Phase 3: Backend Development

What is Backend Development?

Backend = "Behind the Scenes" Logic

Think of your website like a restaurant:

- **Frontend (UI)** = The dining area customers see, the menu, the nice decorations
- **Backend** = The kitchen where food is prepared, recipes are followed, ingredients are stored

Objective of Phase 3:

To build the "brain" of your website - the PHP code that:

1. **Processes user actions** (when someone clicks "Register" or "Login")
2. **Talks to the database** (saves data, retrieves data)
3. **Makes decisions** (is the password correct? is this user an admin?)
4. **Handles security** (protect passwords, prevent hackers)

Roles & What You'll Actually Build:

1. Database Connection (`config/database.php`)

Purpose: Create a bridge between your PHP code and MySQL database

What it does:

```
Your PHP Code ↔ Database Class ↔ MySQL Database
```

Example: When a user logs in, this connection lets PHP check if the email/password exist in the `users` table.

2. User Authentication System

A. Registration Backend

What happens when someone fills the registration form:

1. **User fills form** → enters email, password, name
2. **PHP receives the data** (from `$_POST`)
3. **PHP validates:**
   - Is email format correct?
   - Is password strong enough?
   - Does email already exist in database?
4. **PHP hashes the password** (converts "password123" → "2y$10 abc...xyz" for security)

5. **PHP saves to database:**

```sql
INSERT INTO users (email, password, user_type)
VALUES ('john@example.com', 'hashed_password', 'student')
```

6. **PHP creates profile:**

```sql
INSERT INTO profiles (user_id, first_name, last_name)
VALUES (1, 'John', 'Doe')
```

7. **PHP redirects** → to profile setup page

Role: Accept new users safely into the system

B. Login Backend

What happens when someone tries to log in:

1. **User enters email + password**

2. **PHP checks database:**

```sql
SELECT * FROM users WHERE email = 'john@example.com'
```

3. **PHP verifies password:**

```php
password_verify('entered_password', 'stored_hashed_password')
```

4. **If correct:**
   - Create session: `$_SESSION['user_id'] = 1`
   - Remember user type: `$_SESSION['user_type'] = 'student'`
   - Redirect to dashboard

5. **If wrong:**
   - Show error: "Invalid credentials"

Role: Let registered users access their accounts securely

C. Session Management

Purpose: Remember who is logged in as they navigate pages

Functions you'll create:

```php
isLoggedIn()      // Check if someone is logged in
getUserType()     // Is this user a student, alumni, or admin?
requireLogin()    // Block access if not logged in
```

Example use:

- User visits `profile.php`
- PHP checks: `if (!isLoggedIn()) { redirect to login page }`
- This protects private pages!

Role: Keep users logged in and control access to pages

---

3. Core Features Backend

A. Profile Management

What it does:

- **View Profile:** Fetch user data from database and display it

```sql
SELECT * FROM profiles WHERE user_id = 1
```

- **Edit Profile:** Update database when user changes info

```sql
UPDATE profiles SET first_name='Jane', bio='New bio' WHERE user_id=1
```

- **Upload Picture:** Save uploaded image file, store filename in database
- **Manage Skills:** Add/remove skills from `skills` table

Role: Let users create and update their information

B. Matching System

What it does:

- **Find compatible mentors** for students based on:
  - Same major
  - Similar skills
  - Same location

Example algorithm:

```sql
SELECT * FROM profiles
JOIN users ON profiles.user_id = users.user_id
WHERE users.user_type = 'alumni'
AND profiles.major = 'Computer Science'   -- student's major
AND profiles.location = 'Cairo'           -- student's location
```

- **Handle requests:** Student clicks "Request Mentorship" → save to `mentorship_matches` table
- **Alumni responds:** Update match status to 'accepted' or 'rejected'

Role: Connect students with the right alumni mentors

C. Q&A Forum

What it does:

- **Create Post:**

```sql
INSERT INTO forum_posts (user_id, title, content, category)
VALUES (1, 'How to prepare for interviews?', 'I need advice...', 'Career')
```

- **Show All Posts:**

```sql
SELECT * FROM forum_posts ORDER BY created_at DESC
```

- **Add Reply:**

```sql
INSERT INTO forum_replies (post_id, user_id, content)
VALUES (5, 2, 'Here is my advice...')
```

- **Search Posts:** Filter by keyword or category

Role: Enable discussion and knowledge sharing

D. Job Board

What it does:

- **Alumni posts job:**

```sql
INSERT INTO jobs (posted_by, title, company, description)
VALUES (3, 'Junior Developer', 'Tech Corp', 'We are hiring...')
```

- **Students browse jobs:**

```sql
SELECT * FROM jobs WHERE status='active' ORDER BY posted_date DESC
```

- **Filter jobs:** By type (internship, full-time), location, etc.

Role: Share job opportunities

E. Events Management

What it does:

- **Create event:**

```sql
INSERT INTO events (created_by, title, event_date, max_attendees)
VALUES (4, 'Career Workshop', '2025-12-15', 50)
```

- **Register for event:**

```sql
INSERT INTO event_registrations (event_id, user_id)
VALUES (2, 1)
```

- **Check capacity:** Before registration, count current attendees

Role: Organize and manage campus events

F. Notification System

What it does:

- **Automatically create notifications** when:
  - Someone sends you a mentorship request
  - Someone replies to your forum post
  - A new job is posted in your field

Example:

````
sql

INSERT INTO notifications (user_id, message, type)
VALUES (1, 'You have a new mentorship request', 'match_request')
```


- **Mark as read:** Update `is_read = TRUE`
- **Count unread:** Show badge number

**Role:** Keep users informed of important activities

---

## Summary: Why Do Backend First?

| Without Backend | With Backend |
|----------------|--------------|
| Beautiful login form that does nothing | Login form that actually verifies users |
| Profile page with fake data | Profile page showing real user data from database |
| "Post Job" button that doesn't work | Button that saves job to database |
| Static page | Dynamic, functional website |

## The Flow:
```
Frontend (HTML Form)
    ↓
Backend (PHP Processing)
    ↓
Database (Store/Retrieve Data)
    ↓
Backend (Format Response)
    ↓
Frontend (Display Result)
````

Phase 3: Backend Development - Complete Step-by-Step Guide

Prerequisites Checklist

Before starting, ensure:

- ✅ XAMPP is installed and running
- ✅ Database `alumni_portal` is created
- ✅ All 10 tables are created successfully
- ✅ Test admin user is inserted
- ✅ Project folder structure exists

Foundation (Database Connection & Authentication)

Step 1: Create Database Connection Class

Location: `config/database.php`

Instructions:

1. Navigate to your project folder: `alumni-portal/config/`
2. Create a new file named `database.php`
3. Copy and paste this code:

```php
<?php
class Database {
    // Database credentials
    private $host = "localhost";
    private $db_name = "alumni_portal";
    private $username = "root";
    private $password = "";  // Empty for XAMPP default
    public $conn;
    // Get database connection
    public function getConnection() {
        $this->conn = null;
        try {
            // Create PDO connection
            $this->conn = new PDO(
                "mysql:host=" . $this->host . ";dbname=" . $this->db_name,
                $this->username,
                $this->password
            );
            // Set character set to utf8
            $this->conn->exec("set names utf8");
            // Set error mode to exceptions
            $this->conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

        } catch(PDOException $exception) {
            echo "Connection error: " . $exception->getMessage();
        }
        return $this->conn;
    }
}
?>
```

What this does:

- Creates a reusable class to connect to your MySQL database
- Uses PDO (PHP Data Objects) for secure database operations
- Sets up error handling

Step 2: Test Database Connection

Location: `test_connection.php` (in root folder)

Instructions:

1. Create `test_connection.php` in `alumni-portal/` folder
2. Add this code:

```
<?php
// Include database class
require_once 'config/database.php';
// Create database object
$database = new Database();
$conn = $database->getConnection();
// Check if connection is successful
if($conn) {
    echo "<h2>✅ Database Connection Successful!</h2>";
  // Test query - count tables
    $query = "SHOW TABLES";
    $stmt = $conn->prepare($query);
    $stmt->execute();
    echo "<h3>Tables in database:</h3>";
    echo "<ul>";
    while ($row = $stmt->fetch(PDO::FETCH_NUM)) {
        echo "<li>" . $row[0] . "</li>";
```

```
    }
    echo "</ul>";
    // Count users
    $query = "SELECT COUNT(*) as total FROM users";
    $stmt = $conn->prepare($query);
    $stmt->execute();
    $result = $stmt->fetch(PDO::FETCH_ASSOC);
    echo "<p><strong>Total users in database:</strong> " . $result['total'] .
"</p>";
} else {
    echo "<h2>❌ Database Connection Failed!</h2>";
}
?>
```

3. Open browser and visit: `http://localhost/alumni-portal/test_connection.php`
4. You should see all 10 tables listed and user count

Expected Output:

```
✅ Database Connection Successful!
Tables in database:
- events
- event_registrations
- forum_posts
- forum_replies
- jobs
- mentorship_matches
- notifications
- profiles
- skills
- users

Total users in database: 1
```

Step 3: Create Helper Functions File

Location: `includes/functions.php`

Instructions:

1. Navigate to `alumni-portal/includes/`
2. Create `functions.php`
3. Add this code:

```php
<?php
// Start session if not already started
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

// Check if user is logged in
function isLoggedIn() {
    return isset($_SESSION['user_id']);
}

// Get current user's ID
function getUserId() {
    return $_SESSION['user_id'] ?? null;
}

// Get current user's type (student, alumni, admin)
function getUserType() {
    return $_SESSION['user_type'] ?? null;
}

// Get current user's email
function getUserEmail() {
    return $_SESSION['email'] ?? null;
```

```
}

// Require login - redirect if not logged in
function requireLogin() {
    if (!isLoggedIn()) {
        $_SESSION['error'] = "Please login to access this page";
        header("Location: login.php");
        exit();
    }
}

// Require specific user type
function requireUserType($type) {
    requireLogin();
    if (getUserType() !== $type) {
        $_SESSION['error'] = "Access denied. This page is for " . $type . " only.";
        header("Location: dashboard.php");
        exit();
    }
}

// Require admin access
function requireAdmin() {
    requireUserType('admin');
}

// Sanitize input data
function sanitizeInput($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

```php
// Validate email format
function validateEmail($email) {
    return filter_var($email, FILTER_VALIDATE_EMAIL);
}

// Validate password strength
function validatePassword($password) {
    // At least 8 characters, 1 uppercase, 1 lowercase, 1 number
    if (strlen($password) < 8) {
        return "Password must be at least 8 characters long";
    }
    if (!preg_match("/[A-Z]/", $password)) {
        return "Password must contain at least one uppercase letter";
    }
    if (!preg_match("/[a-z]/", $password)) {
        return "Password must contain at least one lowercase letter";
    }
    if (!preg_match("/[0-9]/", $password)) {
        return "Password must contain at least one number";
    }
    return true;
}

// Generate CSRF token
function generateCSRFToken() {
    if (!isset($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['csrf_token'];
}

// Verify CSRF token
function verifyCSRFToken($token) {
    return isset($_SESSION['csrf_token']) &&
hash_equals($_SESSION['csrf_token'], $token);
```

```
}

// Set success message
function setSuccess($message) {
    $_SESSION['success'] = $message;
}

// Set error message
function setError($message) {
    $_SESSION['error'] = $message;
}

// Get and clear success message
function getSuccess() {
    if (isset($_SESSION['success'])) {
        $message = $_SESSION['success'];
        unset($_SESSION['success']);
        return $message;
    }
    return null;
}

// Get and clear error message
function getError() {
    if (isset($_SESSION['error'])) {
        $message = $_SESSION['error'];
        unset($_SESSION['error']);
        return $message;
    }
    return null;
}

// Redirect with message
function redirect($page, $message = null, $type = 'success') {
    if ($message) {
```

```
        if ($type === 'success') {
            setSuccess($message);
        } else {
            setError($message);
        }
    }
    header("Location: " . $page);
    exit();
}

// Format date
function formatDate($date) {
    return date('F j, Y', strtotime($date));
}

// Format datetime
function formatDateTime($datetime) {
    return date('F j, Y g:i A', strtotime($datetime));
}

// Time ago function (e.g., "2 hours ago")
function timeAgo($datetime) {
    $timestamp = strtotime($datetime);
    $difference = time() - $timestamp;

    if ($difference < 60) {
        return "just now";
    } elseif ($difference < 3600) {
        $mins = floor($difference / 60);
        return $mins . " minute" . ($mins > 1 ? "s" : "") . " ago";
    } elseif ($difference < 86400) {
        $hours = floor($difference / 3600);
        return $hours . " hour" . ($hours > 1 ? "s" : "") . " ago";
    } elseif ($difference < 604800) {
        $days = floor($difference / 86400);
```

```
        return $days . " day" . ($days > 1 ? "s" : "") . " ago";
    } else {
        return formatDate($datetime);
    }
}

// Upload file function
function uploadFile($file, $allowedTypes = ['jpg', 'jpeg', 'png', 'gif'], $maxSize =
5242880, $uploadDir = 'assets/uploads/') {
    // Check if file was uploaded
    if ($file['error'] !== UPLOAD_ERR_OK) {
        return ['success' => false, 'message' => 'File upload error'];
    }

    // Check file size (default 5MB)
    if ($file['size'] > $maxSize) {
        return ['success' => false, 'message' => 'File is too large. Maximum size is ' .
($maxSize / 1048576) . 'MB'];
    }

    // Get file extension
    $fileExtension = strtolower(pathinfo($file['name'], PATHINFO_EXTENSION));

    // Check file type
    if (!in_array($fileExtension, $allowedTypes)) {
        return ['success' => false, 'message' => 'Invalid file type. Allowed: ' .
implode(', ', $allowedTypes)];
    }

    // Generate unique filename
    $newFilename = uniqid() . '_' . time() . '.' . $fileExtension;
    $targetPath = $uploadDir . $newFilename;

    // Create upload directory if it doesn't exist
    if (!file_exists($uploadDir)) {
```

```
        mkdir($uploadDir, 0777, true);
    }

    // Move uploaded file
    if (move_uploaded_file($file['tmp_name'], $targetPath)) {
        return ['success' => true, 'filename' => $newFilename, 'path' =>
$targetPath];
    } else {
        return ['success' => false, 'message' => 'Failed to move uploaded file'];
    }
}
?>
```

What this does:

- Manages user sessions
- Provides security functions (CSRF protection, input sanitization)
- Validates email and password
- Handles file uploads
- Formats dates and times
- Manages success/error messages

Step 4: Create User Registration Backend

Location: `pages/register.php`

Instructions:

1. Navigate to `alumni-portal/pages/`
2. Create `register.php`
3. Add this complete code:

```php
<?php
// Include required files
require_once '../config/database.php';
require_once '../includes/functions.php';

// Initialize variables
$errors = [];
$success = '';

// Check if form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    // Get form data
    $email = sanitizeInput($_POST['email']);
    $password = $_POST['password'];
    $confirm_password = $_POST['confirm_password'];
    $user_type = sanitizeInput($_POST['user_type']);
    $first_name = sanitizeInput($_POST['first_name']);
    $last_name = sanitizeInput($_POST['last_name']);

    // Validate email
    if (empty($email)) {
        $errors[] = "Email is required";
    } elseif (!validateEmail($email)) {
        $errors[] = "Invalid email format";
    }

    // Validate password
    if (empty($password)) {
        $errors[] = "Password is required";
    } else {
        $passwordValidation = validatePassword($password);
        if ($passwordValidation !== true) {
            $errors[] = $passwordValidation;
        }
```

```
}

// Validate confirm password
if ($password !== $confirm_password) {
   $errors[] = "Passwords do not match";
}

// Validate user type
if (empty($user_type) || !in_array($user_type, ['student', 'alumni'])) {
   $errors[] = "Please select a valid user type";
}

// Validate names
if (empty($first_name)) {
   $errors[] = "First name is required";
}
if (empty($last_name)) {
   $errors[] = "Last name is required";
}

// If no errors, proceed with registration
if (empty($errors)) {
   try {
      // Create database connection
      $database = new Database();
      $conn = $database->getConnection();

      // Check if email already exists
      $query = "SELECT user_id FROM users WHERE email = :email";
      $stmt = $conn->prepare($query);
      $stmt->bindParam(':email', $email);
      $stmt->execute();

      if ($stmt->rowCount() > 0) {
         $errors[] = "Email already exists. Please use a different email or login.";
```

```
        } else {
            // Hash password
            $hashed_password = password_hash($password,
PASSWORD_DEFAULT);

            // Begin transaction
            $conn->beginTransaction();

            // Insert into users table
            $query = "INSERT INTO users (email, password, user_type, is_verified,
status)
                    VALUES (:email, :password, :user_type, FALSE, 'active')";
            $stmt = $conn->prepare($query);
            $stmt->bindParam(':email', $email);
            $stmt->bindParam(':password', $hashed_password);
            $stmt->bindParam(':user_type', $user_type);
            $stmt->execute();

            // Get the inserted user_id
            $user_id = $conn->lastInsertId();

            // Insert into profiles table
            $query = "INSERT INTO profiles (user_id, first_name, last_name)
                    VALUES (:user_id, :first_name, :last_name)";
            $stmt = $conn->prepare($query);
            $stmt->bindParam(':user_id', $user_id);
            $stmt->bindParam(':first_name', $first_name);
            $stmt->bindParam(':last_name', $last_name);
            $stmt->execute();

            // Commit transaction
            $conn->commit();

            // Set success message and redirect
            $_SESSION['success'] = "Registration successful! Please login.";
```

```
            header("Location: login.php");
            exit();
        }

    } catch(PDOException $e) {
        // Rollback transaction on error
        if ($conn->inTransaction()) {
            $conn->rollBack();
        }
        $errors[] = "Registration failed: " . $e->getMessage();
    }
  }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Register - Alumni Portal</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      min-height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
```

```
    padding: 20px;
}

.register-container {
    background: white;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);
    width: 100%;
    max-width: 500px;
}

.register-container h2 {
    text-align: center;
    color: #333;
    margin-bottom: 30px;
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    color: #555;
    font-weight: 500;
}

.form-group input,
.form-group select {
    width: 100%;
    padding: 12px;
    border: 1px solid #ddd;
    border-radius: 5px;
```

```
    font-size: 14px;
    transition: border-color 0.3s;
}

.form-group input:focus,
.form-group select:focus {
    outline: none;
    border-color: #667eea;
}

.error {
    background: #fee;
    color: #c33;
    padding: 10px;
    border-radius: 5px;
    margin-bottom: 20px;
    border-left: 4px solid #c33;
}

.error ul {
    margin-left: 20px;
}

.success {
    background: #efe;
    color: #3c3;
    padding: 10px;
    border-radius: 5px;
    margin-bottom: 20px;
    border-left: 4px solid #3c3;
}

.btn {
    width: 100%;
    padding: 12px;
```

```
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      color: white;
      border: none;
      border-radius: 5px;
      font-size: 16px;
      font-weight: 600;
      cursor: pointer;
      transition: transform 0.2s;
    }

    .btn:hover {
      transform: translateY(-2px);
    }

    .login-link {
      text-align: center;
      margin-top: 20px;
      color: #666;
    }

    .login-link a {
      color: #667eea;
      text-decoration: none;
      font-weight: 600;
    }

    .login-link a:hover {
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <div class="register-container">
    <h2>Create Account</h2>
```

```
    <?php if (!empty($errors)): ?>
      <div class="error">
        <strong>Please fix the following errors:</strong>
        <ul>
          <?php foreach ($errors as $error): ?>
            <li><?php echo $error; ?></li>
          <?php endforeach; ?>
        </ul>
      </div>
    <?php endif; ?>

    <?php if (!empty($success)): ?>
      <div class="success"><?php echo $success; ?></div>
    <?php endif; ?>

    <form method="POST" action="">
      <div class="form-group">
        <label for="user_type">I am a:</label>
        <select name="user_type" id="user_type" required>
          <option value="">Select...</option>
          <option value="student" <?php echo (isset($_POST['user_type']) &&
$_POST['user_type'] === 'student') ? 'selected' : ''; ?>>Student</option>
          <option value="alumni" <?php echo (isset($_POST['user_type']) &&
$_POST['user_type'] === 'alumni') ? 'selected' : ''; ?>>Alumni</option>
        </select>
      </div>

      <div class="form-group">
        <label for="first_name">First Name:</label>
        <input type="text" id="first_name" name="first_name"
            value="<?php echo isset($_POST['first_name']) ?
htmlspecialchars($_POST['first_name']) : ''; ?>"
            required>
      </div>
```

```
        <div class="form-group">
            <label for="last_name">Last Name:</label>
            <input type="text" id="last_name" name="last_name"
                value="<?php echo isset($_POST['last_name']) ?
htmlspecialchars($_POST['last_name']) : ''; ?>"
                required>
        </div>

        <div class="form-group">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email"
                value="<?php echo isset($_POST['email']) ?
htmlspecialchars($_POST['email']) : ''; ?>"
                required>
        </div>

        <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <small style="color: #666; font-size: 12px;">At least 8 characters, 1
uppercase, 1 lowercase, 1 number</small>
        </div>

        <div class="form-group">
            <label for="confirm_password">Confirm Password:</label>
            <input type="password" id="confirm_password"
name="confirm_password" required>
        </div>

        <button type="submit" class="btn">Register</button>
    </form>

    <div class="login-link">
        Already have an account? <a href="login.php">Login here</a>
    </div>
```

```
  </div>
</body>
</html>
```

What this does:

- Displays registration form
- Validates all input data
- Checks if email already exists
- Hashes password securely
- Creates user account and profile
- Uses database transactions for data integrity

Step 5: Test Registration

Instructions:

1. Open browser and go to: `http://localhost/alumni-portal/pages/register.php`
2. Fill in the form:
   - User Type: Student
   - First Name: John
   - Last Name: Doe
   - Email: john@student.com
   - Password: Test1234
   - Confirm Password: Test1234
3. Click "Register"
4. Check phpMyAdmin:
   - Open `users` table → Browse → You should see the new user
   - Open `profiles` table → Browse → You ↓ uld see the profile

Expected Result:

- User is created with hashed password
- Profile is created with first and last name
- You're redirected to login page with success message

Step 6: Create Login Backend

Location: `pages/login.php`

Instructions:

1. Create `login.php` in `alumni-portal/pages/`
2. Add this code:

```
<?php
// Include required files
require_once '../config/database.php';
require_once '../includes/functions.php';

// If already logged in, redirect to dashboard
if (isLoggedIn()) {
    header("Location: dashboard.php");
    exit();
}

// Initialize variables
$errors = [];

// Check if form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    // Get form data
    $email = sanitizeInput($_POST['email']);
    $password = $_POST['password'];
    $remember_me = isset($_POST['remember_me']);

    // Validate input
    if (empty($email)) {
        $errors[] = "Email is required";
```

```
}

if (empty($password)) {
   $errors[] = "Password is required";
}

// If no errors, proceed with login
if (empty($errors)) {
   try {
      // Create database connection
      $database = new Database();
      $conn = $database->getConnection();

      // Query to get user by email
      $query = "SELECT user_id, email, password, user_type, status
            FROM users
            WHERE email = :email";
      $stmt = $conn->prepare($query);
      $stmt->bindParam(':email', $email);
      $stmt->execute();

      // Check if user exists
      if ($stmt->rowCount() > 0) {
         $user = $stmt->fetch(PDO::FETCH_ASSOC);

         // Verify password
         if (password_verify($password, $user['password'])) {

            // Check if account is active
            if ($user['status'] !== 'active') {
               $errors[] = "Your account has been deactivated. Please contact admin.";
            } else {
               // Login successful - create session
               $_SESSION['user_id'] = $user['user_id'];
               $_SESSION['email'] = $user['email'];
               $_SESSION['user_type'] = $user['user_type'];

               // Handle "Remember Me"
               if ($remember_me) {
                  // Set cookie for 30 days
                  setcookie('user_email', $email, time() + (86400 * 30), "/");
```

```
                }

                // Redirect based on user type
                if ($user['user_type'] === 'admin') {
                    header("Location: ../admin/dashboard.php");
                } else {
                    header("Location: dashboard.php");
                }
                exit();
            }

        } else {
            $errors[] = "Invalid email or password";
        }
    } else {
        $errors[] = "Invalid email or password";
    }

  } catch(PDOException $e) {
    $errors[] = "Login failed: " . $e->getMessage();
  }
 }
}

// Get success message from session (if redirected from registration)
$success = getSuccess();
?>

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Login - Alumni Portal</title>
 <style>
  * {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
  }
```

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 20px;
}

.login-container {
    background: white;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);
    width: 100%;
    max-width: 450px;
}

.login-container h2 {
    text-align: center;
    color: #333;
    margin-bottom: 30px;
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    color: #555;
    font-weight: 500;
}

.form-group input {
    width: 100%;
    padding: 12px;
    border: 1px solid #ddd;
    border-radius: 5px;
```

```
    font-size: 14px;
    transition: border-color 0.3s;
}

.form-group input:focus {
    outline: none;
    border-color: #667eea;
}

.remember-me {
    display: flex;
    align-items: center;
    margin-bottom: 20px;
}

.remember-me input {
    width: auto;
    margin-right: 8px;
}

.remember-me label {
    margin: 0;
    font-weight: normal;
    color: #666;
}

.error {
    background: #fee;
    color: #c33;
    padding: 10px;
    border-radius: 5px;
    margin-bottom: 20px;
    border-left: 4px solid #c33;
}

.error ul {
    margin-left: 20px;
}

.success {
    background: #efe;
```

```
            color: #3c3;
            padding: 10px;
            border-radius: 5px;
            margin-bottom: 20px;
            border-left: 4px solid #3c3;
        }

        .btn {
            width: 100%;
            padding: 12px;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
            border: none;
            border-radius: 5px;
            font-size: 16px;
            font-weight: 600;
            cursor: pointer;
            transition: transform 0.2s;
        }

        .btn:hover {
            transform: translateY(-2px);
        }

        .register-link {
            text-align: center;
            margin-top: 20px;
            color: #666;
        }

        .register-link a {
            color: #667eea;
            text-decoration: none;
            font-weight: 600;
        }

        .register-link a:hover {
            text-decoration: underline;
        }
    </style>
</head>
```

```
<body>
  <div class="login-container">
    <h2>Welcome Back</h2>

    <?php if (!empty($errors)): ?>
      <div class="error">
        <ul>
          <?php foreach ($errors as $error): ?>
            <li><?php echo $error; ?></li>
          <?php endforeach; ?>
        </ul>
      </div>
    <?php endif; ?>

    <?php if ($success): ?>
      <div class="success"><?php echo $success; ?></div>
    <?php endif; ?>

    <form method="POST" action="">
      <div class="form-group">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email"
            value="<?php echo isset($_POST['email']) ? htmlspecialchars($_POST['email']) :
(isset($_COOKIE['user_email']) ? htmlspecialchars($_COOKIE['user_email']) : ''); ?>"
            required>
      </div>

      <div class="form-group">
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
      </div>

      <div class="remember-me">
        <input type="checkbox" id="remember_me" name="remember_me"
            <?php echo isset($_COOKIE['user_email']) ? 'checked' : ''; ?>>
        <label for="remember_me">Remember me</label>
      </div>

      <button type="submit" class="btn">Login</button>
    </form>
```

```
      <div class="register-link">
        Don't have an account? <a href="register.php">Register here</a>
      </div>
    </div>
</body>
</html>
```

What this does:

- Displays login form
- Validates credentials against database
- Verifies password using `password_verify()`
- Creates session variables
- Implements "Remember Me" functionality
- Redirects to appropriate dashboard

Step 7: Test Login

Instructions:

1. Go to: `http://localhost/alumni-portal/pages/login.php`
2. Enter the credentials you just registered:
   - Email: john@student.com
   - Password: Test1234
3. Click "Login"
4. You should be redirected (will show error for now because dashboard.php doesn't exist yet)

Test with Admin Account:

1. Login with:
   - Email: admin@alumni.com
   - Password: password
2. Should work!

Step 8: Create Basic Dashboard

Location: `pages/dashboard.php`

Instructions:

1. Create `dashboard.php` in `alumni-portal/pages/`
2. Add this code:

```
<?php
// Include required files
require_once '../config/database.php';
require_once '../includes/functions.php';

// Require login
requireLogin();

// Get user information
$user_id = getUserId();
$user_type = getUserType();
$user_email = getUserEmail();

// Create database connection
$database = new Database();
$conn = $database->getConnection();

// Get user profile information
$query = "SELECT p.*, u.email, u.registration_date
        FROM profiles p
        INNER JOIN users u ON p.user_id = u.user_id
        WHERE p.user_id = :user_id";
$stmt = $conn->prepare($query);
$stmt->bindParam(':user_id', $user_id);
$stmt->execute();
$profile = $stmt->fetch(PDO::FETCH_ASSOC);

// Get some statistics
// Count total users
$query = "SELECT COUNT(*) as total FROM users WHERE status = 'active'";
$stmt = $conn->prepare($query);
$stmt->execute();
$total_users = $stmt->fetch(PDO::FETCH_ASSOC)['total'];
```

```
// Count students
$query = "SELECT COUNT(*) as total FROM users WHERE user_type = 'student' AND status = 'active'";
$stmt = $conn->prepare($query);
$stmt->execute();
$total_students = $stmt->fetch(PDO::FETCH_ASSOC)['total'];

// Count alumni
$query = "SELECT COUNT(*) as total FROM users WHERE user_type = 'alumni' AND status = 'active'";
$stmt = $conn->prepare($query);
$stmt->execute();
$total_alumni = $stmt->fetch(PDO::FETCH_ASSOC)['total'];

// Count active mentorship matches
$query = "SELECT COUNT(*) as total FROM mentorship_matches WHERE status IN ('active', 'pending')";
$stmt = $conn->prepare($query);
$stmt->execute();
$total_matches = $stmt->fetch(PDO::FETCH_ASSOC)['total'];
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dashboard - Alumni Portal</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: #f5f7fa;
    }
```

```
.navbar {
    background: white;
    padding: 15px 50px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.navbar h1 {
    color: #667eea;
    font-size: 24px;
}

.navbar .user-info {
    display: flex;
    align-items: center;
    gap: 20px;
}

.navbar .user-info span {
    color: #666;
}

.navbar .user-info .badge {
    background: #667eea;
    color: white;
    padding: 4px 12px;
    border-radius: 20px;
    font-size: 12px;
    text-transform: uppercase;
}

.navbar a {
    color: #667eea;
    text-decoration: none;
    font-weight: 600;
}

.container {
    max-width: 1200px;
```

```
    margin: 40px auto;
    padding: 0 20px;
}

.welcome-section {
    background: white;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    margin-bottom: 30px;
}

.welcome-section h2 {
    color: #333;
    margin-bottom: 10px;
}

.welcome-section p {
    color: #666;
    font-size: 16px;
}

.stats-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

.stat-card {
    background: white;
    padding: 25px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    text-align: center;
}

.stat-card h3 {
    font-size: 36px;
    color: #667eea;
    margin-bottom: 10px;
```

```
}

.stat-card p {
    color: #666;
    font-size: 14px;
}

.quick-actions {
    background: white;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.quick-actions h3 {
    color: #333;
    margin-bottom: 20px;
}

.action-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 15px;
}

.action-btn {
    display: block;
    padding: 20px;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    text-align: center;
    text-decoration: none;
    border-radius: 8px;
    font-weight: 600;
    transition: transform 0.2s;
}

.action-btn:hover {
    transform: translateY(-3px);
}
```

```
      .success-message {
        background: #d4edda;
        color: #155724;
        padding: 15px;
        border-radius: 5px;
        margin-bottom: 20px;
        border-left: 4px solid #28a745;
      }
    </style>
</head>
<body>
    <nav class="navbar">
      <h1>🎓 Alumni Portal</h1>
      <div class="user-info">
        <span><?php echo htmlspecialchars($profile['first_name'] . ' ' . $profile['last_name']);
?></span>
        <span class="badge"><?php echo htmlspecialchars($user_type); ?></span>
        <a href="logout.php">Logout</a>
      </div>
    </nav>

    <div class="container">
      <?php
      $success = getSuccess();
      if ($success):
      ?>
        <div class="success-message"><?php echo $success; ?></div>
      <?php endif; ?>

      <div class="welcome-section">
        <h2>Welcome back, <?php echo htmlspecialchars($profile['first_name']); ?>! 👋</h2>
        <p>Member since <?php echo formatDate($profile['registration_date']); ?></p>
      </div>

      <div class="stats-grid">
        <div class="stat-card">
          <h3><?php echo $total_users; ?></h3>
          <p>Total Members</p>
        </div>
        <div class="stat-card">
```

```
            <h3><?php echo $total_students; ?></h3>
            <p>Active Students</p>
        </div>
        <div class="stat-card">
            <h3><?php echo $total_alumni; ?></h3>
            <p>Active Alumni</p>
        </div>
        <div class="stat-card">
            <h3><?php echo $total_matches; ?></h3>
            <p>Mentorship Connections</p>
        </div>
    </div>

    <div class="quick-actions">
        <h3>Quick Actions</h3>
        <div class="action-grid">
            <a href="profile.php" class="action-btn">My Profile</a>
            <a href="matching.php" class="action-btn">Find Mentors</a>
            <a href="forum.php" class="action-btn">Forum</a>
            <a href="jobs.php" class="action-btn">Job Board</a>
            <a href="events.php" class="action-btn">Events</a>
        </div>
    </div>
  </div>
</body>
</html>
```

What this does:

- Requires user to be logged in
- Displays user information
- Shows platform statistics
- Provides quick navigation links

Step 9: Create Logout

Location: `pages/logout.php`

Instructions:

1. Create `logout.php` in `alumni-portal/pages/`
2. Add this code:

```
<?php
// Include functions
require_once '../includes/functions.php';

// Destroy session
session_start();
session_unset();
session_destroy();

// Clear remember me cookie
if (isset($_COOKIE['user_email'])) {
    setcookie('user_email', '', time() - 3600, '/');
}

// Redirect to login
header("Location: login.php");
exit();
?>
```

Step 10: Test Complete Authentication Flow

Testing Checklist:

1. ✅ **Register New User**
   - Go to register.php
   - Create account with valid data
   - Check database for new user
2. ✅ **Login**
   - Go to login.php
   - Login with new credentials
   - Should redirect to dashboard
3. ✅ **View Dashboard**
   - Should see welcome message
   - Should see statistics
   - Should see your name in navbar

4. ✅ **Logout**
   - Click logout
   - Should redirect to login page
   - Try accessing dashboard.php directly → should redirect to login
5. ✅ **Session Protection**
   - Logout
   - Try to access: `http://localhost/alumni-portal/pages/dashboard.php`
   - Should redirect to login page

🎉 Week 3 Complete!

What you've accomplished:

- ✅ Database connection working
- ✅ Helper functions created
- ✅ User registration system
- ✅ User login system
- ✅ Session management
- ✅ Basic dashboard
- ✅ Logout functionality