WEB APPLICATION SECURITY ASSIGNMENT

GROUP 9
COURSE CODE: CSS

DESIGN AND DEVELOP COURSE REGISTRATION WEB APPLICATION

GROUP MEMBERS

1. SURAJ GIMBA 2019/1/75579CS

2. APENDA GODWIN TERKAA 2019/1/77454CS

3. AJIBOYE ANTHONY AKINYEMI 2019/1/75097CS

4. OKUNADE MOHAMMED 2019/1/76165CS

5. ATHOT SETH AUDU 2019/1/75255CS

GENERAL OVERVIEW:

This web page serves as an interface for students to register for courses within the Cyber Security Science department at the Federal University of Technology. It provides home page, login page, sections for displaying student details, course lists for both first and second semesters, validation checkboxes, and a registration box for selecting courses.

Homepage Documentation:

1. Overview:
   This web page serves as the homepage for the Futminna Course Registration portal. It presents an introduction to the platform, including the university's motto and a welcome message.

2. Structure:
   - Document Type Declaration (`<!DOCTYPE html>`): Specifies the HTML version and type.
   - HTML Structure:
     - `<html>`: Root element of the HTML document.
     - `<head>`: Contains metadata and external resources.
     - `<body>`: Contains visible content and scripts.

3. Content:
   - Header (`<header>`): Displays the university logo and navigation menu.
     - Logo (`<a class="logo">`): Represents the university name or logo.
     - Navigation Menu (`<nav id="navmenu">`): Provides links to different pages of the website.
   - Main Section (`<main>`): Contains the hero section with a background image and welcome message.
     - Hero Section (`<section id="hero">`): Displays the university motto and a welcome message.

4. Styling:
   - CSS Files (`main.css`, `bootstrap.min.css`, `aos.css`, `glightbox.min.css`, `swiper-bundle.min.css`): Provide styling rules for the layout and elements of the homepage.

5. Functionality:
   - The homepage serves as an introduction to the course registration portal, providing users with essential information about the platform.

6. Technologies Used:
   - HTML: Markup language for structuring the homepage.
   - CSS: Stylesheet language for styling the layout and elements.
   - JavaScript: Scripting language for implementing dynamic functionalities.

7. Accessibility and Responsiveness:
   - The homepage is designed to be accessible and responsive across different devices and screen sizes.

8. Maintenance:
   - The code can be maintained by updating content, adjusting styling, and ensuring compatibility with evolving web standards.

9. Limitations:
- The homepage may require further enhancements for user engagement, such as interactive elements or additional sections.

Login Page Documentation:

1. Overview:
   This web page serves as a login interface for users to access the course registration portal. It provides input fields for entering the username and password.

2. Structure:
   - Document Type Declaration (`<!DOCTYPE html>`): Specifies the HTML version and type.
   - HTML Structure:
     - `<html>`: Root element of the HTML document.
     - `<head>`: Contains metadata and external resources.
     - `<body>`: Contains visible content and scripts.

3. Content:
   - Header (`<h1>`): Displays the title "Login" at the top of the page.
   - Form (`<form id="loginForm">`): Allows users to input their username and password for authentication.
   - Username Input (`<input type="text" id="username" name="username" required>`): Field for entering the username.
   - Password Input (`<input type="password" id="password" name="password" required>`): Field for entering the password.
   - Submit Button (`<button type="submit">Login</button>`): Button for submitting the login credentials.

4. Styling:
   - CSS File (`login.css`): Provides styling rules for the layout and elements of the login page.

5. Functionality:
   - Users can input their username and password to authenticate and access the course registration portal.

6. Technologies Used:
   - HTML: Markup language for structuring the login page.
   - CSS: Stylesheet language for styling the layout and elements.
   - JavaScript: Scripting language for implementing dynamic functionalities.

7. Accessibility and Responsiveness:
   - The login page is designed to be accessible and responsive across different devices and screen sizes.

8. Maintenance:
   - The code can be maintained by updating styling, adding validation mechanisms, and ensuring compatibility with evolving web standards.

9. Limitations:
- The login page may require further enhancements for user authentication, such as integrating with a backend server for verification.

School Fees Payment Page Documentation:
1. Overview:
   This web page serves as a course registration application with a payment form. It allows users to input their personal details, select their course preferences, and make payments.

2. Structure:
   - Document Type Declaration (`<!DOCTYPE html>`): Specifies the HTML version and type.
   - HTML Structure:
     - `<html>`: Root element of the HTML document.
     - `<head>`: Contains metadata and external resources.
     - `<body>`: Contains visible content and scripts.

3. Content:
   - Navigation Bar (`<nav class="nav">`): Provides navigation options.
   - Payment Form (`<form class="payment-form">`): Allows users to input personal details, select courses, and make payments.
     - First Page (`<div class="first_page" id="first_page">`): Displays fields for personal details and course selection.
     - Upload Image (`<input type="file" id="img" accept="image/jpeg, image/jpg, image/png">`): Allows users to upload an image.
   - Personal Details Fields: Input fields for first name, last name, faculty, department, and level.
   - Continue Button (`<button class="first-page-btn">`): Allows users to proceed to the next page.

- Second Page (`<div class="second_page" id="second_page">`): Displays fields for email, phone number, and payment.
  - Email and Phone Number Fields: Input fields for email and phone number.
  - Payment Field (`<input type="text" placeholder="Amount to be paid" required id="money">`): Allows users to input the amount to be paid.
  - Back Button (`<button class="second-page-btn">`): Allows users to go back to the first page.
  - Pay Button (`<button class="pay-btn" type="button">`): Allows users to initiate the payment process.

4. Styling:
   - CSS File (`index.css`): Provides styling rules for the layout and elements of the course registration application.

5. Functionality:
   - Users can input their personal details, select courses, and make payments.
   - The application provides visual feedback for successful payments.

6. Technologies Used:
   - HTML: Markup language for structuring the course registration application.
   - CSS: Stylesheet language for styling the layout and elements.
   - JavaScript (`index.js`): Scripting language for implementing dynamic functionalities.

7. Accessibility and Responsiveness:
   - The application is designed to be accessible and responsive across different devices and screen sizes.

8. Maintenance:
- The code can be maintained by updating styling, adding validation mechanisms, and ensuring compatibility with evolving web standards.

Course Registration Page Documentation:

1. Overview:

   This web page serves as a course registration interface for students of the Federal University of Technology, specifically for the Cyber Security Science department. It allows students to select courses for the upcoming semester and provides details about the selected courses.

2. Structure:
   - Document Type Declaration (`<!DOCTYPE html>`): Specifies the HTML version and type.
   - HTML Structure:
     - `<html>`: Root element of the HTML document.

- `<head>`: Contains metadata and external resources.
- `<body>`: Contains visible content and scripts.

3. Content:
- Navigation (`<nav class="navigation">`): Navigation bar for accessing different sections of the page.
- Main Body (`<div class="main_body">`): Container for the course registration page content.
- Course Registration Page (`<div class="courseReg_page">`): Main section for course registration details.
- Header (`<header class="header">`): Displays university and department information along with student details and passport.
- First Semester Course List (`<div class="firstSem_list">`): Displays the list of first-semester courses along with their details and total credit units.
- Second Semester Course List (`<div class="secondSem_list">`): Displays the list of second-semester courses along with their details and total credit units.
- Validation Checkboxes (`<div class="validation">`): Checkboxes for validation by level adviser, HOD, dean, and registrar.
- Print Button (`<div class="printBtn">`): Buttons for navigating back to the home page and printing the course registration details.
- Container (`<div class="container">`): Container for the course registration form.
- Navigation (`<nav class="nav">`): Navigation bar displaying the current path.
- Course Registration Box (`<div class="register-box">`): Contains a form for selecting the session and a list of available courses for registration.

4. Styling:
- CSS Files (`course_reg.css`, `index.css`): Provide styling rules for layout, elements, and responsiveness.

5. Functionality:
- Users can select courses for registration for the specified session.
- The page dynamically updates course lists and credit units based on user selections.
- Validation checkboxes allow designated personnel to verify the registration.

6. Technologies Used:
- HTML: Markup language for structuring the course registration page.
- CSS: Stylesheet language for styling the layout and elements.
- JavaScript: Scripting language for implementing dynamic functionalities.

7. Accessibility and Responsiveness:
- The course registration page is designed to be accessible and responsive across different devices and screen sizes.

8. Maintenance:
- The code can be maintained by updating styling, adding new courses, and ensuring compatibility with evolving web standards.

9. Limitations:
- The page may require further enhancements for user authentication and backend integration for storing registration data.

Backend Documentation:

1. Overview:
The backend of the Futminna Course Registration portal serves as the server-side component responsible for handling data processing, database interactions, and business logic. It facilitates the registration process for students and ensures the integrity and security of the system.

2. Structure:
- PHP Files: Contains server-side scripts written in PHP for processing requests and interacting with the database.
- Database: Stores essential data, including student information, course details, and registration records.
- Configuration Files: Contains settings and configurations for connecting to the database and other server settings.

3. Functionality:
- User Authentication:
  - Authenticate User Credentials: PHP scripts authenticate user credentials (username and password) provided during login.
  - Session Management: Maintain user sessions to track authenticated users and provide secure access to protected resources.

- Course Registration:
  - Retrieve Course Information: PHP scripts retrieve course details (e.g., course code, title, units) from the database for display on registration pages.
  - Process Registration Requests: Handle requests from users to register for courses, validating inputs and updating database records accordingly.
  - Credit Unit Calculation: Calculate total credit units for each semester based on the selected courses and update the student's record in the database.

- User Profile Management:
  - Update User Information: Allow users to update their profile information (e.g., name, department, level) through PHP scripts.
  - Upload Passport: Handle file uploads for user passport photos, storing them in a designated directory on the server and updating the corresponding database records.

- Access Control:
    - Role-based Access Control (RBAC): Implement access control mechanisms based on user roles (e.g., student, level adviser, HOD) to restrict access to certain functionalities and pages.
    - Authorization Checks: Perform authorization checks to ensure that users have the necessary permissions before allowing access to specific resources or performing actions.

4. Database Interactions:
   - Establish Database Connection: PHP scripts establish a connection to the MySQL database using appropriate credentials and configurations.
   - Execute SQL Queries: Execute SQL queries to retrieve, insert, update, or delete data from the database based on user requests and application logic.
   - Handle Database Errors: Handle errors and exceptions that may occur during database interactions, providing appropriate error messages and logging details for debugging purposes.

5. Security Measures:
   - Sanitize Inputs: Sanitize user inputs to prevent SQL injection attacks and cross-site scripting (XSS) vulnerabilities.
   - Parameterized Queries: Use parameterized queries or prepared statements to prevent SQL injection attacks and ensure the integrity of database operations.
   - Password Hashing: Hash and salt user passwords before storing them in the database to enhance security and protect against password breaches.

6. Technologies Used:
   - PHP: Server-side scripting language for backend development.
   - MySQL: Relational database management system (RDBMS) for storing and managing data.
   - Apache or Nginx: Web server software for hosting the PHP backend and serving web pages to clients.
   - IDE or Text Editor: Development tools for writing, editing, and debugging PHP scripts.

Prerequisites
- Web server (e.g., Apache, Nginx) with PHP support
- MySQL database server
- PHP version 7.0 or higher
- Composer (for managing PHP dependencies)

Installation
1. Clone the repository to your local machine:
```
   git clone https://github.com/your_username/futminna-course-registration.git
```

2. Navigate to the project directory:
```
   cd futminna-course-registration
```

3. Install PHP dependencies using Composer:
   composer install

4. Import the SQL database schema:
   - Use the provided SQL dump file (`database.sql`) to create the necessary tables in your MySQL database.

5. Configure the database connection:
   - Update the database configuration settings in the `.env` file with your MySQL database credentials.

6. Set up your web server:
   - Configure your web server to serve the `public` directory as the document root.
   - Ensure that PHP is configured correctly to handle web requests.

Usage
1. Access the application through your web browser.
   - Navigate to `http://localhost/home.html` (replace `localhost` with your server hostname or IP address).

2. Log using your credentials.
   - student will have access to different functionalities.

3. Explore the various features of the application, such as course registration, user management, and administrative tasks.
7. Maintenance:
   - Regular Updates: Keep the backend codebase up-to-date with security patches, bug fixes, and enhancements to ensure optimal performance and security.
   - Database Maintenance: Perform routine database maintenance tasks, such as backup, optimization, and data cleanup, to maintain data integrity and performance.
   - Error Logging and Monitoring: Implement error logging and monitoring mechanisms to track and resolve issues proactively, ensuring smooth operation of the system.

8. Documentation and Testing:
   - Document APIs and Endpoints: Document APIs and endpoints exposed by the backend, including input parameters, response formats, and authentication requirements.
   - Unit Testing: Develop and execute unit tests to verify the functionality of individual backend components and ensure reliability and correctness.
   - Integration Testing: Conduct integration testing to validate the interaction between different backend modules and external systems, such as the database.

9. Limitations:
- Scalability: The backend architecture should be designed with scalability in mind to accommodate increasing user loads and data volumes over time.
- Security Vulnerabilities: Continuous monitoring and proactive measures are necessary to address emerging security threats and vulnerabilities in the backend code and infrastructure.

SECURITY MECHANISMS AND THREAT MODELING IMPLEMENTED

Security Mechanisms Implemented:

1. Password Hashing and Salting:
   - Passwords stored in the database are hashed using a strong cryptographic hashing algorithm (e.g., bcrypt) with an added salt value.
   - This prevents attackers from obtaining plaintext passwords even if they manage to access the database.

2. Input Sanitization:
   - User inputs are sanitized to prevent SQL injection attacks and cross-site scripting (XSS) vulnerabilities.
   - Special characters and malicious code are filtered or escaped before processing user input.

3. Parameterized Queries:
   - SQL queries used for database interactions are parameterized or prepared statements.
   - This prevents SQL injection attacks by separating SQL logic from user input values, making it impossible for attackers to manipulate SQL queries.

4. Role-Based Access Control (RBAC):
   - Access to system functionalities and resources is restricted based on user roles (e.g., student, level adviser, HOD).
   - Each role is assigned specific permissions, allowing users to access only the functionalities relevant to their roles.

5. Secure Session Management:
   - Sessions are securely managed to maintain user authentication and track user activity.
   - Session IDs are generated using cryptographically secure random number generators.
   - Session cookies are marked as secure and HTTPOnly to prevent session hijacking and cross-site scripting attacks.

6. File Upload Security:

   - File uploads, such as user profile pictures, are restricted to specific file types and sizes.
   - Uploaded files are scanned for malicious content using antivirus software or file integrity checks.
   - Uploaded files are stored in a separate directory outside the web root to prevent direct access by attackers.

7. HTTPS Encryption:
   - All communication between the client and server is encrypted using HTTPS protocol.
   - SSL/TLS certificates are used to establish secure connections, encrypting data transmitted over the network and preventing eavesdropping.

Threat Modeling:
1. SQL Injection:
   - Threat: Attackers may attempt to execute malicious SQL queries by injecting SQL code into input fields.
   - Mitigation: Use parameterized queries or prepared statements to sanitize user inputs and prevent SQL injection attacks.

2. Cross-Site Scripting (XSS):
   - Threat: Attackers may inject malicious scripts into web pages viewed by other users, leading to session hijacking or data theft.
   - Mitigation: Implement input validation and output encoding to sanitize user inputs and prevent XSS vulnerabilities.

3. Session Hijacking:
   - Threat: Attackers may steal session IDs or cookies to impersonate authenticated users and gain unauthorized access to the system.
   - Mitigation: Use secure session management techniques, such as generating random and cryptographically secure session IDs, marking cookies as secure and HTTPOnly, and implementing session expiration and reauthentication mechanisms.

4. File Upload Vulnerabilities:
   - Threat: Attackers may upload malicious files containing malware or executable scripts to the server, leading to system compromise.
   - Mitigation: Restrict file upload permissions to specific file types and sizes, scan uploaded files for malicious content, and store uploaded files in a secure directory with limited access permissions.

5. Insecure Direct Object References (IDOR):
   - Threat: Attackers may manipulate object references in URLs or hidden fields to access unauthorized resources or perform unauthorized actions.
   - Mitigation: Implement access controls and authorization checks to validate user permissions before accessing or modifying sensitive resources. Use unique identifiers

or tokens to reference objects and avoid exposing internal resource identifiers directly in URLs.

By implementing these security mechanisms and considering potential threats through threat modeling, the Course Registration system is better equipped to protect user data, prevent unauthorized access, and mitigate security risks effectively.

USAGE INSTRUCTIONS

PREREQUISITES

- Web server (e.g., Apache, Nginx) with PHP support
- MySQL database server
- PHP version 7.0 or higher
- Composer (for managing PHP dependencies)

INSTALLATION

1. Navigate to the project directory:
```
cd Group9 work
```

2. Install PHP dependencies using Composer:
```
composer install
```

3. Import the SQL database schema:
   - Use the provided SQL dump file (`database.sql`) to create the necessary tables in your MySQL database.

4. Configure the database connection:
   - Update the database configuration settings in the `.env` file with your MySQL database credentials.

5. Set up your web server:
   - Configure your web server to serve the `public` directory as the document root.
   - Ensure that PHP is configured correctly to handle web requests.

USAGE

1. Access the application through your web browser.
   - Navigate to `http://localhost/group9_work/home.html` (replace `localhost` with your server hostname or IP address).

2. Log using your credentials.
   - student will have access to different functionalities.

3. Explore the various features of the application, such as course registration, user management, and administrative tasks.