### **Project Structure**

arduino

Copy code

login-project/

├── index.html

├── login.php

├── register.php

├── authenticate.php

├── welcome.php

├── css/

│ └── styles.css

├── js/

│ └── scripts.js

└── includes/

└── db.php

### **Project Description**

Your project is a simple user authentication system with image upload functionality, built using PHP and MongoDB. Here's a detailed breakdown of each component:

#### **1. index.html**

* **Purpose**: The main login page where users enter their credentials to log in.
* **Features**:
  + Contains a form for username and password input.
  + Includes client-side validation via JavaScript to ensure all fields are filled before submission.
  + Links to the registration page for new users.

#### **2. login.php**

* **Purpose**: Handles the registration of new users (not fully described in the provided snippets).

#### **3. register.php**

* **Purpose**: Displays the registration form for new users (not fully described in the provided snippets).

#### **4. authenticate.php**

* **Purpose**: Processes the login form submission.
* **Functionality**:
  + Validates the user's credentials against the data stored in the MongoDB database.
  + If the credentials are correct, it starts a session and redirects the user to the welcome page.
  + If the credentials are incorrect, it displays an error message and redirects back to the login page.

#### **5. welcome.php**

* **Purpose**: The landing page after a successful login.
* **Features**:
  + Displays a welcome message along with the logged-in username.
  + Provides an interface to upload a profile image.
  + Displays the profile image if it exists, or a default avatar if it does not.

#### **6. css/styles.css**

* **Purpose**: Contains the styling for the web pages to ensure a consistent and visually appealing design.

#### **7. js/scripts.js**

* **Purpose**: Contains JavaScript code to handle form validation and image upload functionality.
* **Features**:
  + Validates the login form to ensure no fields are left empty.
  + Handles the image upload process using Fetch API, sending the image to upload.php.

#### **8. includes/db.php**

* **Purpose**: Contains the database connection logic.
* **Functionality**:
  + Provides a function to connect to MongoDB and return the database instance.

### **MongoDB Structure**

* **Database**: login\_project
* **Collection**: users

**Document Example**:  
json  
Copy code  
{

"\_id": {

"$oid": "668bffb58213401fbff49a81"

},

"username": "ss",

"password": "$2y$10$b1RQ1IAD0Xh9dasIkFbS/eaBrC3q84tldr7ES.YjK47DvSwW387Pi",

"password\_default": "$2y$10$AXXTfLFJFSdLJeWnKSS61ejHwtuF.i5YOeH9V2BkfgEBwSmokRWKy",

"profileImage": {

"data": {

"$binary": {

"base64": "your\_base64\_encoded\_string",

"subType": "00"

}

},

"contentType": "image/jpeg"

}

}

### **Features**

1. **User Authentication**:
   * Users can log in with their credentials.
   * Passwords are hashed using password\_hash() and verified using password\_verify().
2. **Profile Image Upload**:
   * Users can upload a profile image after logging in.
   * The image is stored in MongoDB in Base64 format.
   * The uploaded image is displayed on the welcome page, with a default avatar shown if no image is uploaded.
3. **Session Management**:
   * PHP sessions are used to maintain the logged-in state of the user.
   * Session variables store the username for personalized user experience.

### **Enhancements and Considerations**

* **Security**:
  + Ensure that user inputs are sanitized to prevent SQL injection and other attacks.
  + Implement HTTPS to secure data transmission.
* **User Experience**:
  + Improve error handling to provide better feedback to users.
  + Add client-side validation to the registration form.
* **Scalability**:
  + Consider using a more scalable and secure method for handling image uploads and storage, such as cloud storage solutions.