resource requirements of a database system. This makes the project more accessible and easier to maintain.

5. **Scalability:** Designing the system to be easily scalable, allowing for future expansion and the addition of new features without requiring a complete system overhaul.

By achieving these objectives, the project delivers a robust, user-friendly, and easily maintainable personal portfolio website that effectively showcases the user's skills and experience.

System Design

This section details the design and functionality of the personal portfolio website, focusing on its structure, user interface, and content management system.

Website Structure and Navigation:

The website features a clean and intuitive design with five main sections accessible through navigation tabs: Home, About Me, Skills, Projects, and Contact. These tabs are animated to enhance user engagement. A dedicated button allows users to return directly to the login page. The navigation bar also includes three social media links for easy access to the user's profiles, along with a fixed logo and name (non-editable, even for administrators).

User Roles and Permissions:

The website supports two user roles: guest and administrator.

- **Guest Users:** Guests can view all portfolio content (Home, About Me, Skills, Projects, and Contact) but cannot edit it. A contact form allows guests to send messages to the portfolio owner.
- **Administrator Users:** Administrators have significantly more privileges. They can edit the content of the Home, About Me, and Contact sections. For the Skills section, administrators can add and delete skills. In the Projects section, administrators can add, delete, and edit project details.

File Storage Strategy:

The website's files are organized for maintainability and scalability:

- **index.php:** The main entry point, handling user authentication and routing.
- **css/:** Contains all CSS files responsible for the website's visual appearance.
- **script/:** Houses all JavaScript files for dynamic functionality and user interaction.

- **php/:** Contains PHP classes and scripts for handling form submissions, file operations, and other server-side logic.
- **content/:** Stores portfolio content as text files, allowing for easy content management via PHP file handling. Each section (Home, About Me, etc.) likely has its own text file.
- **upload/:** Stores uploaded images, such as project screenshots or profile pictures.
- **img/:** Contains static images used for the website's design and branding elements (logo, background images, etc.).
- **admin/:** Contains the administrator's interface files, providing tools for content management.
- **guest/:** Contains files specific to the guest user experience (though much of this might be handled by index.php and the main PHP files).

This structured approach ensures a clear separation of concerns, making the website easier to maintain and update. The use of text files for content storage simplifies the system, avoiding the complexity of a database for this relatively small-scale project.

Implementation

The development of this dynamic personal portfolio website utilizes a combination of front-end and back-end technologies, each playing a crucial role in creating a functional and visually appealing online presence.

1. HTML (HyperText Markup Language):

HTML forms the foundational structure of the website. It defines the content elements, such as headings, paragraphs, images, and links, providing the basic framework for the layout and organization of the webpage.

2. CSS (Cascading Style Sheets):

CSS is responsible for the visual presentation of the website. It dictates the styling aspects, including colors, fonts, spacing, layout, and responsiveness

3. JavaScript:

JavaScript adds interactivity and dynamic behavior to the website. While HTML provides the structure and CSS the styling, JavaScript enables elements like the animations on the navigation tab and potentially other interactive features that enhance the user experience. It can handle client-side operations, improving the responsiveness and user engagement of the portfolio.

4. PHPMailer:

PHPMailer is a PHP library specifically designed for sending emails. In this project, it's used to handle the functionality of the contact form. When a user submits a message through the contact form, PHPMailer takes the information and sends it to the designated email address. It simplifies the email sending process, managing the complexities of SMTP (Simple Mail Transfer Protocol) configurations.

5. PHP (Hypertext Preprocessor):

PHP is the heart of the back-end logic for this website. It handles all server-side operations, including:

Content Management: PHP interacts with the text files storing the portfolio content, enabling the administrator to add, edit, and delete information.

User Authentication: PHP likely handles the login process, verifying user credentials and granting access based on user roles (guest or administrator).

Form Processing: It receives and processes data submitted through forms, including the contact form (working in conjunction with PHPMailer).

Dynamic Content Generation: PHP fetches and displays the content from the text files, dynamically generating the web pages based on the requested section (Home, About Me, etc.).

FileHandling Class:

This class manages portfolio content stored in text files, offering methods for reading, writing, editing, searching, and deleting content. It avoids database use for simplicity.

Key Methods:

- __construct(): Initializes the file path.
- readContent(): Reads the file content.
- editContent(): Replaces the entire file content.
- editLineByIndex(): Edits a specific line.
- searchContent(): Searches for a term within the file.
- displayContent(): Displays content on the webpage.
- displayLineByIndex(): Displays a specific line.
- getLineFromFile(): Returns a specific line.
- getFilePath(): Returns the file path.
- addNewContent(): Appends new, sanitized content.
- deleteLineByIndex(): Deletes a specific line.

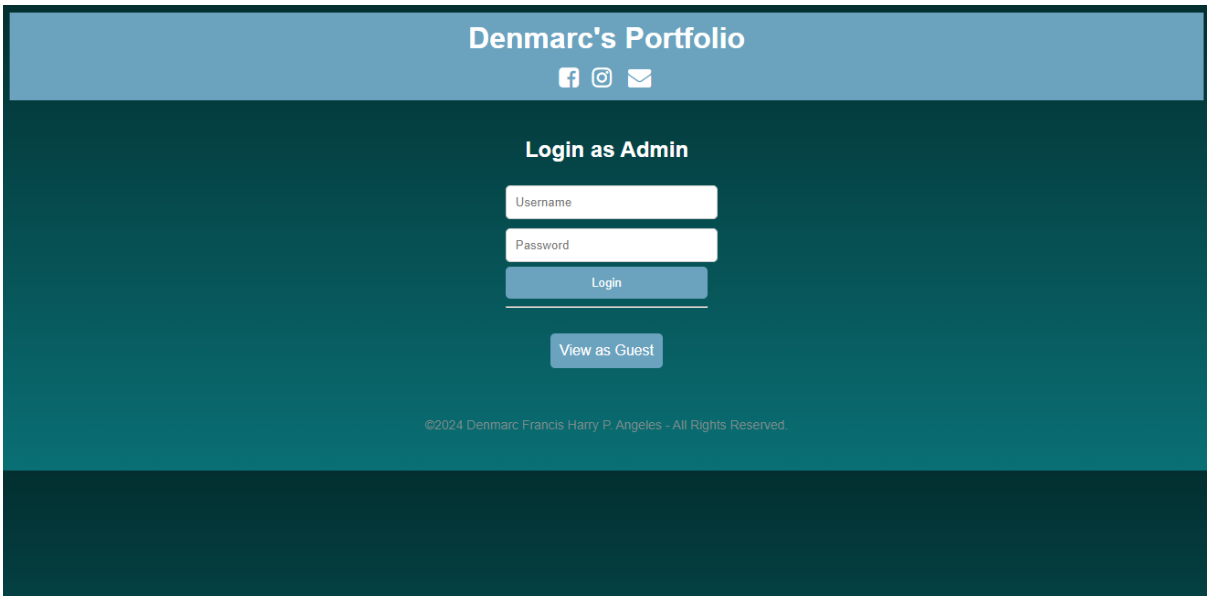
MailHandler Class:

This class uses PHPMailer to send emails from contact forms. It configures Gmail SMTP settings.

Key Methods:

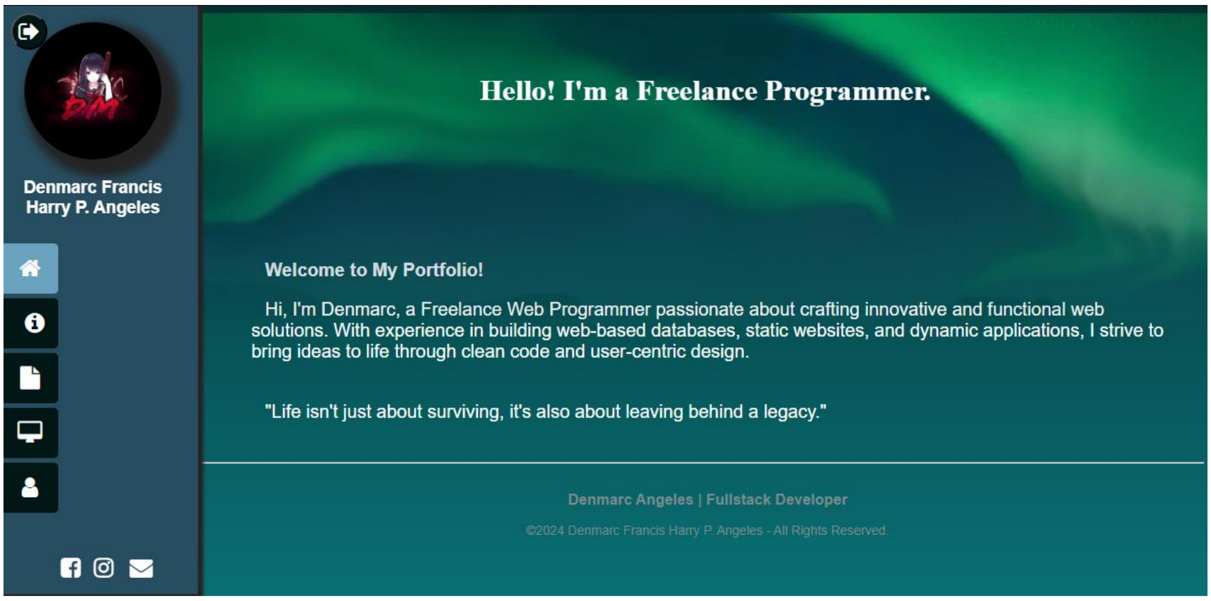
- __construct(): Initializes PHPMailer and SMTP settings (improve security by avoiding hardcoded credentials).
- sendMail(): Sends a general email (admin contact form).
- sendContact(): Sends a simplified contact email (guest contact form).
- formatMessage(): Formats the email body.
- alertAndRedirect(): Displays an alert and redirects the user.

Screenshots


Denmarc's Portfolio

Login as Admin

©2024 Denmarc Francis Harry P. Angeles - All Rights Reserved.


Denmarc Francis Harry P. Angeles

Hello! I'm a Freelance Programmer.

Welcome to My Portfolio!

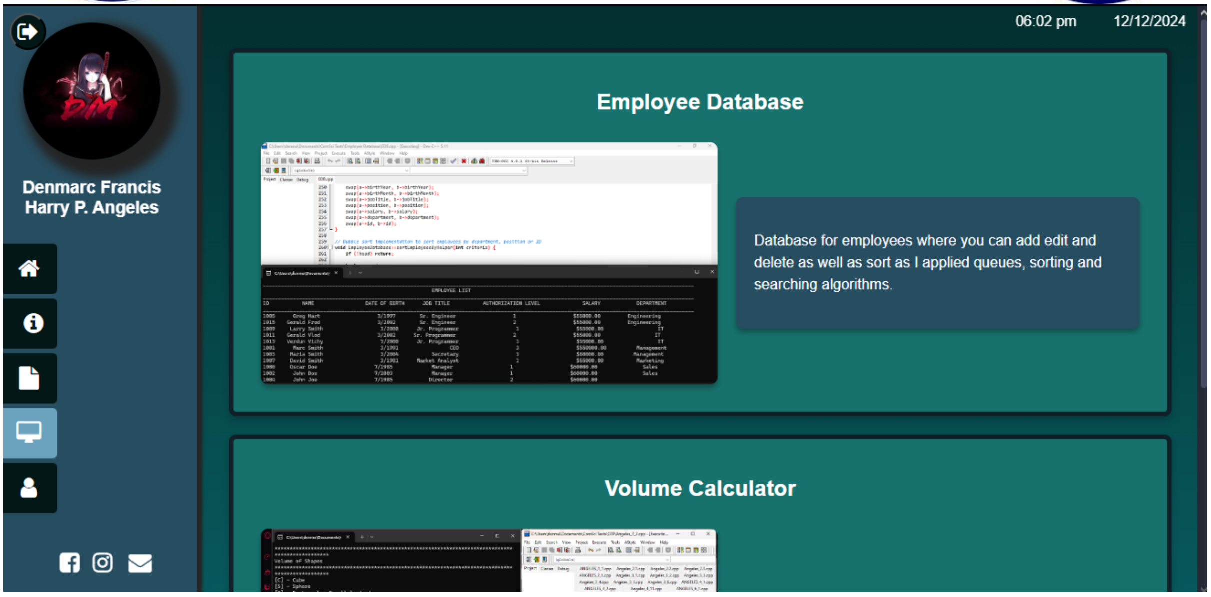
Hi, I'm Denmarc, a Freelance Web Programmer passionate about crafting innovative and functional web solutions. With experience in building web-based databases, static websites, and dynamic applications, I strive to bring ideas to life through clean code and user-centric design.

"Life isn't just about surviving, it's also about leaving behind a legacy."

Denmarc Angeles | Fullstack Developer

©2024 Denmarc Francis Harry P. Angeles - All Rights Reserved.

School of Information Technology Education
BACHELOR OF SCIENCE IN COMPUTER SCIENCE



06:02 pm 12/12/2024

Denmarc Francis Harry P. Angeles

Employee Database

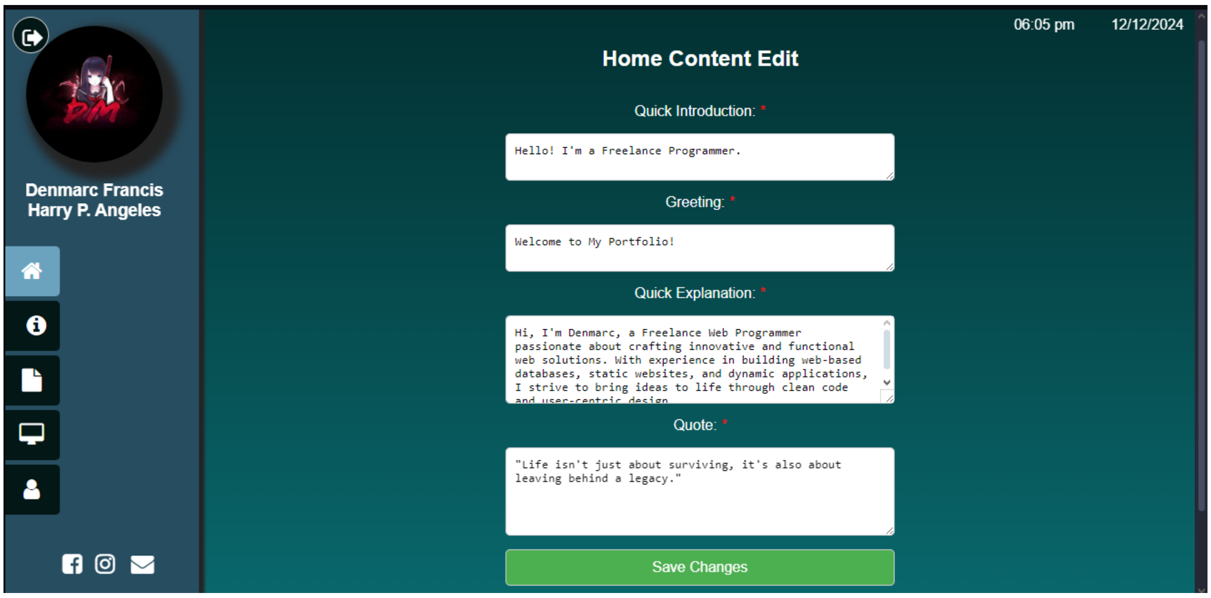
```
    swap(a->birthYear, b->birthYear);
    swap(a->birthMonth, b->birthMonth);
    swap(a->jobTitle, b->jobTitle);
    swap(a->position, b->position);
    swap(a->salary, b->salary);
    swap(a->department, b->department);
    swap(a->id, b->id);
}

// Bubble sort implementation to sort employees by department, position or ID
void EmployeeDatabase::sortEmployeesByHelper(int criteria) {
    if (!head) return;
```

```
                                   EMPLOYEE LIST
ID        NAME              DATE OF BIRTH    JOB TITLE       AUTHORIZATION LEVEL        SALARY        DEPARTMENT
1005    Greg Hart               3/1997      Sr. Engineer             1               $55000.00      Engineering
1015    Gerald Fred             3/2002      Sr. Engineer             2               $55000.00      Engineering
1009    Larry Smith             3/2000     Jr. Programmer             1               $55000.00              IT
1011    Gerald Vlad             3/2002     Sr. Programmer            2               $55000.00             IT
1013    Verdun Vichy            3/2000     Jr. Programmer             1               $55000.00              IT
1001    Marc Smith              3/1991                CEO             3               $550000.00     Management
1003    Maria Smith             3/2004          Secretary             3               $80000.00      Management
1007    David Smith             3/1981     Market Analyst             1               $55000.00       Marketing
1000    Oscar Doe               7/1985           Manager             1               $60000.00          Sales
1002    John Doe                7/2003           Manager             1               $60000.00          Sales
1004    John Joe                7/1985          Director             2               $60000.00
```

Database for employees where you can add edit and delete as well as sort as I applied queues, sorting and searching algorithms.

Volume Calculator

```
Volume of Shapes
[C] - Cube
[S] - Sphere
```



06:05 pm 12/12/2024

Denmarc Francis Harry P. Angeles

Home Content Edit

Quick Introduction: *

Hello! I'm a Freelance Programmer.

Greeting: *

Welcome to My Portfolio!

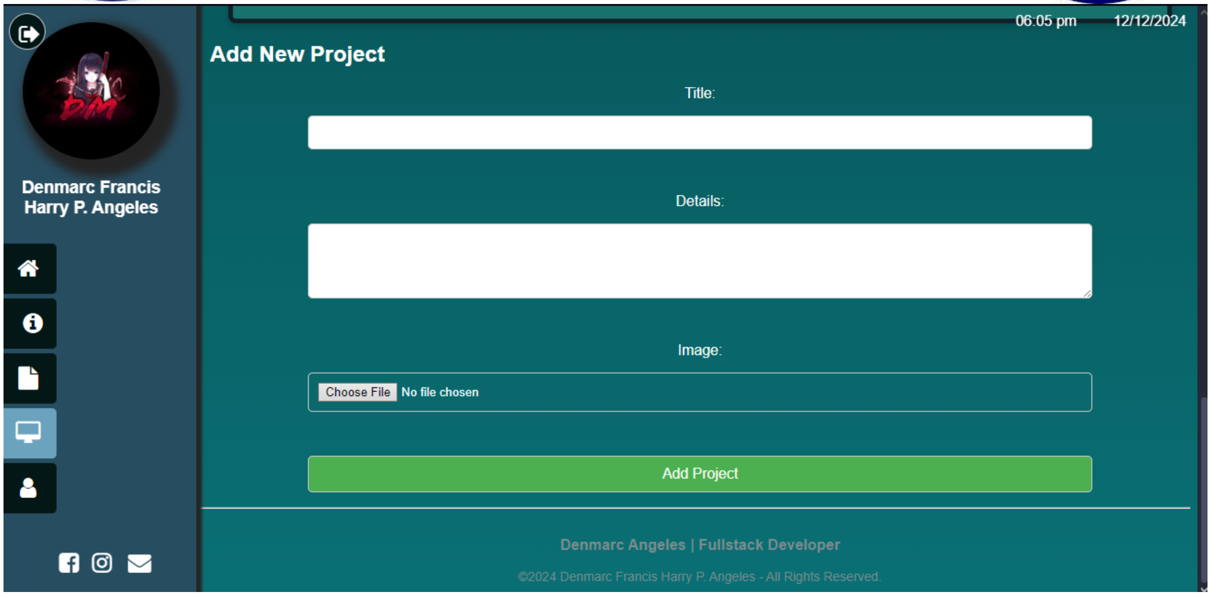
Quick Explanation: *

Hi, I'm Denmarc, a Freelance Web Programmer passionate about crafting innovative and functional web solutions. With experience in building web-based databases, static websites, and dynamic applications, I strive to bring ideas to life through clean code and user-centric design

Quote: *

"Life isn't just about surviving, it's also about leaving behind a legacy."

Save Changes



06:05 pm 12/12/2024

Denmarc Francis Harry P. Angeles

Add New Project

Title:

Details:

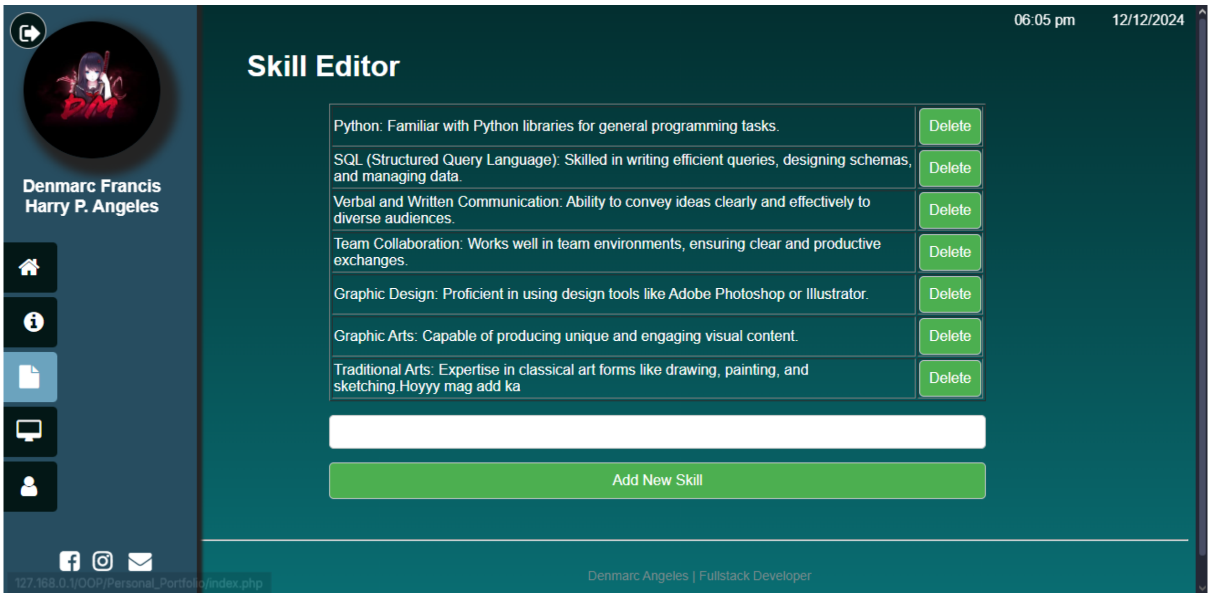
Image:

Choose File No file chosen

Add Project

Denmarc Angeles | Fullstack Developer

©2024 Denmarc Francis Harry P. Angeles - All Rights Reserved.



06:05 pm 12/12/2024

Denmarc Francis Harry P. Angeles

Skill Editor

Python: Familiar with Python libraries for general programming tasks.	Delete
SQL (Structured Query Language): Skilled in writing efficient queries, designing schemas, and managing data.	Delete
Verbal and Written Communication: Ability to convey ideas clearly and effectively to diverse audiences.	Delete
Team Collaboration: Works well in team environments, ensuring clear and productive exchanges.	Delete
Graphic Design: Proficient in using design tools like Adobe Photoshop or Illustrator.	Delete
Graphic Arts: Capable of producing unique and engaging visual content.	Delete
Traditional Arts: Expertise in classical art forms like drawing, painting, and sketching.Hoyyy mag add ka	Delete

Add New Skill

127.168.0.1/OOP/Personal_Portfolio/index.php

Denmarc Angeles | Fullstack Developer

Testing and Debugging

During the testing phase, various data inputs and repeated button presses were used to identify potential bugs within the system. 4 issues were discovered:

1. **Duplicate Submissions:** The contact form and potentially other forms exhibited a bug where submitting the form resulted in two identical submissions being processed instead of a single submission.
2. **Unauthorized File Modification:** A critical bug was found where the guest contact form unexpectedly modified files within the `content` directory. This is a severe security vulnerability, indicating a flaw in the input validation or access control mechanisms.
3. **Index Deletion:** In addition to unauthorized modification, the testing revealed that the guest contact form also deleted indices (lines) from certain text files in the `content` directory.
4. **Blank Submission Errors:** The system failed to prevent blank submissions in fields requiring text input. This indicates a lack of input validation.

Challenges and Solutions

The identified bugs—duplicate submissions, unauthorized file modification, index deletion, and blank submission errors—were resolved through a code review process. The review uncovered redundant code segments and the use of incorrect methods, leading to the identified issues. Corrective actions included revising specific code sections to eliminate the root causes of these errors and prevent their recurrence.

Specifically, the duplicate submission issue was addressed by identifying and removing redundant scripts within the form submission handlers. This likely involved eliminating duplicate calls to the database or file writing functions, ensuring that each form submission triggers the intended action only once.

The unauthorized file modification and index deletion bugs, which represented significant security vulnerabilities, were resolved by carefully examining the FileHandling class and the methods used to process guest contact form submissions. The review likely revealed that the guest user had unintended write access to the content directory. Corrective actions involved restricting access to this directory, ensuring that only authorized users (administrators) have write permissions. This may have involved implementing stricter access control checks within the PHP code that handles file operations.

Finally, the blank submission errors were resolved by adding input validation to the relevant forms. This likely involved adding both client-side (JavaScript) validation to provide immediate feedback to the user and server-side (PHP) validation to ensure data integrity before processing the form data. The server-side validation would prevent the processing of forms with missing required fields.

The improvements to the FileHandling class, specifically addressing repeated submissions for content addition, suggest that redundant or incorrectly implemented methods were identified and corrected. This likely involved consolidating or refactoring the code to ensure that content addition operations are performed consistently and accurately. The remediation process demonstrates a good understanding of debugging techniques and the importance of writing clean, efficient, and secure code.

Conclusion

This project successfully developed a dynamic personal portfolio website, demonstrating proficiency in various web development technologies. The use of HTML, CSS, and JavaScript for the front-end provided a visually appealing and user-friendly interface. The back-end, implemented using PHP and leveraging file handling for content management, allowed for efficient and flexible updates. The integration of PHPMailer facilitated seamless email communication. While the project initially presented challenges such as duplicate submissions, unauthorized file modifications, and inadequate input validation, a thorough code review and subsequent revisions effectively addressed these issues, resulting in a more robust and secure system.

The project demonstrates a solid understanding of fundamental web development principles, including front-end design, back-end logic, database interaction (albeit simplified through file handling), and security considerations. Future enhancements could include database integration for improved scalability and more sophisticated user authentication mechanisms. Overall, this project serves as a valuable demonstration of practical web development skills and problem-solving abilities.

References

https://www.w3schools.com

- **CSS designs and template**
- **HTML tags**
- **PHP manuals**

https://sourceforge.net/projects/phpmailer.mirror/

- source file for PHPMailer

https://www.youtube.com/watch?v=GmCFeLhA-fA&t=423s

- How to use PHPMailer

https://www.youtube.com/watch?v=bOqTCDfc7Tk

- Form Handling Tutorial in PHP

https://www.youtube.com/playlist?list=PLyKBLKYqadGkqb70sP212RBP5Pqao7vgq

- PHP File Systems – Beginners Guide

https://www.youtube.com/watch?v=JaRq73y5MJk

- File Uploading Tutorial in PHP