virtual lean workshop platform

2024, **BIT**

Web Technology

Aloys Shema RUKUNDO

REG NO: 222014243

5/23/2024

PROJECT NAME: VIRTUAL LEAN WORKSHOP PLATFORM

PROJECT DESCRIPTION DOCUMENT

The Virtual Lean Workshop Platform is a web-based system designed to facilitate the management and operation of workshops. Developed using HTML, JavaScript, CSS, Bootstrap, and PHP, the platform connects to a MySQL database to handle various data and processes. The system supports workshop administration, instructor management, user management, module management, attendee tracking, and problem resolution.

Planning

The goal of this project was to develop a virtual lean workshop platform that can be used to track workshop activities and participant involvement, and to generate comprehensive reports. The main objectives of the platform are:

User-Friendly Interface: Create an intuitive interface that allows easy management of workshop activities and accurate tracking of schedules.

Efficiency: Automate manual tracking processes to reduce errors and save time, ensuring quick and accurate data reads.

Future Enhancements: Improve the user interface and adopt new technologies to enhance reporting techniques and simplify processes.

Design

Our Virtual Lean Workshop Platform features a user-friendly and aesthetically pleasing interface. The system comprises three main interfaces based on user roles: Admin, Instructor, and Attendee.

Interfaces

Admin Interface:

Admins can log in using a secure form that verifies email and password against the database.

Upon successful login, admins are directed to the Admin Dashboard, where they can access real-time system information and manage workshops, instructors, attendees, and courses.

Admin functionalities include recording new workshops, updating existing workshops, viewing and managing instructor information, and assigning instructors to modules.

Instructor Interface:

Instructors log in to access their dedicated interface.

They can update their information, view assigned modules, and manage attendee interactions.

Instructors can also record and update attendance for modules they are teaching.

Attendee Interface:

Attendees log in to view and update their information.

They can access workshop details, subscribe to modules, and interact with the platform based on their subscription.

Attendees can also report issues and track their attendance.

Functional Requirements

Data Management: Admins can perform CRUD (Create, Read, Update, Delete) operations on workshops, instructors, attendees, and modules.

Attendance Tracking: Instructors can record and update attendance for each module session.

Issue Reporting: Attendees can report issues, and admins can reply to these reports.

Non-Functional Requirements

User-Friendly Interface: Easy navigation and intuitive design.

Security: Only authorized users can access the system.

Performance: Fast processing of data and quick retrieval times.

Development

The Virtual Lean Workshop Platform was developed using HTML, CSS, JavaScript, Bootstrap, and PHP. The development environment utilized Sublime Text Editor for writing and editing code. Key tools and technologies used include:

HTML & Bootstrap: For creating the front-end with a rich set of user interface components.

JavaScript: For client-side scripting and interactivity.

PHP: For server-side scripting and connecting to the MySQL database.

MySQL: Database management, designed using XAMPP server.

Sublime Text: Source code editor with features like syntax highlighting and code completion.

XAMPP: Local server environment for testing and development.

4. Testing

The system's performance was tested and monitored by using manual testing methods to ensure quick response time and data accuracy. During this stage, we found some serious defects and bugs and worked our best to fix them. Firstly, forms were able to submit empty text fields into database to solve this problem we first checked if user filled some data in text field to process the process of inserting that new record in database. Secondly, clicking delete or update buttons without selecting a record to delete or update used to cause bugs in the background, to solve this problem before running update or delete code we first check a condition to see if there is exactly a record selected and if yes that specific record would now be deleted or updated and if no record selected the message is displayed telling a user to select a record to delete or update. After fixing the above two bugs system now runs correctly without any bug and it have high accuracy of data as we expected before developing this virtual lean workshop platform.

5.Deployment

Virtual lean workshop platform was deployed on a local laptop for testing and performance monitoring. Starting from index webpage which consist of three categories where you chou who you are in system then , it direct to login or register form only user who type in Email and password that matches any user database record if not it display on your screen an error message telling you that you enter incorrect username and or password or user not found. We also filled other remaining forms with data and submit them the result was what we expected because data were submitted into database tables exactly the way we wanted. The key functionality of our virtual lean workshop platform system was to track information's of our guests, his or her daily work flow.

Overview of virtual lean workshop platform

Overview

The Virtual Lean Workshop Platform is a web-based system designed to facilitate the management and operation of workshops. Developed using HTML, JavaScript, CSS, Bootstrap, and PHP, the platform connects to a MySQL database to handle various data and processes. The system supports workshop administration, instructor management, user management, module management, attendee tracking, and problem resolution.

System Architecture

The system's architecture is based on a relational database with several interconnected tables. The core functionalities include managing workshops, instructors, users, modules, attendees, and attendance. Additionally, it handles applications, issues reported by attendees, and their respective resolutions.

Database Structure

Workshop Table

```
CREATE TABLE workshop (
  wshopID INT AUTO_INCREMENT PRIMARY KEY,
  wname VARCHAR(50),
  physical_addres VARCHAR(25),
  url VARCHAR(50),
  email VARCHAR(100),
  phone VARCHAR(13),
  password VARCHAR(255) NOT NULL,
  created_at TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  createdBy VARCHAR(20) NOT NULL,
  updated_at TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()
);
```

- wshopID: Unique identifier for each workshop.

- wname: Name of the workshop.

-physical_address: Physical address of the workshop.

- url: Workshop's URL.

- email: Contact email of the workshop.

- phone: Contact phone number of the workshop.

- password: Password for workshop admin access.

- created_at: Timestamp when the workshop was created.

- createdBy: User who created the workshop entry.

- updated_at: Timestamp when the workshop entry was last updated.

Instructor Table

```
CREATE TABLE instructor (
  instructorID INT AUTO_INCREMENT PRIMARY KEY,
  wshopID INT,
  name VARCHAR(50),
  phone VARCHAR(24),
  email VARCHAR(30),
  FOREIGN KEY (wshopID) REFERENCES workshop(wshopID),
  created_at TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  updated_at TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  created_by VARCHAR(30) DEFAULT NULL,
  updated_by VARCHAR(30) DEFAULT NULL,
  password VARCHAR(255) DEFAULT 'name'
);
```

- instructorID: Unique identifier for each instructor.

- wshopID: ID of the workshop the instructor is associated with.

- name: Name of the instructor.

- phone: Phone number of the instructor.

- email: Email address of the instructor.

- password: Password for instructor access.

- created_at: Timestamp when the instructor entry was created.

- updated_at: Timestamp when the instructor entry was last updated.

- created_by: User who created the instructor entry.

- updated_by: User who last updated the instructor entry.

users Table

```
CREATE TABLE users (
  usr_id INT(11) NOT NULL,
  email VARCHAR(100) NOT NULL,
  userName VARCHAR(30) DEFAULT NULL,
  password VARCHAR(255) DEFAULT NULL,
  createdBy VARCHAR(30) NOT NULL,
  created_at TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  updated_at TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()
);
```

- usr_id: Unique identifier for each user.
- email: Email address of the user.
- userName: Username for the user.
- password: Password for user access.
- createdBy: User who created the user entry.
- created_at: Timestamp when the user entry was created.
- updated_at: Timestamp when the user entry was last updated.

Instructor Modules Table

```
CREATE TABLE instructor_moules (
  id INT AUTO_INCREMENT PRIMARY KEY,
  instructorID INT NOT NULL,
  moduleID INT NOT NULL,
  assignedBy VARCHAR(30) NOT NULL,
  createdAt TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  updatedAt TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  updatedBy VARCHAR(30) NOT NULL
);
```

- id: Unique identifier for each instructor-module assignment.
- instructorID: ID of the instructor assigned to the module.
- moduleID: ID of the module assigned to the instructor.
- assignedBy: User who assigned the module to the instructor.
- createdAt: Timestamp when the assignment was created.
- updatedAt: Timestamp when the assignment was last updated.
- updatedBy: User who last updated the assignment.

Modules Table

```
CREATE TABLE modules (
  ModuleID INT AUTO_INCREMENT PRIMARY KEY,
  Course_name VARCHAR(50) UNIQUE KEY,
  Course_code VARCHAR(6) UNIQUE KEY,
  created_at TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  updated_at TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  created_by VARCHAR(30) DEFAULT NULL,
  updated_by VARCHAR(30) DEFAULT NULL
);
```

- ModuleID: Unique identifier for each module.
- Course_name: Name of the course.

-Course_code: Unique code for the course.

- created_at: Timestamp when the module was created.
- updated_at: Timestamp when the module was last updated.
- created_by: User who created the module.
- updated_by: User who last updated the module.

Attendees Table

```
CREATE TABLE attendees (
  attendeeID INT AUTO_INCREMENT PRIMARY KEY,
  wshopID INT,
  reg_no VARCHAR(8) UNIQUE KEY,
  name VARCHAR(50),
  phone VARCHAR(20),
  email VARCHAR(50),
  created_at TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  updated_at TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  created_by INT(11) DEFAULT NULL,
  updated_by VARCHAR(20) DEFAULT NULL,
  password VARCHAR(255) NOT NULL DEFAULT 'reg_no',
  FOREIGN KEY (wshopID) REFERENCES workshop(wshopID)
);
```

- attendeeID: Unique identifier for each attendee.

-wshopID: ID of the workshop the attendee is associated with.

- reg_no: Registration number for the attendee.

- name: Name of the attendee.

- phone: Phone number of the attendee.

- email: Email address of the attendee.

- password: Password for attendee access.

- created_at: Timestamp when the attendee entry was created.

- updated_at: Timestamp when the attendee entry was last updated.

- created_by: User who created the attendee entry.

- updated_by: User who last updated the attendee entry.

Attendees Modules Table

```
CREATE TABLE attendees_modules (
  ID INT AUTO_INCREMENT PRIMARY KEY,
  attendeeID INT,
  moduleID INT,
  createdAt TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  updatedAt TIMESTAMP NULL DEFAULT NULL,
  createdBy VARCHAR(30) DEFAULT NULL,
  updatedBy VARCHAR(30) NOT NULL,
  FOREIGN KEY (attendeeID) REFERENCES attendees(attendeeID),
  FOREIGN KEY (moduleID) REFERENCES modules(moduleID)
);
```

- ID: Unique identifier for each attendee-module assignment.
- attendeeID: ID of the attendee assigned to the module.
- moduleID: ID of the module assigned to the attendee.
- createdAt: Timestamp when the assignment was created.
- updatedAt: Timestamp when the assignment was last updated.
- createdBy: User who created the assignment.
- updatedBy: User who last updated the assignment.

Attendees Attendance Table

```
CREATE TABLE attendees_attendance (
  att_id INT AUTO_INCREMENT PRIMARY KEY,
  moduleID INT,
  attendeeID INT,
  instructorID INT,
  attendance CHAR(1),
  datec DATE,
  created_at TIMESTAMP NOT NULL DEFAULT current_timestamp(),
  updated_at TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  created_by INT(11) DEFAULT NULL,
  updated_by INT(11) DEFAULT NULL,
 FOREIGN KEY (attendeeID) REFERENCES attendees(attendeeID),
FOREIGN KEY (moduleID) REFERENCES modules(moduleID)
);
```

- att_id: Unique identifier for each attendance record.
- moduleID: ID of the module.
- attendeeID: ID of the attendee.
- instructorID: ID of the instructor.
- attendance: Attendance status (e.g., 'P' for present, 'A' for absent).
- datec: Date of the attendance record.
- created_at: Timestamp when the attendance record was created.
- updated_at: Timestamp when the attendance record was last updated.
- created_by: User who created the attendance record.
- updated_by: User who last updated the attendance record.

Application Table

```
CREATE TABLE application (
  app_id INT AUTO_INCREMENT PRIMARY KEY,
  attendeeID INT,
  pay_method VARCHAR(50),
  amount VARCHAR(200),
  proof VARCHAR(200),
  FOREIGN KEY (attendeeID) REFERENCES attendees(attendeeID)
);
```

- app_id: Unique identifier for each application.
- attendeeID: ID of the attendee submitting the application.
- pay_method: Payment method used by the attendee.
- amount: Payment amount.
- proof: Proof of payment.

Problem Issued Table

```
CREATE TABLE problem_issued (
  id INT AUTO_INCREMENT PRIMARY KEY,
  reporter INT,
  email VARCHAR(30),
  problem VARCHAR(250),
  reported_at TIMESTAMP NOT NULL DEFAULT current_timestamp

() ON UPDATE current_timestamp(),
  FOREIGN KEY (reporter) REFERENCES attendees(attendeeID)
);
```

- id: Unique identifier for each problem report.
- reporter: ID of the attendee reporting the problem.
- email: Email address of the reporter.
- problem: Description of the problem.
- reported_at: Timestamp when the problem was reported.

Problem Reply Table

```
CREATE TABLE problem_reply (
  id INT AUTO_INCREMENT PRIMARY KEY,
  rep INT,
  reply VARCHAR(250),
  replied_at TIMESTAMP NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  FOREIGN KEY (rep) REFERENCES problem_issued(id)
);
```

- id: Unique identifier for each problem reply.
- rep: ID of the problem report being replied to.
- **reply**: Reply to the problem.
- replied_at: Timestamp when the reply was made.

System Workflow

User Registration and Login

1. Workshop Admin Registration: The admin registers a workshop by providing the necessary details such as workshop name, physical address, URL, email, and phone number. A password is also set during registration.

2. Instructor Registration: Instructors are added by the workshop admin. Instructors are associated with a workshop and provided with their contact details and a password.

3. User Registration: Users can register on the platform with their email, username, and password.

Module Management

1. Module Creation: Admins can create modules by providing course names and codes.

2. Assign Modules to Instructors: Admins assign modules to instructors who will be responsible for delivering the content.

Attendee Management

1. Attendee Registration: Attendees register for workshops by providing their details, including registration number, name, phone, and email.

2. Assign Modules to Attendees: Admins or instructors assign modules to attendees.

Attendance Management

1. Record Attendance: Instructors record attendance for each module session. Attendance status is marked, and the date is recorded.

Application and Payment

1. Submit Applications: Attendees submit applications for workshop participation, specifying payment methods and providing proof of payment.

Issue Reporting and Resolution

1. Report Issues: Attendees can report problems they encounter. Each problem is recorded with a description and the reporter's details.

2. Reply to Issues: Admins or designated users can reply to reported problems, providing resolutions or further instructions.

Frontend and Backend Integration

The frontend of the Virtual Lean Workshop Platform is developed using HTML, CSS, JavaScript, and Bootstrap for responsive design. The backend is powered by PHP, which handles data processing and communication with the MySQL database.

Conclusion

The Virtual Lean Workshop Platform offers a comprehensive solution for managing workshops, instructors, users, modules, and attendees. The well-defined database structure and seamless integration of frontend and backend technologies ensure an efficient and user-friendly experience for all stakeholders involved.