VAL.gg - Presentation Roadmap

1. Introduction

- **Project Title**: VAL.gg - Valorant Gaming Wiki
- **Purpose**: Final project demonstrating full-stack CRUD operations
- **Target**: Academic requirement showcasing web development skills
- **Brief Overview**: A Valorant-themed platform with user management and interactive features

2. What It Does

Core Purpose

- Gaming wiki platform focused on Valorant content
- User account management system
- Interactive community features through commenting
- Demonstration of complete CRUD functionality

Main Objectives

- Showcase Create, Read, Update, Delete operations
- Implement secure user authentication
- Provide interactive user experience
- Demonstrate database integration skills

4. Technical Framework Stack

Frontend Technologies

- **HTML5**: Structure and semantic markup
- **CSS3**: Styling and responsive design
- **Bootstrap**: CSS framework for responsive design and UI components
- **jQuery**: JavaScript library for AJAX requests

Backend Technologies

- **PHP**: Server-side scripting and business logic
- **MySQL**: Database management and data storage
- **Apache**: Web server for hosting and request handling
- **jQuery**: JavaScript library for DOM manipulation

5. Key Features Demonstration

Authentication System

- **User Registration**: New account creation with validation
- **Secure Login**: Password authentication and session handling
- **Session Management**: 1-hour cookie expiration for security
- **Logout Functionality**: Proper session termination

CRUD Operations

- **Create**: User accounts, comments, profile data
- **Read**: User profiles, comment display, content viewing
- **Update**: Profile information, comment editing
- **Delete**: Comment removal

Security Features

- **Session Cookies**: 1-hour expiration for enhanced security
- **Input Validation**: Form data verification and sanitization
- **Error Handling**: Proper feedback for user actions
- **Data Protection**: Secure handling of user information

6. Database Structure

Tables and Relationships

- **Users Table**: Account information and credentials
- **Comments Table**: User-generated content and interactions

Data Management

- **Relationships**: Foreign key constraints and data integrity
- **Validation**: Server-side data verification
- **Security**: Protected against SQL injection and XSS

7. Live Demo Walkthrough

1. **User Registration**: Create new account
2. **Login Process**: Authenticate existing user
3. **Profile Management**: Edit user information
4. **Comment Interaction**: Create, edit, and delete comments
5. **Session Expiry**: 1-hour cookie timeout
6. **Responsive Design**: Mobile/tablet compatibility

1. User Registration (AJAX, PHP, Validation & Hashing)

```php
<?php
$username = $_POST['username'] ?? '';
$email = $_POST['email'] ?? '';
$age = $_POST['age'] ?? '';
$password = $_POST['password'] ?? '';
$confirmPassword = $_POST['confirm_password'] ?? '';

if (!is_numeric($age) || $age <= 0) {
    echo json_encode(['success' => false, 'message' => 'Age must be a positive number.']);
    exit;
}
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo json_encode(['success' => false, 'message' => 'Invalid email address.']);
    exit;
}
if ($password !== $confirmPassword) {
    echo json_encode(['success' => false, 'message' => 'Passwords do not match.']);
    exit;
}
$hashedPassword = password_hash($password, PASSWORD_BCRYPT);
$insert = mysqli_query($con, "INSERT INTO users (Username, Email, Age, Password) VALUES ('$username', '$email', '$age', '$hashedPassword')");
if ($insert) {
    echo json_encode(['success' => true, 'message' => 'Account Registered, Please Log in.']);
} else {
    echo json_encode(['success' => false, 'message' => 'Registration failed.']);
}
```

- **Input validation**
- **Secure password hashing**
- **AJAX response for user feedback**

2. Login Authentication (AJAX, PHP, Session)

```php
<?php
if (isset($_POST['username'], $_POST['password'])) {
    $username = mysqli_real_escape_string($con, $_POST['username']);
    $password = mysqli_real_escape_string($con, $_POST['password']);

    $result = mysqli_query($con, "SELECT * FROM users WHERE Username='$username'");
    $row = mysqli_fetch_assoc($result);

    if ($row && password_verify($password, $row['password'])) {
        $_SESSION['valid'] = $row['username'];
        $_SESSION['id'] = $row['id'];
        $_SESSION['start_time'] = time();
        echo json_encode(['success' => true]);
    } else {
        echo json_encode(['success' => false, 'message' => 'Invalid Credentials']);
    }
}
```

- **Secure password verification**
- **Session creation for login**
- **AJAX-based login feedback**

3. Session Timeout Security

```php
<?php
require_once(__DIR__ . "/config.php");

if (isset($_SESSION['start_time'])) {
    $elapsed_time = time() - $_SESSION['start_time'];
    if ($elapsed_time > SESSION_TIMEOUT) {
        session_unset();
        session_destroy();
        header("Location: login.php?error=session_expired");
        exit;
    } else {
        $_SESSION['start_time'] = time();
    }
}
```

- **1-hour session expiration for security**
- **Automatic logout and redirect**

4. Comment CRUD (AJAX, PHP, Ownership Check)

```php
<?php
if (!isset($_SESSION['id']) || empty($_SESSION['id'])) {
    http_response_code(401);
    echo "You must be logged in to comment.";
    exit;
}
$user_id = $_SESSION['id'];
$page_id = mysqli_real_escape_string($con, $_POST['page_id']);
$comment = mysqli_real_escape_string($con, $_POST['comment']);
$query = "INSERT INTO comments (user_id, page_id, comment) VALUES ('$user_id', '$page_id', '$comment')";
if (mysqli_query($con, $query)) {
    echo "success";
} else {
    http_response_code(500);
    echo "Error: " . mysqli_error($con);
}
```

- **Comment creation**
- **User authentication required**
- **Secure SQL insertion**

5. Edit & Delete Comments (Ownership & AJAX)

```php
<?php
if (!isset($_SESSION['id'])) {
    http_response_code(401);
    echo "Unauthorized";
    exit;
}
$comment_id = intval($_POST['comment_id']);
$new_comment = mysqli_real_escape_string($con, $_POST['comment']);
$res = mysqli_query($con, "SELECT user_id FROM comments WHERE id=$comment_id");
$row = mysqli_fetch_assoc($res);
if (!$row || $row['user_id'] != $_SESSION['id']) {
    http_response_code(403);
    echo "Forbidden";
    exit;
}
if (mysqli_query($con, "UPDATE comments SET comment='$new_comment' WHERE id=$comment_id")) {
    echo "success";
} else {
    http_response_code(500);
    echo "Error updating comment.";
}
```

- **Only comment owner can edit**
- **Secure update with AJAX**

6. Frontend AJAX for Login/Register

```
1. $(function() {
2.     $('#login-form').on('submit', function(e) {
3.         e.preventDefault();
4.         var formData = $(this).serialize();
5.         $.ajax({
6.             type: 'POST',
7.             url: 'php/ajax_login.php',
8.             data: formData,
9.             dataType: 'json',
10.             success: function(response) {
11.                 if (response.success) {
12.                     $('#login-message').html("<div class='message' id='success'><p>Login Successful</p></div>");
13.                     setTimeout(function() { location.reload(); }, 1500);
14.                 } else {
15.                     $('#login-message').html("<div class='message' id='error'><p>" + response.message +
"</p></div>");
16.                 }
17.             }
18.         });
19.     });
20. });
```

- **AJAX form submission**
- **Real-time feedback**
- **No page reload on error/success**

7. Database Table Creation (Security & Integrity)

```php
<?php
mysqli_query($con, "
CREATE TABLE IF NOT EXISTS users (
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(200),
    email VARCHAR(200),
    age INT(11),
    password VARCHAR(255)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
");

mysqli_query($con, "
CREATE TABLE IF NOT EXISTS comments (
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    user_id INT(11) NOT NULL,
    page_id VARCHAR(255) NOT NULL,
    comment TEXT NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
");
```

- **Table structure**
- **Foreign key for comment ownership**
- **Data integrity**

8. Session Security (Logout)

```php
<?php
session_start();
session_unset();
session_destroy();
header("Location: ../login.php");
exit;
```

- **Secure session termination**
- **Redirect to login**