



Republic of the Philippines

Region V Bicol

**BICOL UNIVERSITY**

Polangui, Albay



**Computer Studies Department**

---

## **Lab Submission for Week 12**

*(Phase 2 and Phase 3)*

# *ProFolio*

*Where you let your work shine and watch opportunities follow*

<b>Member</b>	<b>Role</b>
Ofracio, Aran Joshua P.	Project Manager
Dolon, Ella Mae R.	Frontend Developer
Lumayad, Victor Ronel P.	Backend Developer
Dela Cruz, Paulina C.	Database Manager
Odoño, Alona P.	GitHub Manager
Noble, Noelissa S.	Documentation Officer

**BSIT 2A**

**Dr. Guillermo V. Red, Jr.**

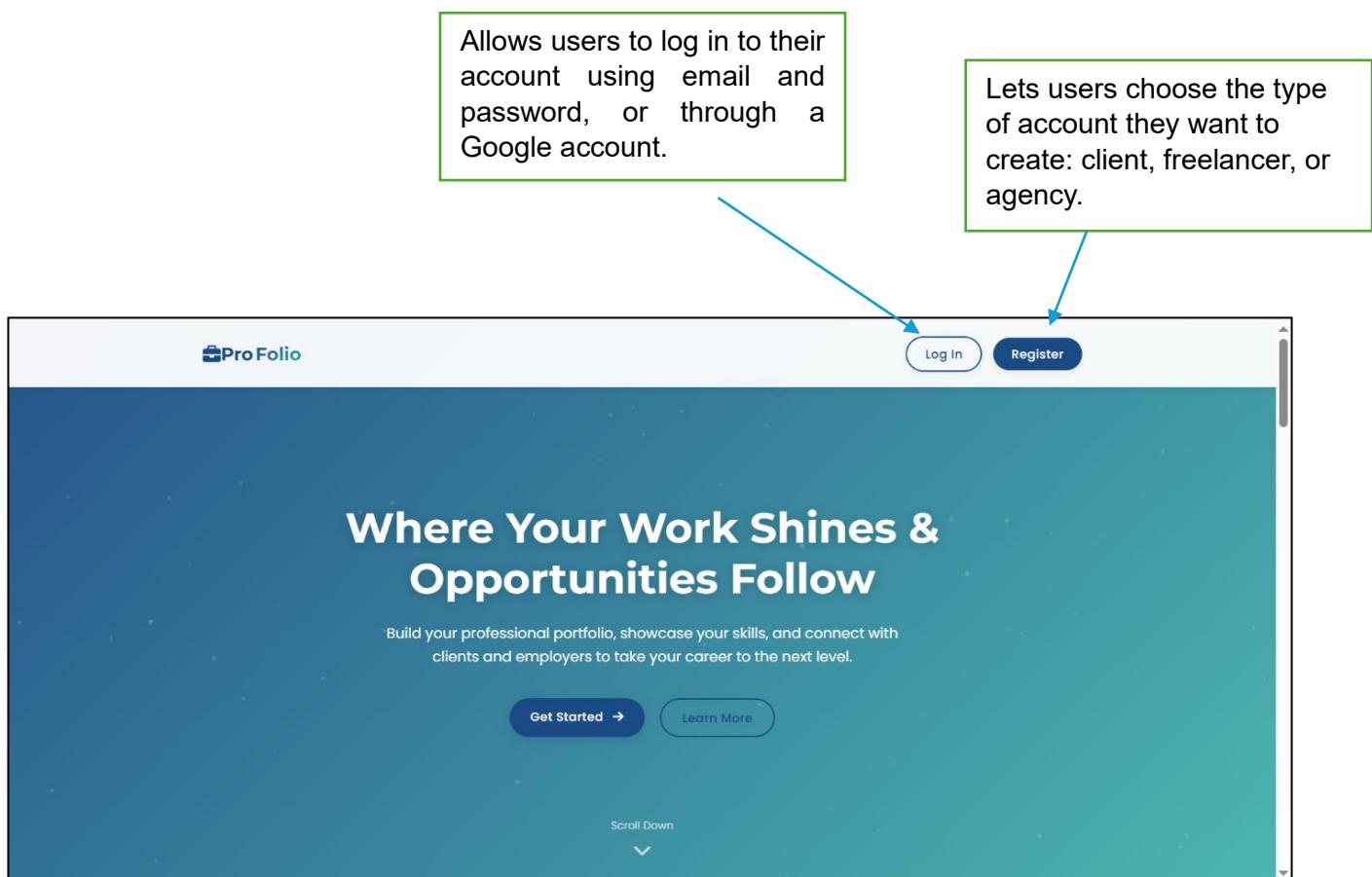
Course Professor

April 20, 2025

# Screenshots of ALL working frontend pages and Annotations showing which feature/use case each page covers

- **Home Page**

- **Feature:** Main Home Page
- **Use Case:** The home page is the first thing users see when visiting the platform. It introduces what ProFolio is about, showcases its features, explains how it works, and provides clear navigation for logging in or signing up based on user type. When they visit the platform, it introduces what ProFolio is about, highlights its core features, and guides users to get started. It includes sections like navigation, feature highlights, how the system works, and important footer links of the platform. From here, users can choose to register as a client, freelancer, or agency, or log in to their existing account. Each section is designed to introduce users to the platform's purpose, features, and how it works.



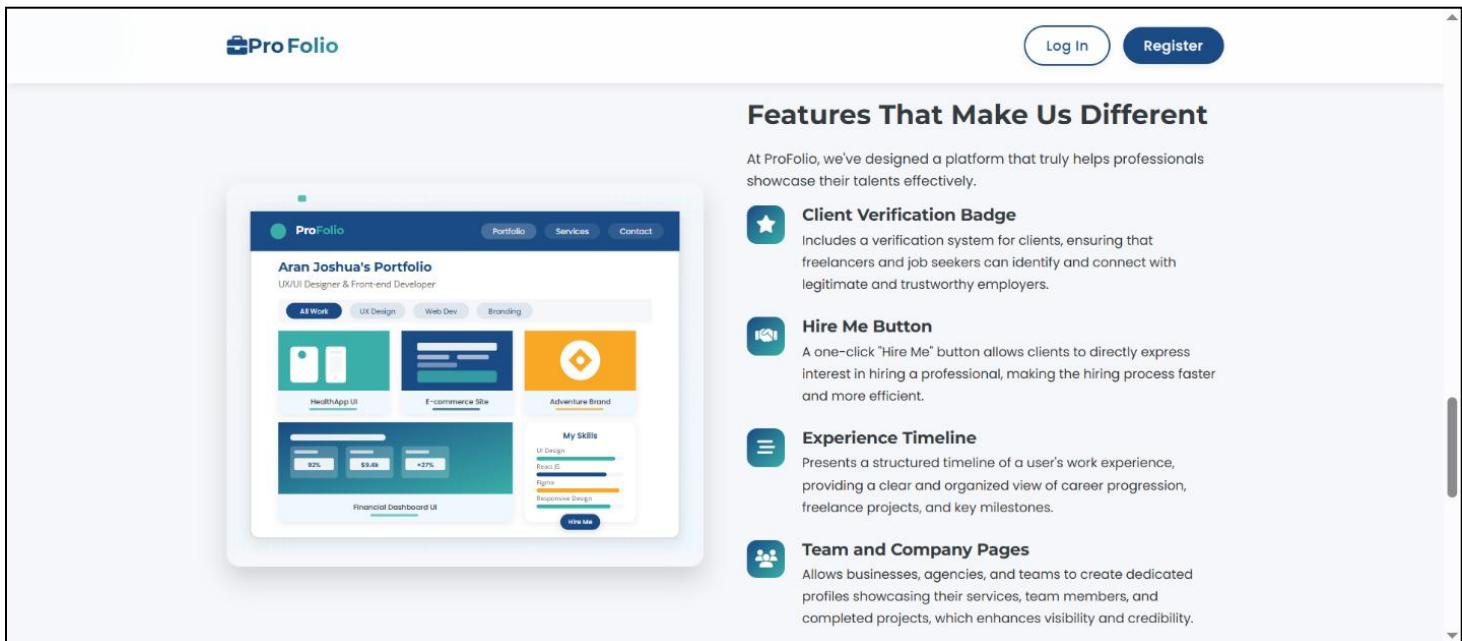
## Scrolling Down in Homepage

The screenshot shows the top portion of the ProFolio homepage. At the top left is the ProFolio logo. To the right are 'Log In' and 'Register' buttons. Below the header is a section titled 'Making a Difference' with the tagline 'Empowering professionals worldwide to showcase their talents'. Four large statistics are displayed: '150 Countries Reached', '2.5 Million Users', '500 K+ Projects Completed', and '98 % Satisfaction Rate'.

This screenshot shows the 'How ProFolio Works' section. It features three numbered steps: 1. Create Your Profile (Icon: person writing), 2. Get Discovered (Icon: magnifying glass), and 3. Connect & Collaborate (Icon: handshake). Below each step is a brief description. The background has a light teal gradient.

This screenshot shows the 'Who Benefits from ProFolio?' section. It highlights four user groups: Freelancers, Job Seekers, Businesses, and Clients & Recruiters, each with a corresponding icon and a brief description. The background features a large grey circular graphic on the right side.

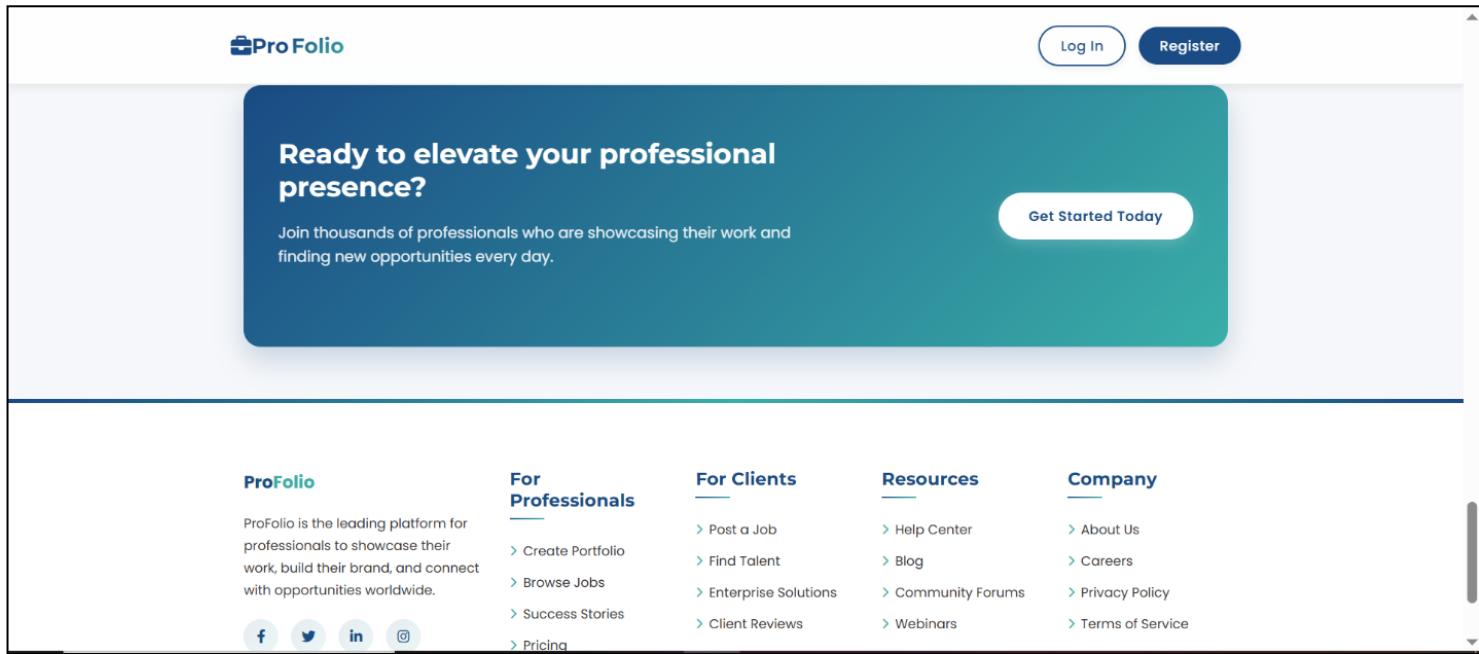
## Scrolling Down in Homepage



The screenshot shows the top portion of the ProFolio homepage. At the top left is the logo "ProFolio". To the right are "Log In" and "Register" buttons. Below the header, a large section titled "Features That Make Us Different" lists four key features with icons and descriptions:

- Client Verification Badge**: Includes a verification system for clients, ensuring that freelancers and job seekers can identify and connect with legitimate and trustworthy employers.
- Hire Me Button**: A one-click "Hire Me" button allows clients to directly express interest in hiring a professional, making the hiring process faster and more efficient.
- Experience Timeline**: Presents a structured timeline of a user's work experience, providing a clear and organized view of career progression, freelance projects, and key milestones.
- Team and Company Pages**: Allows businesses, agencies, and teams to create dedicated profiles showcasing their services, team members, and completed projects, which enhances visibility and credibility.

On the left side of the main content area, there is a preview of a user's portfolio named "Aran Joshua's Portfolio" for "UX/UI Designer & Front-end Developer". The portfolio includes sections for "All Work", "UX Design", "Web Dev", and "Branding", featuring thumbnail images of a HealthApp UI, an E-commerce Site, and an Adventure Brand, along with a progress bar for "My Skills" and a "View My Portfolio" button.



The screenshot shows the middle section of the ProFolio homepage. A large teal-colored call-to-action box contains the text "Ready to elevate your professional presence?". Below it, a smaller text says "Join thousands of professionals who are showcasing their work and finding new opportunities every day." To the right of the text is a "Get Started Today" button. At the top right of this section are "Log In" and "Register" buttons.

At the bottom of the page is a dark blue footer navigation bar with several sections:

- ProFolio**: Describes ProFolio as the leading platform for professionals to showcase their work, build their brand, and connect with opportunities worldwide. It includes social media links for Facebook, Twitter, LinkedIn, and Instagram.
- For Professionals**: Includes links for "Create Portfolio", "Browse Jobs", "Success Stories", and "Pricing".
- For Clients**: Includes links for "Post a Job", "Find Talent", "Enterprise Solutions", and "Client Reviews".
- Resources**: Includes links for "Help Center", "Blog", "Community Forums", and "Client Reviews".
- Company**: Includes links for "About Us", "Careers", "Privacy Policy", and "Terms of Service".

## ● Admin Login

- **Feature:** Login form for administrators
- **Use Case:** Allows admin users to securely log into the dashboard using their credentials. It includes input validation and backend integration to ensure only authorized users can gain access to the admin system.

The screenshot shows the 'Admin Control Panel' login interface for 'ProFolio'. At the top, a banner reads 'Authorized Personnel Only'. Below it are fields for 'Admin Email' (with a validation message 'Please fill out this field.') and 'Password'. There are 'Remember me' and 'Forgot password?' links, and a large blue 'Sign In' button. A note at the bottom states 'Secure administrator login - IP addresses are logged'. A green callout box on the right side of the screen contains the text: 'Utilizes admin email and password for logging in to ensure that only authorized personnel have access to the admin panel.' An arrow points from this text box towards the 'Admin Email' field.

## ● Admin - Dashboard

- **Feature:** Main interface for administrative control
- **Use Case:** Serves as an overview page of the platform, providing a high-level summary of platform activity, user engagement, and administrative tasks.

The screenshot shows the 'Admin Dashboard' for 'ProFolio'. On the left is a sidebar with 'Admin Panel' (System Administrator), 'Dashboard', 'User Management' (Verification Requests 24, Reported Users 2), 'Platform Policies', 'Analytics', 'Settings', and 'Logout'. The main area has a search bar 'Search users, reports, or policies...'. It displays 'Overview of platform statistics and activities' with four cards: '12,486 Total Users' (↑ 12% from last month), '24 Pending Approval' (↑ 8% from last week), '7 Active Reports' (↓ 3% from last week), and '867 Active Projects' (↑ 15% from last month). Below this is a 'Verification Queue' table with columns: USER, TYPE, SUBMITTED, PORTFOLIO, CREDENTIALS, CONTACT, STATUS, and ACTIONS. The table lists four entries: Sarah Johnson (Freelancer, Web Developer), Tech Solutions Inc. (Agency, Software Development), Michael Chen (Client, E-commerce), and Priya Sharma (Freelancer, UX/UI Designer). Each row shows document counts (e.g., 3 Documents, 5 Documents) and status buttons (Pending Review, Approve, Reject).

## • Admin – User Management

- **Feature:** User management interface
- **Use Case:** Enables the admin to view and manage a list of all registered users. Admins can filter users by type or account status to locate specific user groups. This makes it easier to organize users and monitor platform activity.

**User Type Filter** – lets the admin filter users by account type (All Users, Freelancer, Client, Agencies).

The screenshot shows the 'User Management' section of the ProFolio Admin Panel. On the left, there's a sidebar with 'User Management' selected. The main area has a search bar and several filters at the top: 'User Type' (set to 'All Users'), 'Status' (set to 'All Statuses'), 'Registration Date' (set to 'All Time'), and 'Sort By' (set to 'Newest First'). Below these are four summary boxes: 'Total Users' (12,486), 'Verified Users' (2,843), 'Suspended Users' (47), and 'New This Month' (326). At the bottom is a table titled 'Users' with columns for USER, TYPE, STATUS, EMAIL, JOINED, LAST ACTIVE, PROJECTS, and ACTIONS. A specific user, Sarah Johnson, is highlighted.

When 'All Users' is selected and the filter is applied, the system first displays the message: "Filters applied successfully!"

This screenshot shows the same 'User Management' page after applying the 'All Users' filter. A green success message 'Filters applied successfully!' is displayed in the top right. The status filter dropdown is open, showing 'All Statuses' as the selected option. The rest of the interface remains the same, including the user statistics and the table below.

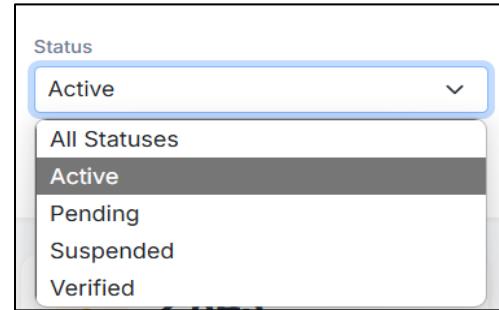
USER	TYPE	STATUS	EMAIL	JOINED	LAST ACTIVE	PROJECTS	ACTIONS
Sarah Johnson ID: #45892	Freelancer	Verified	sarah@example.com	Jan 12, 2025	Today	24	
Tech Solutions Inc. ID: #45973	Agency	Verified	info@techsolutions.com	Feb 25, 2025	Yesterday	38	
Michael Chen ID: #45821	Client	Pending Verification	michael@ecomstore.com	Mar 05, 2025	3 days ago	12	
Priya Sharma ID: #46012	Freelancer	Verified	priya@designstudio.com	Mar 18, 2025	Today	16	
David Rodriguez ID: #45734	Freelancer	Suspended	david@example.com	Jan 05, 2025	20 days ago	5	

0 selected

When 'All Statuses' is selected, the system will show users of all statuses (verified, pending verification, suspended, etc.)

This screenshot shows a zoomed-in view of the status filter dropdown. The 'All Statuses' option is highlighted in grey, indicating it is selected. Other options shown include Active, Pending, Suspended, and Verified.

When '**Active**' is selected, the list shows users who currently have access to their accounts and are allowed to use the platform's features.



<input type="checkbox"/>	USER	TYPE	STATUS	EMAIL	JOINED	LAST ACTIVE	PROJECTS	ACTIONS
<input type="checkbox"/>	Sarah Johnson ID: #45892	Freelancer	Verified	sarah@example.com	Jan 12, 2025	Today	24	
<input type="checkbox"/>	Tech Solutions Inc. ID: #45973	Agency	Verified	info@techsolutions.com	Feb 25, 2025	Yesterday	38	
<input type="checkbox"/>	Michael Chen ID: #45821	Client	Pending Verification	michael@ecomstore.com	Mar 05, 2025	3 days ago	12	
<input type="checkbox"/>	Priya Sharma ID: #46012	Freelancer	Verified	priya@designstudio.com	Mar 18, 2025	Today	16	

0 selected

When '**Pending**' is selected, only users whose accounts are still under review or awaiting approval will be shown, helping the admin easily identify users who require further action.

ProFolio

User Management

Search users by name, ID, or email...

April 19, 2025

Admin Panel System Administrator

Dashboard

User Management

Verification Requests 24

Reported Users 7

Platform Policies

Analytics

Settings

12,486 Total Users

2,843 Verified Users

47 Suspended Users

326 New This Month

12,486 Total Users

2,843 Verified Users

47 Suspended Users

326 New This Month

Users

<input type="checkbox"/>	USER	TYPE	STATUS	EMAIL	JOINED	LAST ACTIVE	PROJECTS	ACTIONS
<input type="checkbox"/>	Michael Chen ID: #45821	Client	Pending Verification	michael@ecomstore.com	Mar 05, 2025	3 days ago	12	

Logout

0 selected

ProFolio

User Management

Search users by name, ID, or email...

April 19, 2025

Admin Panel System Administrator

Dashboard

User Management

Verification Requests 24

Reported Users 7

Platform Policies

Analytics

Settings

12,486 Total Users

2,843 Verified Users

47 Suspended Users

326 New This Month

12,486 Total Users

2,843 Verified Users

47 Suspended Users

326 New This Month

Users

<input type="checkbox"/>	USER	TYPE	STATUS	EMAIL	JOINED	LAST ACTIVE	PROJECTS	ACTIONS
<input type="checkbox"/>	David Rodriguez ID: #45734	Freelancer	Suspended	david@example.com	Jan 05, 2025	20 days ago	5	

Logout

0 selected

When '**Suspended**' is selected, the system shows users who have been restricted from accessing the platform due to violations of terms or other administrative reasons

If **Freelancer** is selected and **Verified** is applied, the system will display only freelancers whose accounts have been verified.

The screenshot shows the ProFolio Admin Panel interface. On the left, there's a sidebar with navigation links like Dashboard, User Management (which is currently active), Verification Requests, Reported Users, Platform Policies, Analytics, and Settings. The main area has a search bar at the top. Below it, there are filters for User Type (set to 'Freelancers') and Status (set to 'Verified'). Key statistics are displayed: 12,486 Total Users, 2,843 Verified Users, 47 Suspended Users, and 326 New This Month. A large table below lists users with columns for USER, TYPE, STATUS, EMAIL, JOINED, LAST ACTIVE, PROJECTS, and ACTIONS. The first two users listed are Sarah Johnson and Priya Sharma, both marked as Verified. A blue arrow points from the 'User Management' link in the sidebar to the 'User Type' filter dropdown.

If **Client** is selected and **All Statuses** is applied, the system will display only clients, regardless of their account status. If the only status available for clients is **Pending**, the system will display only clients with a Pending status.

This screenshot is similar to the previous one but with different filter settings. The 'User Type' is still set to 'Freelancers', but the 'Status' dropdown is now set to 'Active'. The user list remains the same, showing Sarah Johnson and Priya Sharma as verified freelancers. A blue arrow points from the 'User Management' link in the sidebar to the 'Status' dropdown.

If **Agency** is selected and **All Statuses** is applied, the system will display only agencies, regardless of their account status. If the only status available for agencies is **Verified**, the system will display only agencies with a Verified status.

In this final screenshot, the 'User Type' is set to 'Freelancers' and the 'Status' is set to 'Suspended'. The user list now includes David Rodriguez, who is marked as suspended. A blue arrow points from the 'User Management' link in the sidebar to the 'Status' dropdown. A green notification bar at the top right says 'Filters applied successfully!'.

## • Verification Requests

- **Feature:** User verification management
- **Use Case:** This page lets the system administrators to review, approve, or reject verification requests submitted by users who want to be verified on the platform.

Advanced filters for sorting users based on status, user types and verification submission date.

USER	TYPE	SUBMITTED	PORTFOLIO	CREDENTIALS	CONTACT	STATUS	ACTIONS
Sarah Johnson ID: #45892	Freelancer Web Developer	April 16, 2025 2 hours ago	<a href="#">View</a>	3 Documents	sarah@example.com +1 (555) 123-4567	Pending Review	
Tech Solutions Inc. ID: #45973	Agency Software Development	April 16, 2025 5 hours ago	<a href="#">View</a>	5 Documents	info@techsolutions.com +1 (555) 987-6543	Pending Review	

USER	TYPE	SUBMITTED	PORTFOLIO	CREDENTIALS	CONTACT	STATUS	ACTIONS
Michael Chen ID: #45821	Client E-commerce	April 15, 2025 1 day ago	<a href="#">View</a>	1 Document	michael@ecomstore.com +1 (555) 234-5678	Additional Info Requested	
Priya Sharma ID: #46012	Freelancer UX/UI Designer	April 15, 2025 1 day ago	<a href="#">View</a>	4 Documents	priya@designstudio.com +1 (555) 345-6789	Pending Review	
Alex Rodriguez ID: #45981	Freelancer Mobile Developer	April 14, 2025 2 days ago	<a href="#">View</a>	2 Documents	alex@mobiledev.com +1 (555) 456-7890	Pending Review	
Global Marketing LLC ID: #45982	Agency Digital Marketing	April 14, 2025 2 days ago	<a href="#">View</a>	4 Documents	contact@globalmarketing.com	Additional Info Requested	

Portfolio Preview: Tech Solutions Inc.

**Tech Solutions Inc.**  
Software Development  
Verification Pending

**CONTACT INFORMATION**  
info@techsolutionsinc.com  
+1 (555) 987-6543  
techsolutionsinc.com  
San Francisco, CA

**VERIFICATION STATUS**  
Documents Provided: 3 files

**Reviews**

**John Smith**  
March 15, 2025  
Exceptional work quality and professional communication. Delivered the project ahead of schedule with all requirements met. Would definitely hire again.

**Emily Johnson**  
February 28, 2025  
Great experience working with this professional. Responsive, skilled, and delivered exactly what was needed for our project. Highly recommended.

**Approve Verification** **Reject**

An ‘Approve Verification’ button to verify users who meet the platform’s requirements and submitted complete, valid credentials.

A ‘Reject’ button to prevent unqualified, incomplete, or suspicious users from being verified.

**Reject Verification Request**

You are about to reject the verification request for User Name.

Rejection Reason:

Select a reason

Please provide detailed feedback for the user...

**Cancel** **Confirm Rejection**

Enables selection of a specific reason for rejecting the verification request, along with an input field for providing detailed feedback.

**Reject Verification Request**

You are about to reject the verification request for User Name.

Rejection Reason:

Select a reason

- Incomplete information provided
- Portfolio quality does not meet standards
- Insufficient credentials or experience
- Platform policy violation
- Suspicious or fraudulent activity
- Other (please specify)

## • Admin – Reported Users

- **Feature:** Reported user management with report details and action controls.
- **Use Case:** This page in the admin panel is used to monitor and manage user behavior on the platform. It allows admins to review reports submitted by other users, including reasons for the report. Based on these details, admins can take appropriate actions such as warning, suspending, or banning users who violate the platform's rules.

The screenshot shows the 'Reported Users' section of the ProFolio Admin Panel. At the top, there are four summary boxes: 'Active Reports' (7), 'High Priority' (3), 'Suspended Users' (12), and 'Resolved Reports' (89). Below this is a table titled 'Active Reports' with columns for ID, Reported User, Reason, Reporter, Date, Severity, Status, and Actions. The table lists five reports from users like CodeMaster486, WebStudio Pro, TechExpert, DigitalNomad, and FastLancer, each with a reporter like DavidMiller, CreativeStudio, RealTechCorp, StartupInc, or SystemAlert, and various status and severity indicators.

ID	REPORTED USER	REASON	REPORTER	DATE	SEVERITY	STATUS	ACTIONS
#R7201	CodeMaster486	Inappropriate Content Profile contains offensiv...	DavidMiller + 3 others	Apr 16, 2025 3 hours ago	High	Pending Review	
#R7199	WebStudio Pro	Payment Dispute Client payment issue	CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	Under Investigation	
#R7196	TechExpert	Profile Impersonation Claiming false company ...	RealTechCorp ID: #8781	Apr 15, 2025 1 day ago	Low	Pending Review	
#R7193	DigitalNomad	Fraudulent Activity Taking payment without ...	StartupInc + 2 others	Apr 15, 2025 1 day ago	High	Under Investigation	

This screenshot shows the 'Active Reports' table with a green callout box pointing to the 'High' filter button. The table displays three reports: #R7201 (Inappropriate Content, High severity), #R7193 (Fraudulent Activity, High severity), and #R7182 (Account Compromise, High severity). All three reports are currently under investigation.

Filters the table to show only high-severity reports or serious issues like fraudulent activity.

This screenshot shows the 'Active Reports' table with a green callout box pointing to the 'Medium' filter button. The table displays two reports: #R7199 (Payment Dispute, Medium severity) and #R7190 (Copyright Infringement, Medium severity). Both reports are pending review.

Displays only medium-severity reports, which are less urgent than high ones but still require attention, such as client payment issues or the use of copyrighted material.

This screenshot shows the 'Active Reports' table with a green callout box pointing to the 'Low' filter button. The table displays two reports: #R7196 (Profile Impersonation, Low severity) and #R7187 (Inappropriate Communication, Low severity). Both reports are pending review.

Shows only low-severity reports or minor issues that can be addressed later.

**Reported Users**

Review and manage user reports and violations.

April 19, 2025

ID	REPORTED USER	REASON	REPORTER	DATE	SEVERITY	STATUS	ACTIONS
#R7201	CodeMaster486 ID: #12458	Inappropriate Content Profile contains offensiv...	DavidMiller + 3 others	Apr 16, 2025 3 hours ago	High	Pending Review	
#R7199	WebStudio Pro ID: #10982	Payment Dispute Client payment issue	CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	Under Investigation	
#R7196	TechExpert ID: #9872	Profile Impersonation Claiming false company ...	RealTechCorp ID: #8761	Apr 15, 2025 1 day ago	Low	Pending Review	

Export Successful

Reports have been exported to CSV file.

The 'Export' button is used to download the report data, specifically the reported users list into a CSV file format. In this screenshot, a message "Export Successful" appears after clicking the button, and a file containing the reports is downloaded.

**Active Reports**

REPORTED USER	REASON	REPORTER	DATE	SEVERITY	STATUS	ACTIONS
CodeMaster486 ID: #12458	Inappropriate Content Profile contains offensiv...	DavidMiller + 3 others	Apr 16, 2025 3 hours ago	High	Pending Review	
WebStudio Pro ID: #10982	Payment Dispute Client payment issue	CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	Under Investigation	
TechExpert ID: #9872	Profile Impersonation Claiming false company ...	RealTechCorp ID: #8761	Apr 15, 2025 1 day ago	Low	Pending Review	
DigitalNomad ID: #11256	Fraudulent Activity Taking payment without ...	StartupInc + 2 others	Apr 15, 2025 1 day ago	High	Under Investigation	
MarketingGuru ID: #10345	Copyright Infringement Using copyrighted mate...	CreativeAgency ID: #9034	Apr 14, 2025 2 days ago	Medium	Pending Review	

When the suspend icon is clicked, a confirmation dialog pop-up appears, asking the admin to confirm the suspension of the reported user.

This page says

Are you sure you want to suspend user #12458? This action cannot be undone immediately.

OK Cancel

REPORTER	DATE	SEVERITY	STATUS	ACTIONS
DavidMiller + 3 others	Apr 16, 2025 3 hours ago	High	User Suspended	
CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	User Suspended	
RealTechCorp ID: #8761	Apr 15, 2025 1 day ago	Low	Pending Review	
StartupInc + 2 others	Apr 15, 2025 1 day ago	High	Under Investigation	
CreativeAgency ID: #9034	Apr 14, 2025 2 days ago	Medium	Pending Review	

User Suspended

User #10982 has been suspended successfully.

Active Reports						
REPORTED USER	REASON	REPORTER	DATE	SEVERITY	STATUS	ACTIONS
CodeMaster486 ID: #12458	Inappropriate Content Profile contains offensiv...	DavidMiller + 3 others	Apr 16, 2025 3 hours ago	High	Pending Review	
WebStudio Pro ID: #10982	Payment Dispute Client payment issue	CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	Under Investigation	
TechExpert ID: #9872	Profile Impersonation Claiming false company ...	RealTechCorp ID: #8761	Apr 15, 2025 1 day ago	Low	Pending Review	
DigitalNomad ID: #11256	Fraudulent Activity Taking payment without ...	StartupInc + 2 others	Apr 15, 2025 1 day ago	High	Under Investigation	
MarketingGuru ID: #10345	Copyright Infringement Using copyrighted mate...	CreativeAgency ID: #9034	Apr 14, 2025 2 days ago	Medium	Pending Review	

This pop-up allows an admin to resolve a user complaint.

It includes a dropdown of multiple resolution types, giving more flexibility on how reports are handled.

Resolve Report #R7201

Resolution Type: Warning Issued

Resolution Notes:

Cancel Confirm Resolution

REPORTED USER	REASON	REPORTER	DATE	SEVERITY	STATUS	ACTIONS
CodeMaster486 ID: #12458	Inappropriate Content Profile contains offensiv...	DavidMiller + 3 others	Apr 16, 2025 3 hours ago	High	Pending Review	
WebStudio Pro ID: #10982	Payment Dispute Client payment issue	CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	Under Investigation	
TechExpert ID: #9872	Profile Impersonation Claiming false company ...	RealTechCorp ID: #8761	Apr 15, 2025 1 day ago	Low	Pending Review	
DigitalNomad ID: #11256	Fraudulent Activity Taking payment without ...	StartupInc + 2 others	Apr 15, 2025 1 day ago	High	Under Investigation	
MarketingGuru ID: #10345	Copyright Infringement Using copyrighted mate...	CreativeAgency ID: #9034	Apr 14, 2025 2 days ago	Medium	Pending Review	

Once the admin confirms a resolution, the system displays a confirmation message at the bottom.

Resolve Report #R7199

Resolution Type: Content Removed

Cancel Confirm Resolution

REPORTED USER	REASON	REPORTER	DATE	SEVERITY	STATUS	ACTIONS
WebStudio Pro ID: #10982	Payment Dispute Client payment issue	CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	Under Investigation	
TechExpert ID: #9872	Profile Impersonation Claiming false company ...	RealTechCorp ID: #8761	Apr 15, 2025 1 day ago	Low	Pending Review	
DigitalNomad ID: #11256	Fraudulent Activity Taking payment without ...	StartupInc + 2 others	Apr 15, 2025 1 day ago	High	Under Investigation	

Active Reports						
REPORTED USER	REASON	REPORTER	DATE	SEVERITY	STATUS	ACTIONS
WebStudio Pro ID: #10982	Payment Dispute Client payment issue	CreativeStudio ID: #11523	Apr 16, 2025 6 hours ago	Medium	Under Investigation	
TechExpert ID: #9872	Profile Impersonation Claiming false company ...	RealTechCorp ID: #8761	Apr 15, 2025 1 day ago	Low	Pending Review	
DigitalNomad ID: #11256	Fraudulent Activity Taking payment without ...	StartupInc + 2 others	Apr 15, 2025 1 day ago	High	Under Investigation	
MarketingGuru ID: #10345	Copyright Infringement Using copyrighted mate...	CreativeAgency ID: #9034	Apr 14, 2025 2 days ago	Medium	Pending Review	
DesignWizard ID: #10782	Inappropriate Communic... Unprofessional messagi...	ClientFirst ID: #12087	Apr 14, 2025 2 days ago	Medium	Pending Review	

Report Resolved  
Report #R7201 has been marked as resolved.

## ● Admin – Platform Policies

- **Feature:** Platform Policy Management Panel
- **Use Case:** Allows administrators to add, edit, or publish policies such as Terms of Service, Privacy Protection, or Community Guidelines. This section ensures that the platform maintains transparency and clearly communicates rules to its users.

Allows the admin to create a new policy entry with a custom title, short description, and full content.

Lets the admin choose whether the new policy is intended for **public display** (visible to users on the website) or **internal only** (visible to admin staff for internal reference).

After the admin creates and saves a new policy '**Privacy Protection**' it will automatically appear in this section.

Allows the admin to create a new policy entry with a custom title, short description, and full content.

**Platform Policies**  
Manage and update platform policies and guidelines

**Active Policies**

- Terms of Service**  
Legal agreement between ProFolio and its users defining the rules, rights, and restrictions for using the platform.  
Last updated: April 12, 2025 Updated by: Admin Mark Public
- Privacy Policy**  
Outlines how ProFolio collects, uses, stores, and protects user data, including data retention and user rights.  
Last updated: April 5, 2025 Updated by: Admin Jane Public
- Community Guidelines**  
Standards for acceptable behavior on ProFolio, including prohibited content and anti-harassment policies.  
Last updated: March 28, 2025 Updated by: You Public
- Verification Guidelines**  
Requirements and processes for user verification, including documentation and review standards.  
Last updated: March 15, 2025 Updated by: Admin John Public
- Payment Terms**  
Rules governing payments, fee structures, refunds, and dispute resolution for financial transactions.  
Last updated: February 28, 2025 Updated by: Admin Jane Public

**+ New Policy**

**Add New Policy**

**Policy Title**

**Description**  
This policy aims to protect user privacy and outline how their data is collected and used.

**Content**  
ProFolio is committed to safeguarding your personal data. We collect only essential information and use it solely to improve your experience on our platform. All data is stored securely, and we never share it with third parties without consent. By using ProFolio, you agree to the terms outlined in this policy.

**Visibility**

**Cancel** **Add Policy**

Lets the admin choose whether the new policy is intended for **public display** (visible to users on the website) or **internal only** (visible to admin staff for internal reference).

**Platform Policies**  
Manage and update platform policies and guidelines

**Active Policies**

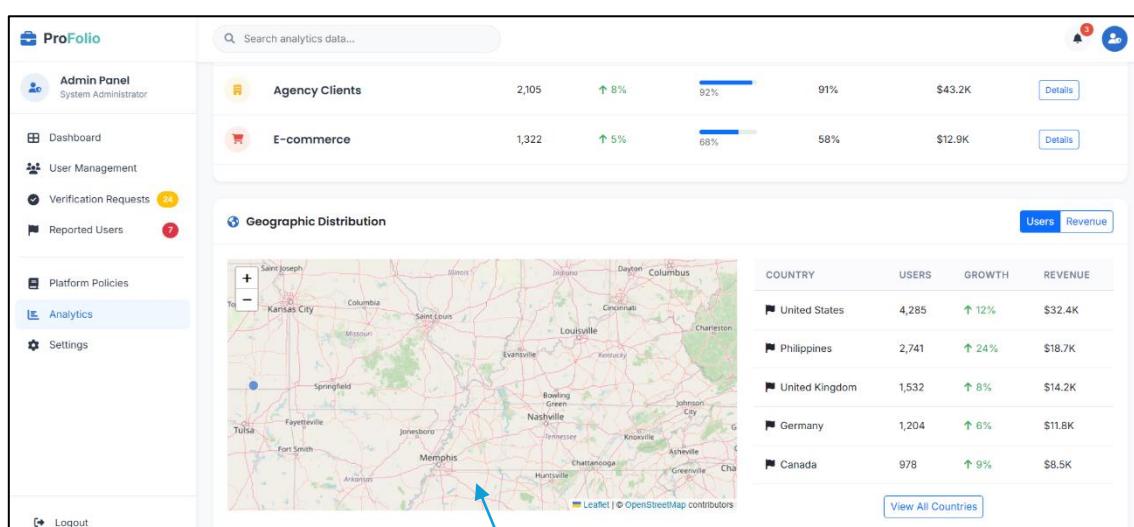
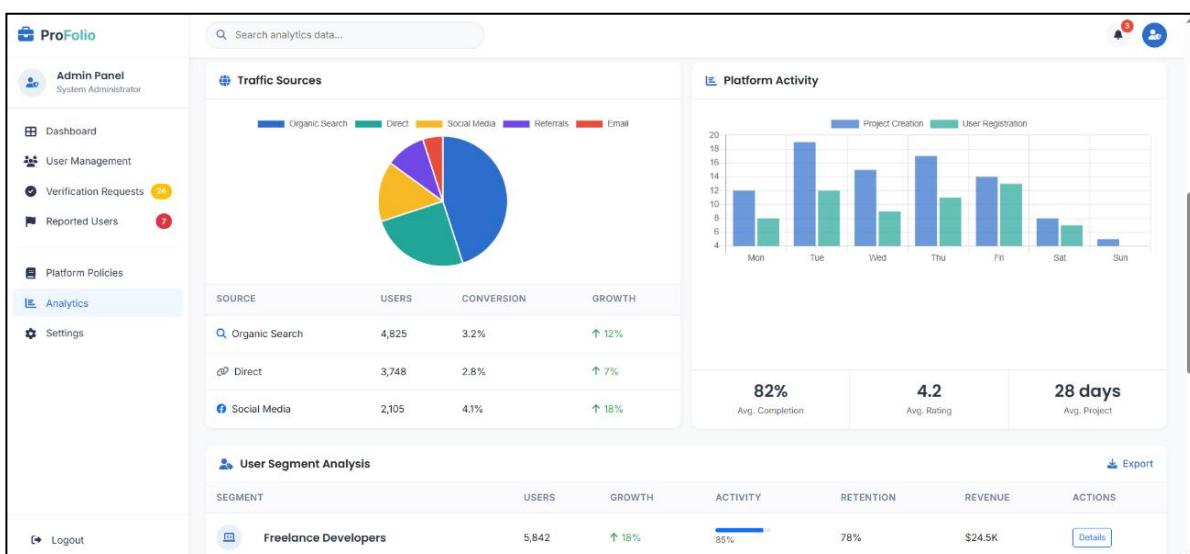
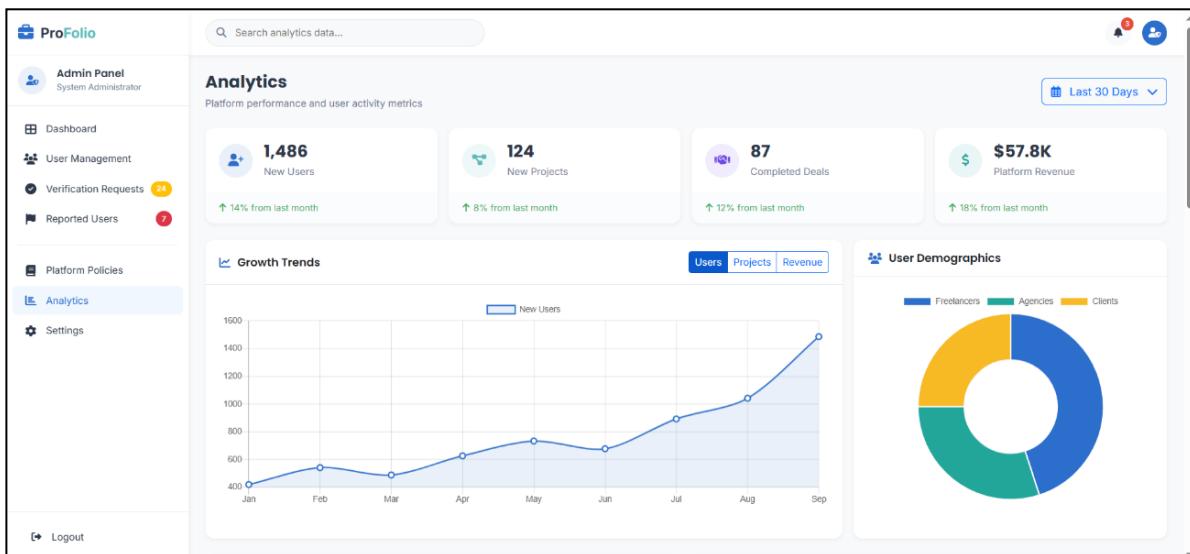
- Terms of Service**  
Legal agreement between ProFolio and its users defining the rules, rights, and restrictions for using the platform.  
Last updated: April 12, 2025 Updated by: Admin Mark Public
- Privacy Protection**  
This policy aims to protect user privacy and outline how their data is collected and used.  
Created: April 20, 2025 Updated by: You Public
- Community Guidelines**  
Standards for acceptable behavior on ProFolio, including prohibited content and anti-harassment policies.  
Last updated: March 28, 2025 Updated by: You Public
- Verification Guidelines**  
Requirements and processes for user verification, including documentation and review standards.  
Last updated: March 15, 2025 Updated by: Admin John Public
- Payment Terms**  
Rules governing payments, fee structures, refunds, and dispute resolution for financial transactions.  
Last updated: February 28, 2025 Updated by: Admin Jane Public
- Intellectual Property Rights**  
Guidelines on ownership, licensing, and usage rights for content uploaded to the platform.  
Last updated: February 10, 2025 Updated by: Admin Mark Public
- Internal Moderation Guidelines**  
Standards and protocols for moderators when reviewing reports and enforcing platform policies.  
Last updated: January 15, 2025 Updated by: You Internal Only

**Logout**

After the admin creates and saves a new policy '**Privacy Protection**' it will automatically appear in this section.

## ● Analytics

- **Feature:** Tracking and visualization of platform data
- **Use Case:** This page gives administrators insight to platform usage and engagement, showing key stats like new users, projects, deals, and revenue, along with monthly growth. It helps track progress, spot trends, and make informed decisions. It also has user demographics chart, displaying the types of users on the platform for more targeted strategies.



A geographic distribution of users on the platform created using Leaflet. This helps the admin understand which countries are most active or profitable, which can guide decisions related to marketing and support.

## • Settings

➤ **Feature:** Website administration and configuration

➤ **Use Case:** Enables website administrators to manage and configure various aspects of the site's functionality, security, and overall system settings. Through this page, admins can customize platform information and site appearance, set security, configure SMTP or email settings, and manage backups.

This screenshot shows the 'General' tab of the Admin Settings page. It includes sections for Platform Information (Platform Name: ProFolio, Platform Tagline: Connect, Create, Collaborate), Regional Settings (Default Timezone: UTC), Theme & Appearance (Primary Color: #2c6ecb, Secondary Color: #5cbbba), and Contact Information (Support Email: support@profolio.com, Support Phone: +1 (555) 123-4567). A 'Save All Changes' button is at the top right.

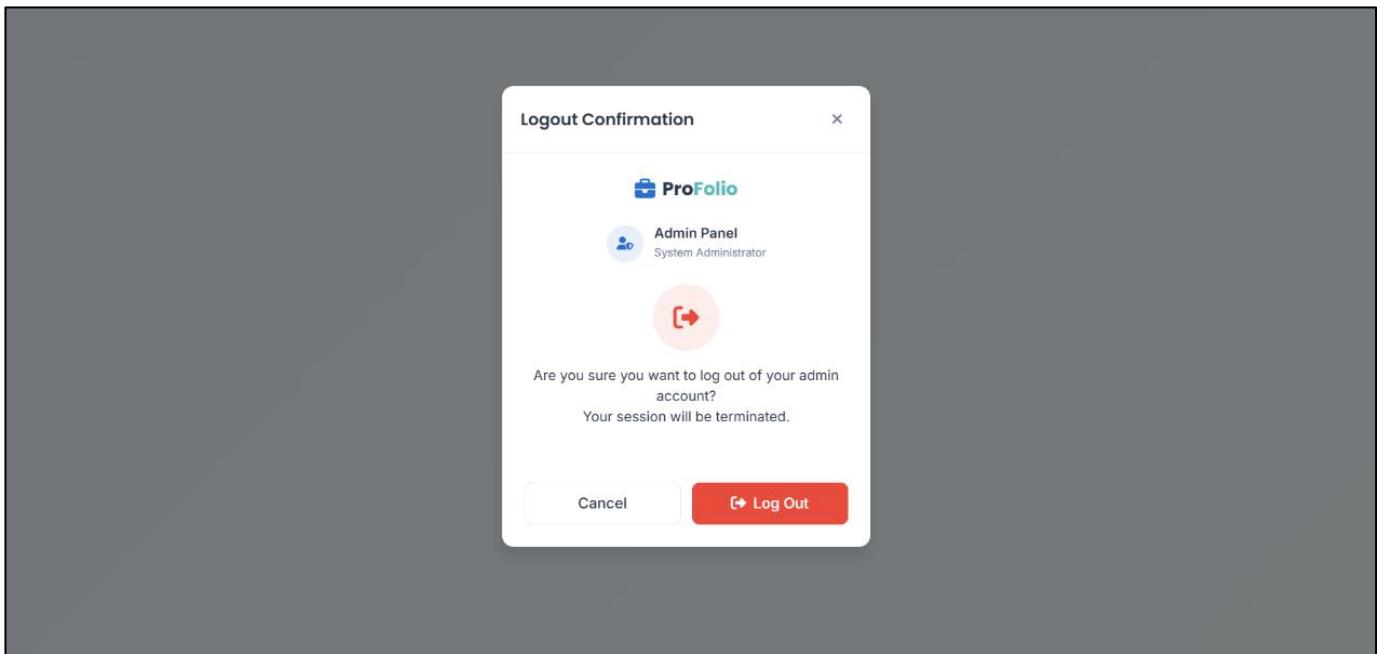
This screenshot shows the 'Security' tab of the Admin Settings page. It includes sections for Authentication (Require Two-Factor Authentication for Admins, Require Email Verification for New Users), Password Policy (Minimum Password Length: 10, requiring Uppercase Letters, Lowercase Letters, Numbers, and Special Characters), and Session Management (Session Timeout: 60 minutes, Force Logout on Password Change). A 'Save All Changes' button is at the top right.

This screenshot shows the 'Email' tab of the Admin Settings page. It includes sections for SMTP Configuration (SMTP Host: smtp.profolio.com, Port: 587, Encryption: TLS, Username: no-reply@profolio.com, Password: \*\*\*\*\*), and Email Preferences (Send Admins Daily Report, Send System Alert Notifications, Daily Email Rate Limit: 5000). On the right, there are sections for Notification Templates (New User Registration, Password Reset, Verification Approved, Verification Rejected, Account Warning) and a list of checkboxes for various notification types. A 'Save All Changes' button is at the top right.

This screenshot shows the 'Backup' tab of the Admin Settings page. It includes sections for Automated Backups (Enable Automated Backups, Backup Frequency: Weekly, Backup Time: 02:00 AM, Time is in UTC, Retention Period: 90 days) and Backup Storage (Storage Location: Amazon S3, Bucket Name: profolio-backups, Access Key: \*\*\*\*\*, Secret Key: \*\*\*\*\*). A 'Test Connection' button is at the bottom right. A 'Create Manual Backup' button is also present.

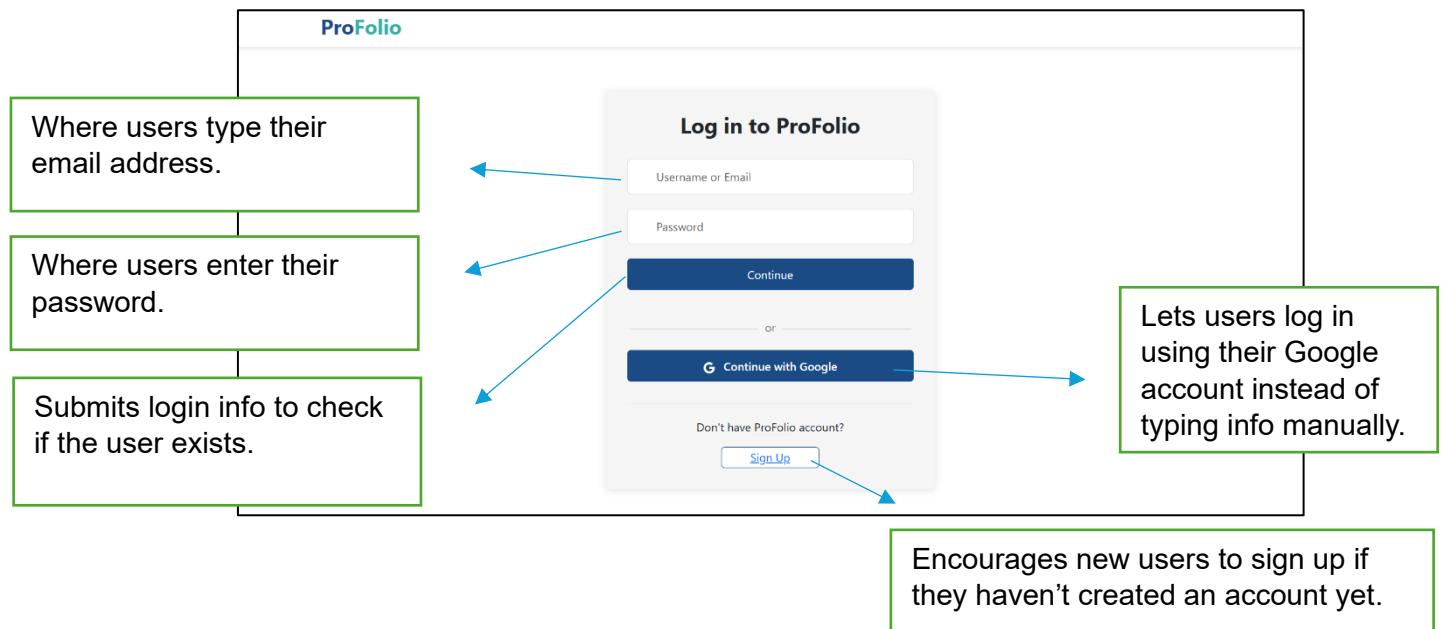
- **Admin Logout**

- **Feature:** Logout confirmation or button
- **Use Case:** This functionality allows administrators to securely end their session. Upon logging out, the system clears or invalidates the session variables used for authentication, making sure that no other user can access admin features without logging in again. This supports secure session handling and reinforces system protection from unauthorized access.



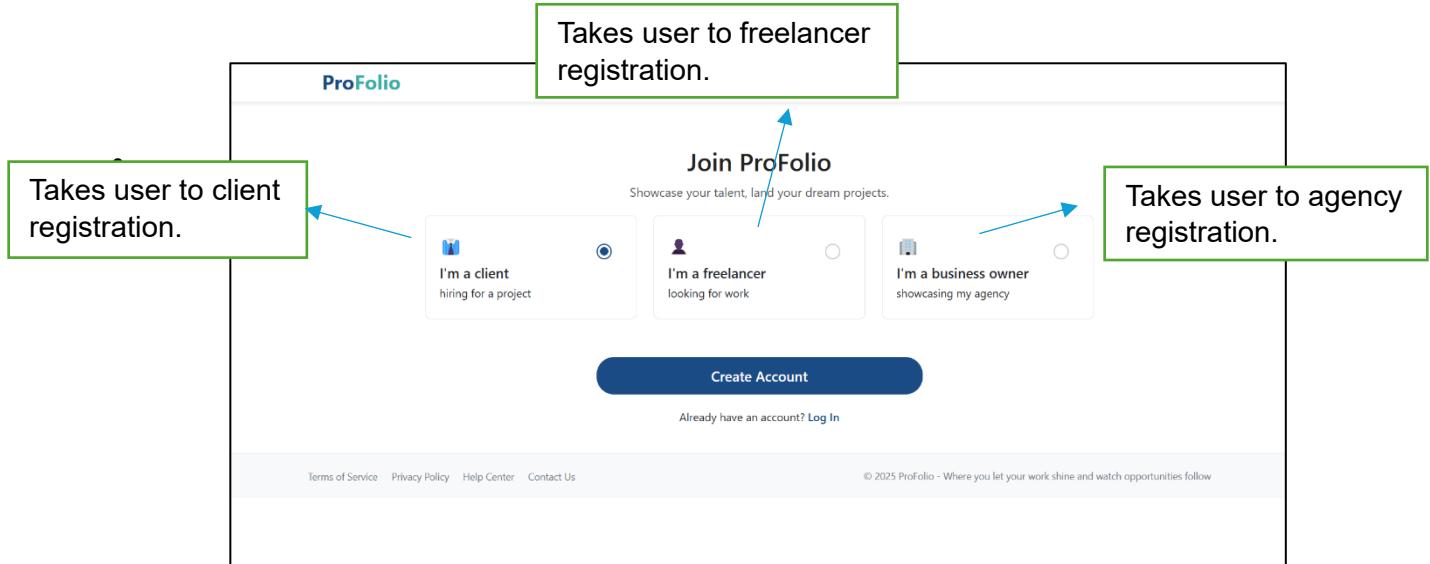
## • User - Login

- **Feature:** General User Login Page
- **Use Case:** Allows users (client, freelancer, agency) to log in to their account using email and password. This form also includes a redirection link to registration pages.



## • Register

- **Feature:** User Registration Selection Page
- **Use Case:** Lets users choose the type of account they want to create — client, freelancer, or agency. This acts as a gateway to the proper registration form based on user role.

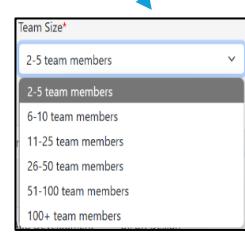
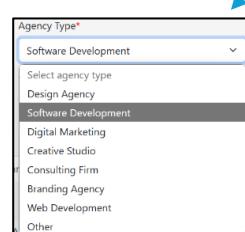


## • Register – as an Agency

- **Feature:** Agency Registration Form
- **Use Case:** Allows agencies to register by entering detailed business information, uploading a logo, and defining their services. This form collects data to differentiate agency accounts from freelancers or clients

- **Agency Name Field** – official name of the agency
- **Logo Upload** – file input for agency logo
- **Agency Type Dropdown** – used to select the type/category of agency
- **Team Size Field** – lets agencies specify how many people are in the team
- **Agency Description** – a short paragraph describing the agency's background and focus

The screenshot shows the 'Register your Team' page. It includes fields for 'Agency Name\*', 'Agency Logo' (with a file upload button), 'Agency Type\*' (dropdown menu), and 'Team Size\*' (dropdown menu). Below these is a text area for 'Agency Description\*'. Two blue arrows point from the 'Agency Type' and 'Team Size' dropdowns to enlarged versions of those specific dropdown menus.



- **Services Offered** – allows the user to choose multiple service offerings (e.g., web design, Ui/UX design, graphic design)
- **Agency Website Field** – optional link to the official site
- **First Name and Last Name** – personal details of the account owner
- **Job Title Field** – specifies their role in the agency (e.g., CEO, HR Manager)
- **Work Email Field** – required email address for login and contact

The screenshot shows the 'Services Offered' section. It features a grid of service categories: Web Design, Web Development, Mobile App Development, UI/UX Design, Branding, SEO, Content Marketing, Social Media, Graphic Design, Animation, E-commerce, and Custom. Below this is the 'Agency Website (Optional)' field with a placeholder URL.

- **Password Field** – required for account security
- **Country and City Fields** – capture agency's location for directory purposes
- **Submit Button** – completes the registration and sends all data to the backend

The screenshot shows the final registration step. It includes fields for 'Password\*' and 'Country\*' (Philippines selected). A 'City\*' field is also present. Below these are two checkboxes: one for receiving emails about finding talent and growing the agency, and another for accepting the 'Terms of Service' and 'User Agreement' and 'Privacy Policy'. At the bottom is a large blue 'Create my account' button. A note at the bottom right says 'Already have an account? Log In'.

Gives users the option to receive email tips or project suggestion.

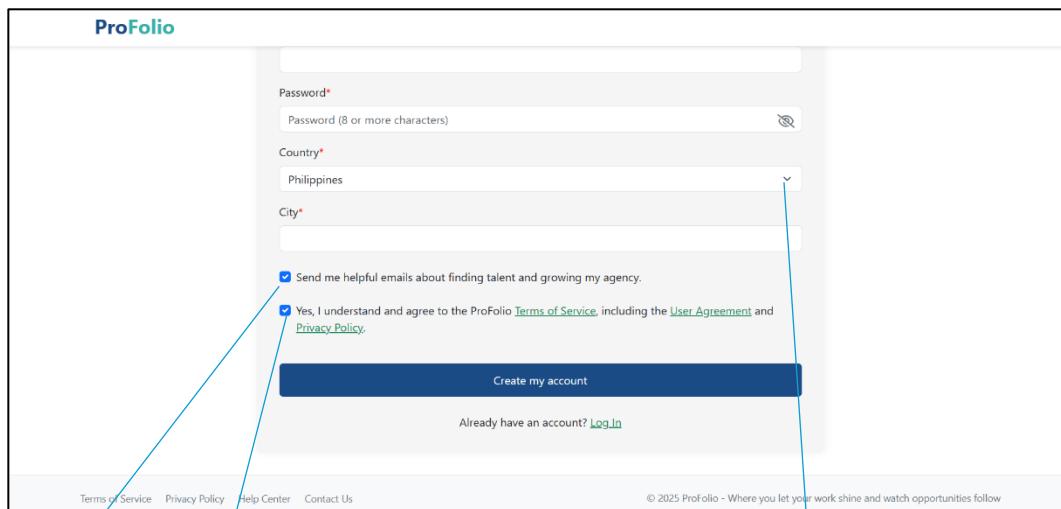
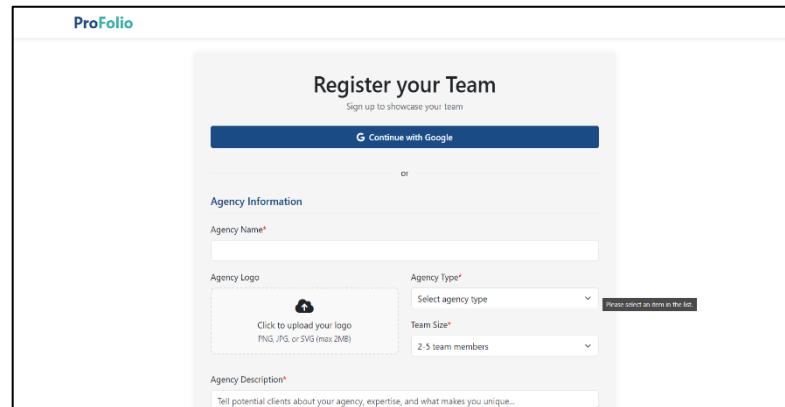
Users must agree before creating their account.



## • Register – as a Client

- **Feature:** Client Registration Form
- **Use Case:** This page allows clients to register an account by providing their personal and location information. It includes fields like first name, last name, email, password, and country. The form ensures that each user provides valid and complete details before they can use the platform.

- **First Name/Last Name Fields** – collect the user's full name
- **Email Input Field** – user provides their email for identification
- **Password Field** – stores a hashed version of the password in the database
- **Country Field** – captures the user's location for analytical purposes
- **Submit/Register Button** – sends data to backend for account creation.



ProFolio

Password\*  
Country\*  
City\*  
 Send me helpful emails about finding talent and growing my agency.  
 Yes, I understand and agree to the ProFolio [Terms of Service](#), including the [User Agreement](#) and [Privacy Policy](#).

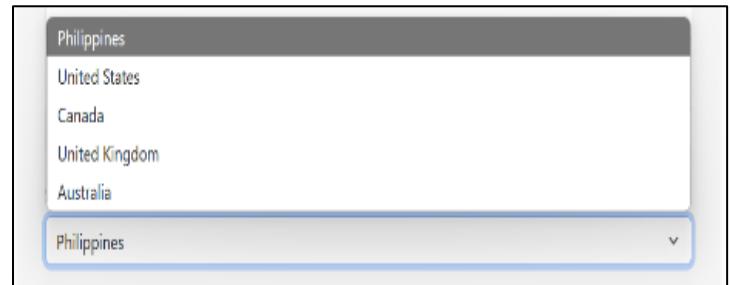
Create my account

Already have an account? [Log In](#)

© 2025 ProFolio - Where you let your work shine and watch opportunities follow

Gives users the option to receive email tips or project suggestions.

Users must agree before creating their account.



## • Register – as a Freelancer

- **Feature:** Freelancer Registration with Google Sign-In Integration
- **Use Case:** Enables freelancers to sign up using traditional input fields or by connecting their Google account for faster registration. This improves the user experience and minimizes manual data entry.

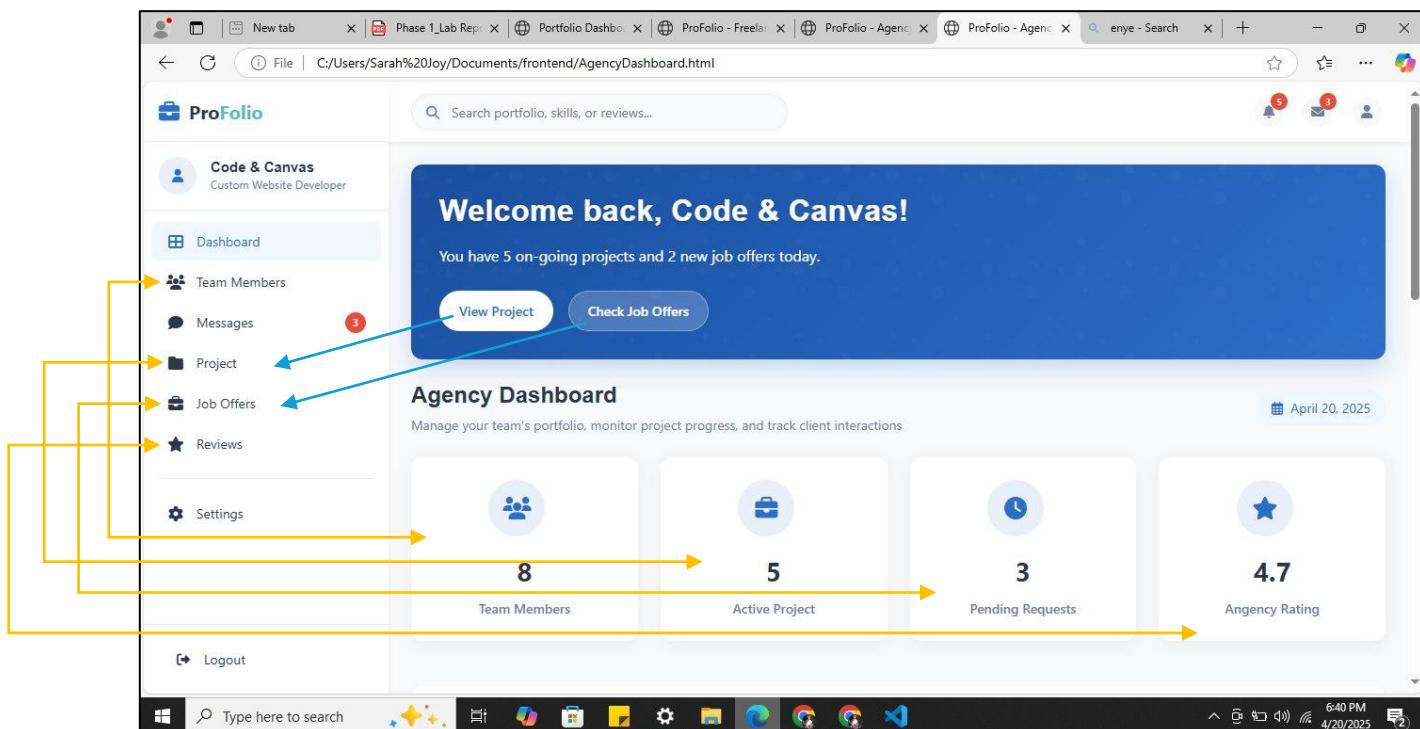
Let users sign in or register using their Google account through a secure process. This saves time and avoids manual data entry. Once authenticated, Google shares basic info like name and email with the platform to complete registration or login.

The screenshot shows the ProFolio sign-up page for freelancers. At the top, it says "Sign up as a freelancer" and "Sign up to showcase your expertise". Below this is a blue button labeled "G Continue with Google". Underneath the button, there is a horizontal line with the word "or" in the center. Following the "or" are two input fields: "First name\*" and "Last name\*". Below these fields is another input field labeled "Email\*".

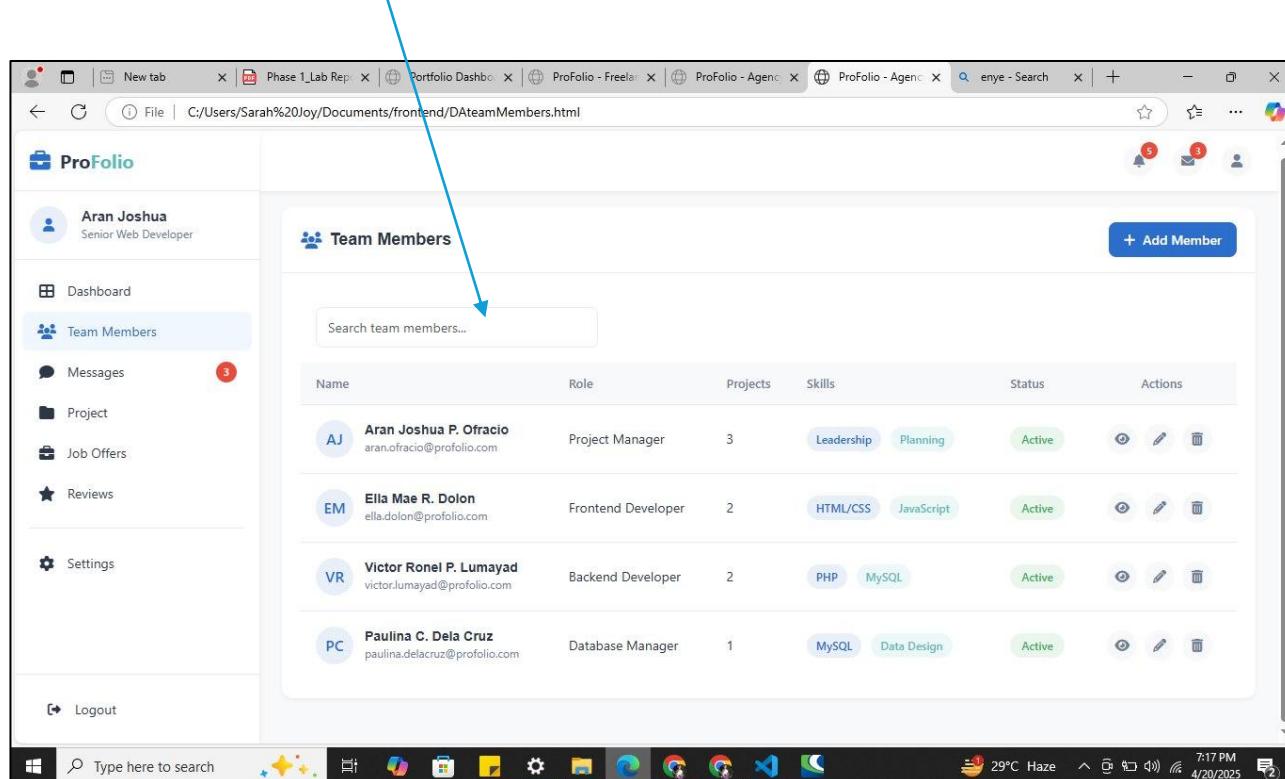
The screenshot shows the ProFolio sign-up page with traditional input fields. It includes fields for "Email\*", "Password\*", "Country" (set to Philippines), and checkboxes for "Send me helpful emails to find rewarding work and job leads." and "Yes, I understand and agree to the ProFolio [Terms of Service](#), including the [User Agreement](#) and [Privacy Policy](#)". At the bottom is a blue "Create my account" button and a link "Already have an account? [Log In](#)".

## • Agency – Dashboard

- **Feature:** Overview of Agency Dashboard
- **Use Case:** The dashboard provides agencies an all-in-one view of ongoing projects, available job offers, and tools to search for, add, and manage team members.



Enables member search.



## ● Agency – Team Members

➤ **Feature:** Overview of Team Management

➤ **Use Case:** Enables agencies to view, organize, and manage their internal team members within the platform.

Displays the member being searched for.

The screenshot shows the 'Team Members' page of the ProFolio platform. On the left is a sidebar with navigation links: Dashboard, Team Members (selected), Messages, Project, Job Offers, Reviews, Settings, and Logout. The main area has a header 'Team Members' with a search bar containing 'Aran'. Below is a table with columns: Name, Role, Projects, Skills, Status, and Actions. A row for 'Aran Joshua P. Ofrecio' is selected, showing details: Name (Aran Joshua P. Ofrecio), Role (Project Manager), Projects (3), Skills (Leadership, Planning), Status (Active), and Actions (Edit, Delete). A blue arrow points from the 'Aran' in the search bar to the selected row.

Enables agencies to add members to their teams, input member information, set role and status, and assign projects.

The screenshot shows the 'Add Member' modal window. It includes fields for Full Name (Aran Joshua P. Ofrecio), Email (aran.ofrecio@profolio.com), Role (Project Manager), Projects Assigned (3), Skills (HTML/CSS, JavaScript, PHP, MySQL, UI/UX, Leadership, Planning, Data Design), and Status (Active). At the bottom are 'Cancel' and 'Update Member' buttons. A blue arrow points from the 'Aran' in the full name field to the 'Update Member' button.

A dropdown containing multiple roles a member can be assigned to.

The screenshot shows the 'Team Members' section of the ProFolio application. On the left sidebar, under 'Team Members', there is a 'Messages' item with a red notification badge showing '3'. The main area displays a table of team members with columns for Name, Role, Status, and Actions. A modal window is open for adding a new member. It contains fields for 'Full Name' (with placeholder 'Enter full name'), 'Email' (placeholder 'Enter email address'), and 'Role' (a dropdown menu titled 'Select role' listing: Project Manager, Frontend Developer, Backend Developer, UI/UX Designer, Database Manager, Quality Assurance, and DevOps Engineer). The 'Active' status is selected in another dropdown below. At the bottom of the modal are 'Cancel' and 'Save Member' buttons. A blue arrow points from the text above to the 'Role' dropdown.

A dropdown menu for selecting a status.

The screenshot shows the same 'Team Members' section of the ProFolio application. The 'Messages' item on the sidebar still has a red '3' badge. The main area shows the same table of team members. A modal window is open for adding a new member, identical to the one in the previous screenshot. A blue arrow points from the text above to the 'Status' dropdown in the modal. This dropdown menu lists 'Active', 'Inactive', and 'Pending', with 'Active' being the selected option. The rest of the modal and application interface are consistent with the first screenshot.

The screenshot shows the 'Team Members' section of the ProFolio application. It lists four team members: Aran Joshua P. Ofrecio (Project Manager), Ella Mae R. Dolon (Frontend Developer), Victor Ronel P. Lumayad (Backend Developer), and Paulina C. Dela Cruz (Database Manager). Each member has a profile icon, name, role, projects, skills, status, and actions (Edit, Delete). A blue arrow points from the 'Edit' icon in the top right corner of the first member's row to a callout box.

An icon to view the member's profile.

Enables editing of the member's profile.

The screenshot shows the 'Edit Team Member' modal for Aran Joshua P. Ofrecio. It includes fields for Full Name (Aran Joshua P. Ofrecio), Email (aran.ofrecio@profolio.com), Role (Project Manager), Projects Assigned (3), Skills (Leadership, Planning), Status (Active), and Actions (Edit, Delete). A blue arrow points from the 'Edit Profile' button in the original list to this modal.

The screenshot shows the 'Edit Team Member' modal for Aran Joshua P. Ofrecio. It includes fields for Full Name (Aran Joshua P. Ofrecio), Email (aran.ofrecio@profolio.com), Role (Project Manager), Projects Assigned (3), Skills (HTML/CSS, JavaScript, PHP, MySQL, UI/UX, Leadership), Status (Planning, Data Design), and Actions (Edit, Delete). A blue arrow points from the 'Edit' icon in the top right corner of the member's row in the list to this modal.

The screenshot shows the 'Team Members' section of the ProFolio application. It lists the same four team members as the previous screenshots. A blue arrow points from the 'Delete' icon in the top right corner of the first member's row to a callout box.

Removes a member from the team.

The screenshot shows a 'Confirm Delete' dialog box asking if the user is sure they want to delete the team member. It includes 'Cancel' and 'Delete' buttons. A blue arrow points from the 'Delete' button in the dialog to the dialog itself.

## ● Job Offers

- **Feature:** Job Offer Management
- **Use Case:** The job offer management section allows agencies to view and manage available job offers, helping them easily track the status of interested clients.

The screenshot shows the ProFolio web application interface. On the left is a sidebar with navigation links: Dashboard, Team Members, Messages (with 3 notifications), Project, Job Offers (selected), Reviews, and Settings. Below the sidebar is a logout link. The main content area is titled "Job Offers" and displays two job offers: "E-commerce Website Development" by Sarah Johnson - Fashion Brand and "Mobile App UI Redesign" by Thomas Green - FitTrack. Each offer card includes a "New Offer" button. Below the cards, there is a "Project Description" section for each, followed by a list of required skills: E-commerce, Web Development, Payment Integration, UI/UX Design, Mobile, Material Design, and Wireframing. At the bottom of the main content area are three buttons: "Accept Offer" (green), "Decline" (red), and "View Details". The browser's address bar shows the file path: C:/Users/Sarah%20Joy/Documents/frontend/DAJobOffers.html. The system tray at the bottom right shows the date as Sunday, April 20, 2025, the time as 8:43 PM, and the weather as 30°C Clear.

This screenshot is identical to the one above, showing the ProFolio Job Offers page. Three callout boxes with arrows point to the "Accept Offer", "Decline", and "View Details" buttons. The "Accept Offer" button is highlighted with a green border and contains the text "Agree to take on the job and start the project.". The "Decline" button is highlighted with a red border and contains the text "Politely reject the offer.". The "View Details" button is highlighted with a blue border and contains the text "See more information about the job before deciding.".

Agree to take on the job and start the project.

Politely reject the offer.

See more information about the job before deciding.

## ● Client Dashboard

- **Feature:** Overview of Client Dashboard
- **Use Case:** The dashboard provides clients with a centralized view of active projects, total spending, feedback history, and freelancers hired, while also offering recommended talents with a quick “Hire Me” option for easy hiring.

The screenshot shows a web browser window for 'ProFolio' with the URL 'C:/Users/Sarah%20Joy/Documents/frontend/dashboardClient.html'. The page is titled 'Welcome back, Aran!' and displays a summary of client activity: 5 active projects, 12 freelancers hired, 18 pieces of feedback given, and a total spent of \$12,430. The left sidebar includes links for Dashboard, Project, Messages (with 3 notifications), Reviews, and Settings. A 'Logout' button is at the bottom. The taskbar at the bottom shows various application icons and the date/time as 4/20/2025, 11:07 PM.

The screenshot shows the 'Recommended Talents' section of the Client Dashboard. It features three talent profiles: Sarah Johnson (Full Stack Developer, available now), Michael Chen (UI/UX Designer, available next week), and Jessica Patel (Digital Marketing Specialist, available now). Each profile includes a thumbnail, name, title, experience, skills, reviews, and a 'HIRE ME' button. The left sidebar is identical to the previous screenshot, and the taskbar at the bottom is visible.

## • Freelancer - Dashboard

- Feature: Overview panel and navigation hub
- Use Case: This functionality gives freelancers a summary of recent activity, including profile views, messages, and engagement stats. It helps users stay informed about their performance and navigate to key actions like viewing the portfolio or checking messages.

The screenshot shows the ProFolio Freelancer Dashboard. At the top, a blue banner displays a welcome message: "Welcome back, Aran!" followed by a summary of activity: "Your profile has received 15 new views this week and 3 unread messages. Your portfolio has gained 25% more engagement compared to last month." Below the banner are two buttons: "View Portfolio" and "Check Messages". The main area is titled "Freelancer Dashboard" with the subtitle "Track your portfolio performance and client engagement". It features four performance metrics: 1,248 (Portfolio Views), 15 (New Views), 92 (Reviews), and 87% (Engagement). On the left sidebar, there's a user profile for "Aran Joshua, Senior Web Developer" and a navigation menu with links for Dashboard, Portfolio, Messages (with a red notification badge), Reviews, Settings, and Logout. The top right corner shows notification badges for 5 messages and 3 unread messages, along with a user icon.

Redirects to the 'Portfolio' tab.

Redirects to the 'Messages' tab.

Displays real-time date for users to keep track of their schedules and time-sensitive agendas.

Provides a preview of some of the user's main portfolios that they want to showcase.

Redirects to 'Messages' tab.

The screenshot shows the "Featured Portfolio Items" and "Recent Messages" sections of the dashboard. In the "Featured Portfolio Items" section, there are two items: "Web Development Portfolio" (with a blue icon) and "UI/UX Design Portfolio" (with a green icon). Each item has a "View Details" button. Below the first item, it says "Front-End React Node.js" and "A collection of responsive, performant web applications with modern architecture and seamless user experiences." In the "Recent Messages" section, there are two messages from "Sarah Wilson" and "Michael Thompson". Sarah's message says: "Hi Aran, I was really impressed with your portfolio work. I'd love to discuss potential collaboration on an upcoming project...". Michael's message says: "I really loved your portfolio, especially...". Both messages have "Reply" and "Read" buttons. The sidebar and top navigation are identical to the previous screenshot.

The screenshot shows the ProFolio dashboard. On the left, there's a sidebar with a profile picture of Aran Joshua (Senior Web Developer), a search bar, and navigation links for Dashboard, Portfolio, Messages (with 3 notifications), Reviews, and Settings. At the bottom of the sidebar is a Logout link. The main content area has a header "Recent Activity". It displays three notifications: 1. "New Portfolio Item Added" (2 hours ago) - You added "Brand Identity Package for Tech Startup" to your portfolio. 2. "New Message Received" (Yesterday) - Sarah Wilson sent you a message about your web design work showcased in your portfolio. 3. "New Review Received" (2 days ago) - You received a 5-star review from Michael Thompson for your "Mobile App UI Design" work. Below these notifications is a section for "Client Testimonials" with a "View All Reviews" link.

Redirects to  
'Reviews tab.'

This screenshot shows the "Client Testimonials" page. It features three testimonial cards. 1. Robert Chen (CEO, TechSolutions Inc.) says: "Aran is an exceptional developer who delivered beyond our expectations. His attention to detail and creativity brought our brand vision to life." He has a 5-star rating. 2. Jessica Martinez (Marketing Director, Retail Brand) says: "Working with Aran was a pleasure. He understood our requirements perfectly and delivered a stunning website that has significantly increased our conversions." She also has a 5-star rating. 3. Michael Thompson (Founder, AppFusion) says: "The mobile app Aran designed for us has received outstanding feedback from users. His intuitive designs and attention to user experience made all the difference." He has a 5-star rating. At the top right of this page is a "View All Reviews" link.

## • Freelancer – Portfolio

- **Feature:** Profile and work showcase section
- **Use Case:** This functionality allows freelancers to present their skills, experience, and completed work. It helps attract potential clients by displaying relevant projects and enhancing the freelancer's professional credibility.

The **Portfolio Management** interface shows categorized project portfolios with tags, view counts, update timestamps, and options to view or edit entries. This section helps users present their work to attract potential clients.

The screenshot displays the ProFolio application interface. On the left, a sidebar menu includes 'Dashboard', 'Portfolio' (which is currently selected), 'Messages' (with 3 notifications), 'Reviews', and 'Settings'. The main area is titled 'Portfolio Management' and features a search bar at the top. Below it, there are two portfolio entries:

- Web Development Portfolio**: Showcase of my web development projects, including responsive websites, progressive web apps, and interactive UI components. It lists 8 projects, 245 views, and was updated 5 days ago. Buttons for 'View Portfolio' and 'Edit' are available.
- UI/UX Design Portfolio**: Collection of my design work including user interfaces, wireframes, prototypes, and case studies on design thinking. It lists 6 projects, 183 views, and was updated 2 weeks ago. Buttons for 'View Portfolio' and 'Edit' are available.

At the top right of the main area, there are notification icons for messages and reviews.

The **Portfolio Tips** section, offering best practices for optimizing project showcases. It highlights three key suggestions: targeting specific audiences, presenting top-quality work, and keeping portfolio content regularly updated.

The screenshot shows the 'Portfolio Tips' section within the ProFolio application. The sidebar on the left remains the same. The main content area is titled 'Portfolio Tips' and contains three tips:

- Target Your Audience**: Customize your portfolio for different client types. Create industry-specific showcases that highlight relevant experience.
- Showcase Your Best Work**: Quality over quantity. Include your strongest projects and explain your role, approach, and results achieved.
- Keep It Updated**: Regularly refresh your portfolio with new projects. Remove older work that no longer represents your skills.

Starts the multi-step form  
for building a portfolio.

The screenshot shows the ProFolio dashboard for Aran Johnson, Senior Web Developer. On the left is a sidebar with links for Dashboard, Portfolio (selected), Messages (3 notifications), Reviews, and Settings. The main area displays two portfolio cards:

- Web Development Portfolio**: Showcase of my web development projects, including responsive websites, progressive web apps, and interactive UI components. It includes a screenshot of a Node.js application with code: 

```
projects = [ { title: 'E-commerce Platform', tech: ['React', 'Node.js', 'MongoDB'] }, { title: 'Travel App', tech: ['Vue', 'Express'] } ]; renderProjects(projects);
```

. Tags: React, Node.js, Full-Stack, JavaScript. Stats: 245 views, 8 projects, Updated 5 days ago. Buttons: View Portfolio, Edit.
- UI/UX Design Portfolio**: Collection of my design work including user interfaces, wireframes, prototypes, and case studies on design thinking. It includes a screenshot of a Figma component structure diagram. Tags: Figma, Wireframing, Prototyping, User Research. Stats: 183 views, 6 projects, Updated 2 weeks ago. Buttons: View Portfolio, Edit.

In the bottom right corner, there is a large blue button labeled "+ Create Portfolio" with the text "Showcase your work in a new professional portfolio tailored to specific skills or industries." above it.

The screenshot shows the same ProFolio dashboard as above, but now with a modal window open over the content. The modal has a blue header with a plus sign and the text "Create New Portfolio". Below the header is a descriptive text: "Showcase your work in a new professional portfolio tailored to specific skills or industries." At the bottom of the modal is a blue button labeled "+ Create Portfolio".

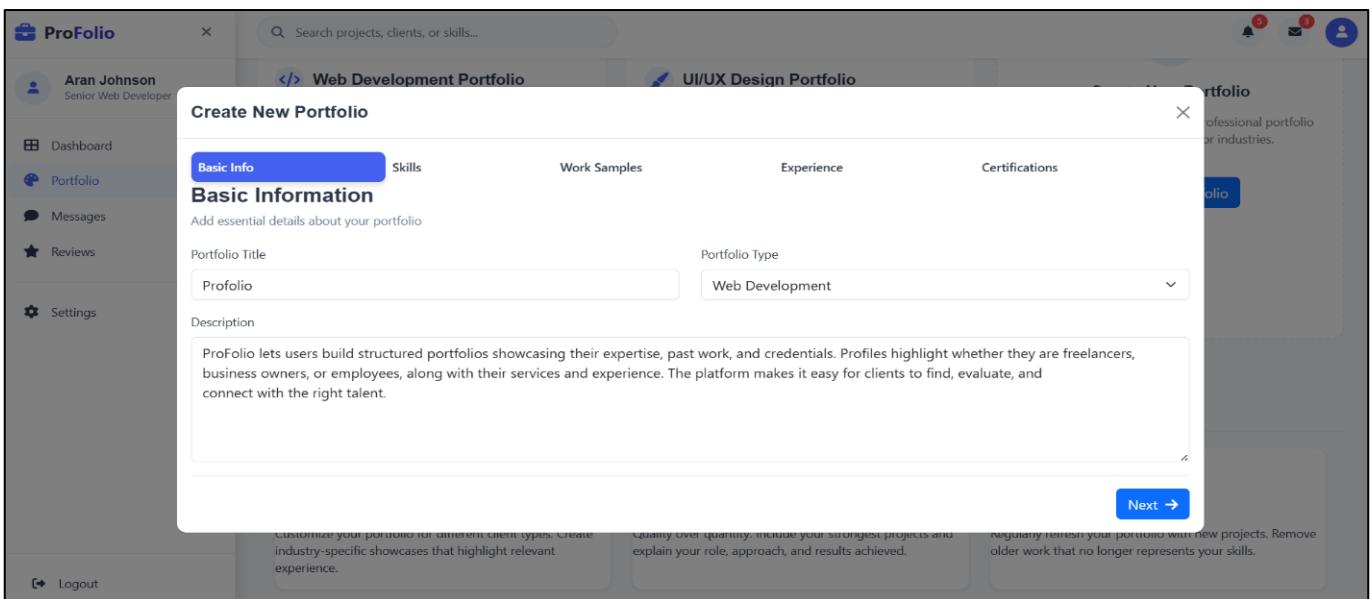
The main content area below the modal shows the "Portfolio Tips" section from the previous screenshot.

**Target Your Audience**: Customize your portfolio for different client types. Create industry-specific showcases that highlight relevant experience.

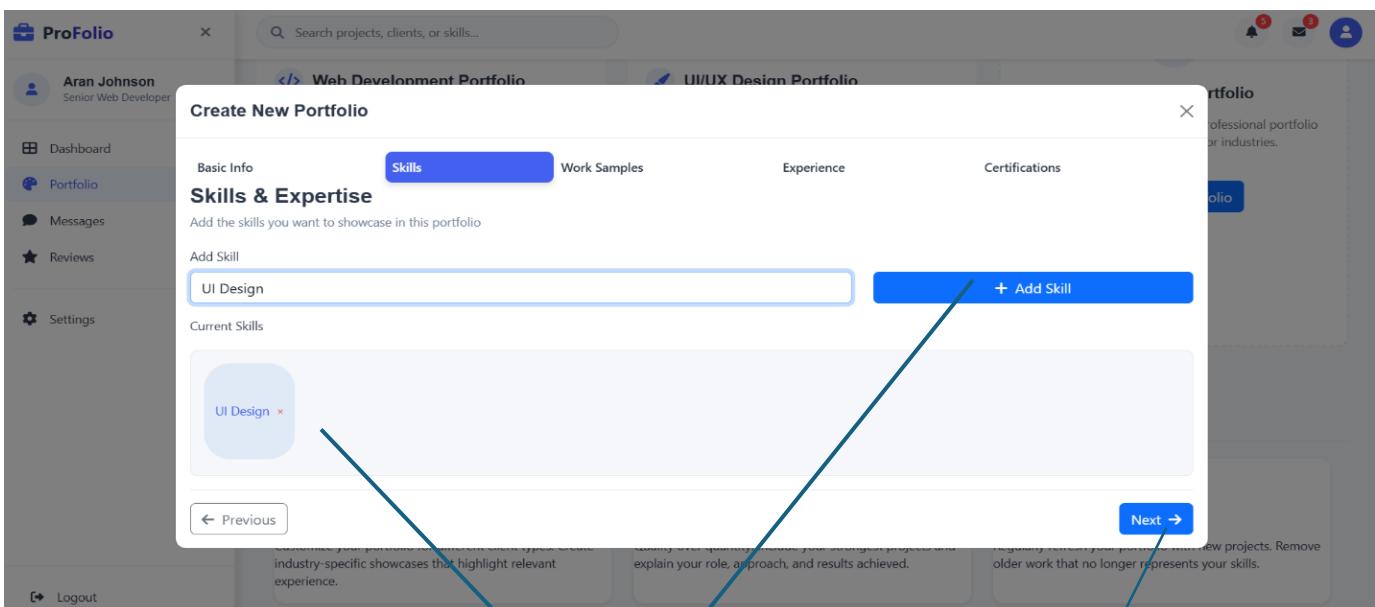
**Showcase Your Best Work**: Quality over quantity. Include your strongest projects and explain your role, approach, and results achieved.

**Keep It Updated**: Regularly refresh your portfolio with new projects. Remove older work that no longer represents your skills.

At the bottom of the page, there is a copyright notice: "© 2025 ProFolio. All rights reserved." and links to "Privacy Policy • Terms of Service • Help Center".



Once the user completes filling out the **Basic Information** section, they can proceed to the next step of the portfolio builder.



Once a skill is added, it will immediately appear in the **Current Skills** section.

The system proceeds to the next step of the portfolio builder.

**Create New Portfolio**

**Work Samples**

Add examples of your best work

Project Title: ProFolio

Client/Company: ProFolio

Description: ProFolio lets users build structured portfolios showcasing their expertise, past work, and credentials. Profiles highlight whether they are freelancers, business owners, or employees, along with their services and experience. The platform makes it easy for clients to find, evaluate, and connect with the right talent.

Project URL (optional): https://

Image URL (optional): https://

+ Add Work Sample

Users can also add a **Project URL** and an **Image URL** when submitting a work sample. These will be included and displayed in the **Added Work Samples** section.

Once the user completes the work sample form and clicks the “**Add Sample**” button, the entered sample is immediately added and displayed in the “**Added Work Samples**” section.

**Added Work Samples**

ProFolio

ProFolio lets users build structured portfolios showcasing their expertise, past work, and credentials. Profiles highlight whether they are freelancers, business owners, or employees, along with their services and experience. The platform makes it easy for clients to find, evaluate, and connect with the right talent.

← Previous

Next →

**Create New Portfolio**

**Professional Experience**  
Add relevant work experience to highlight your expertise

Job Title: Web Developer      Company: ProFolio

Start Date: April 2025      End Date:

I currently work here

Description:  
Developed and maintained responsive websites, customized WordPress themes, and integrated APIs to deliver user-friendly and high-performing web solutions.

+ Add Experience

**Added Experience**

Web Developer  
ProFolio | April 2025 - Present  
Developed and maintained responsive websites, customized WordPress themes, and integrated APIs to deliver user-friendly and high-performing web solutions.

Once the user completes the professional experience sample form and clicks the “**Add Experience**” button, the entered experience is immediately added and displayed in the “**Added Work Samples**” section.

**Create New Portfolio**

Start Date:  End Date:   
 I currently work here

Description:

+ Add Experience

**Added Experience**

Web Developer  
ProFolio | April 2025 - Present  
Developed and maintained responsive websites, customized WordPress themes, and integrated APIs to deliver user-friendly and high-performing web solutions.

< Previous      Next >

The system proceeds to the next step of the portfolio builder.

**Create New Portfolio**

Basic Info Skills Work Samples Experience Certifications

### Certifications & Education

Add your certifications, courses, and educational background

Certification Name	Issuing Organization
Full-Stack Web Development Certification	Coursera in collaboration with University of Michigan
Date Issued	Certificate URL (optional)
March 2025	<a href="https://">https://</a>
Description (optional)	
Covered HTML, CSS, JavaScript, Node.js, Express, MongoDB, and deployment. Built full-stack projects with real-world use cases.	

+ Add Certification

**Added Certifications**

Add your certifications to showcase your credentials

Once the user completes the form and clicks the “**Add Certification**” button, the entered certifications is immediately added and displayed in the “**Added Certifications**” section.

**Create New Portfolio**

Date Issued Certificate URL (optional)

Description (optional)

+ Add Certification

**Added Certifications**

Full-Stack Web Development Certification  
Coursera in collaboration with University of Michigan | Mar 2025  
Covered HTML, CSS, JavaScript, Node.js, Express, MongoDB, and deployment. Built full-stack projects with real-world use cases.

← Previous

Save Portfolio

**Portfolio Management**

Showcase your work and attract potential clients

myportfolio.dev

```
projects = [ { title: 'E-commerce Platform', tech: ['React', 'Node.js', 'MongoDB'] }, { title: 'Travel App', tech: ['Vue', 'Express'] } ];
renderProjects(projects);
```

**Web Development Portfolio**

React Node.js Full-Stack JavaScript

Showcase of my web development projects, including responsive websites, progressive web apps, and interactive UI components.

245 views 8 projects Updated 5 days ago

[View Portfolio](#) [Edit](#)

design.portfolio.io

**UI/UX Design Portfolio**

Figma Wireframing Prototyping User Research

Collection of my design work including user interfaces, wireframes, prototypes, and case studies on design thinking.

183 views 6 projects Updated 2 weeks ago

[View Portfolio](#) [Edit](#)

myweb.dev.portfolio.com

</>

**ProFolio**

UI Design

ProFolio lets users build structured portfolios showcasing their expertise, past work, and credentials. Profiles highlight whether they are freelancers, business owners, or employees, along with their services and experience. The platform makes it easy for clients to find, evaluate, and connect with the right talent.

0 views 0 projects Just created

[View Portfolio](#) [Edit](#)

Once a portfolio item is saved, it will automatically appear in the Portfolio Management section with a “Just Created” label to indicate that it was recently

# Screenshot of GitHub folder structure

```
|  
+-- frontend  
|   README.md  
|  
+-- admin  
|   +-- css  
|       adminDashboard.css  
|       adminReports.css  
|       adminUsers.css  
|  
|   +-- html  
|       adminAnalytics.html  
|       adminDashboard.html  
|       adminLogin.html  
|       adminLogout.html  
|       adminPolicies.html  
|       adminReports.html  
|       adminSettings.html  
|       adminUsers.html  
|       adminVerification.html  
|       index (1).html  
|       login.html  
|       register (1).html  
|       registerAgency.html  
|       registerClient.html  
|       registerFreelancer.html  
|  
|   +-- images  
|       logo.png  
|  
\-- javascript  
    adminDashboard (2).js  
    adminDashboard.js  
    adminVerification.js
```

```
|  
+-- agency  
|   +-- css  
|       AgencyDashboard.css  
|       DAJobOffers.css  
|       DAteamMembers.css  
|  
|   +-- html  
|       AgencyDashboard.html  
|       DAJobOffers.html  
|       DAteamMembers.html  
|  
|   \-- images  
|       gitkeep  
|  
+-- client  
|   +-- css  
|       .gitkeep  
|  
|   +-- html  
|       .gitkeep  
|  
|   \-- images  
|       .gitkeep  
|  
+-- freelancer  
|   +-- css  
|       freelancerDashboard.css  
|       freelancerPortfolio.css  
|  
|   +-- html  
|       freelancerDashboard.html  
|       freelancerPortfolio.html  
|       freelancerProjects.html  
|  
|   \-- images  
|       .gitkeep
```

```
|  
+---database  
|   Table_user.pdf  
|  
+---docs  
|   +---lab11  
|       Admin_Verification.PNG  
|       Admin_Dashboard.PNG  
|       Admin_Settings.PNG  
|       Admin_User.PNG  
|       Agency_Dashboard.jpg  
|       analytics.PNG  
|       homepage.1.png  
|       homepage.2.png  
|       homepage.3.PNG  
|       homepage.4.PNG  
|       homepage.5.PNG  
|       homepage.6.PNG  
|       Job_Offers.jpg  
|       Reported_Users.PNG  
|       Team_Members.jpg  
|  
\---planning  
    Phase-1_Architecture-Diagram_ProFolio.pdf  
    Phase-1_ERD_ProFolio.pdf  
    Phase-1_Project-Proposal_ProFolio.pdf  
    Phase-1_Use-Case-Diagram_ProFolio.pdf
```

```
C:.  
|   README.md  
|  
+---backend  
|   |   server.js  
|  
|   +---connection  
|       connection.php  
|  
|   +---CSS  
|       agency.css  
|       client.css  
|       freelancer.css  
|       index.css  
|       login.css  
|       register.css  
|  
|   +---models  
|       connection.php  
|  
|   +---PHP  
|       index.php  
|       login.php  
|       register.php  
|       registerAgency.php  
|       registerClient.php  
|       registerFreelancer.php  
|  
|   \---routes  
|       connection.php
```

# Overview of Back-end Setup

The backend setup provides the foundation for managing server-side logic, user data, and secure operations in the project. It runs on a local server environment using XAMPP, which includes Apache for handling HTTP requests, PHP for backend scripting, and MySQL for managing the database. The system is designed with a clear project structure that separates logic for different user roles: clients, freelancers, and agencies each with their own registration and login functionalities.

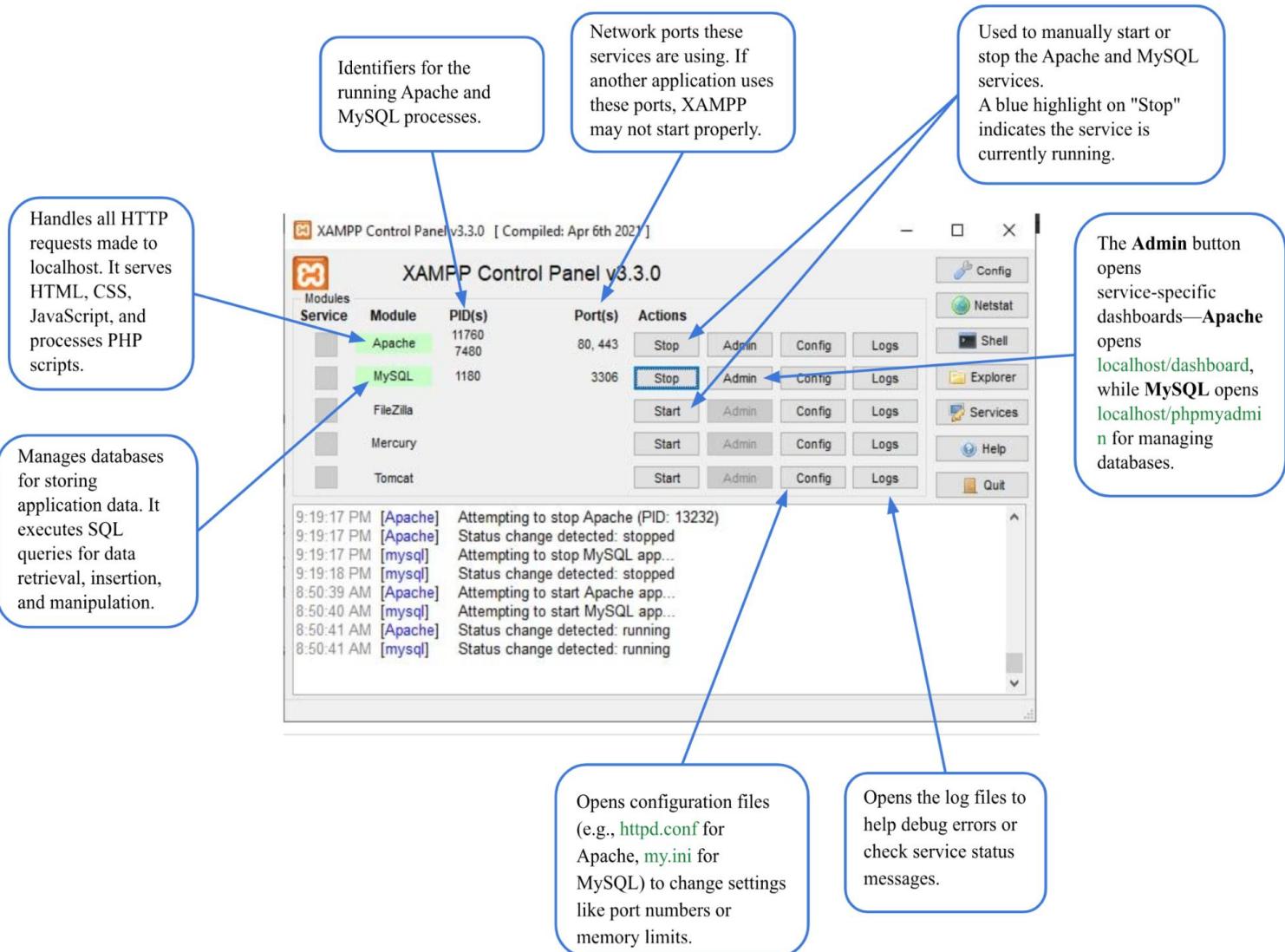
This setup also integrates Google APIs for enhanced features like address autofill or secure login, and includes a database schema to organize user information, job data, and messages. A flow diagram outlines how data moves through the system, from user input on the frontend to processing and storage on the backend, ensuring smooth interaction between all components.

## Local Server Environment

We use XAMPP as our local server environment for development, which includes Apache as our web server, PHP as our scripting language, and MySQL as our database system. When we access our web pages through localhost, Apache serves the content and handles HTTP requests. PHP runs on the Apache server to process backend logic, while MySQL manages our data storage and retrieval.

XAMPP provides a user-friendly control panel that allows us to easily start, stop, and configure services. Apache handles all HTML, CSS, JavaScript, and PHP scripts, while MySQL executes queries for data manipulation. The Admin button opens dashboards like **localhost/dashboard** for Apache and **phpMyAdmin** for MySQL, enabling us to manage our server and database visually. Additionally, we can access configuration files and log files to troubleshoot issues or modify default settings such as port numbers and memory limits. This setup is essential for testing dynamic websites and applications before deployment.

## ProFolio Structure



The backend architecture of the Profolio project follows a modular and organized structure designed to promote maintainability, scalability, and ease of collaboration. The root directory **/PROFOLIO/** is divided into several key subdirectories and PHP scripts, each serving a specific purpose within the system.

```
/PROFOLIO/
└── connection/
    └── connection.php      → central DB connection file
└── CSS/
└── dashboard/
└── PHP/
    ├── index.php          → landing page (entry point)
    ├── login.php           → login logic (backend)
    ├── register.php        → general registration handler
    ├── registerAgency.php  → agency-specific registration
    ├── registerClient.php  → client-specific registration
    └── registerFreelancer.php  → freelancer-specific registration
└── uploads/              → stores uploaded files
```

## /connection/

- **connection.php**

This file establishes a centralized MySQL database connection using the `mysqli` extension. By isolating database connectivity here, it promotes code reusability and ensures that changes to connection parameters (like host, username, or password) only need to be updated in one place.

## /CSS/

- This directory is dedicated to frontend styling, including files such as `main.css`. It provides a central location for all design-related resources, ensuring that styling is separated from logic and functionality. This aligns with best practices for separation of concerns.

## /dashboard/

- Contains files related to the user dashboard interface. Once a user successfully logs in or registers, they are directed to this area where they can interact with the platform (e.g., view messages, update profiles, track activity). This modular approach allows for future expansion such as separate dashboards for freelancers, clients, and agencies.

## /PHP/

This folder contains the core backend logic, with each script fulfilling a specific function in the authentication and registration workflow:

- **index.php**  
Acts as the entry point or landing page of the application. It may contain redirection logic, basic navigation, or initial session checks.
- **login.php**  
Handles the login logic, including input validation, authentication against the database, session creation, and user role redirection.
- **register.php**  
Serves as a general registration handler, possibly including logic to route users based on the type they select (freelancer, client, or agency) and basic validation common to all types.

- registerAgency.php  
Responsible for agency-specific registration logic. This could include collecting and validating additional fields such as company name, registration documents, or team member details.
- registerClient.php  
Manages the registration process for clients, likely capturing information such as company or individual client details and project preferences.
- registerFreelancer.php  
Dedicated to freelancer registration, possibly handling fields such as skills, experience, portfolio links, and availability.

## /uploads/

- This directory is used to store user-uploaded files such as:
  - Profile images
  - Resumes
  - Portfolio samples

## Database

The Profolio database is designed as the backbone of the platform, efficiently managing all critical user and business data for a freelancing ecosystem that connects agencies, freelancers, and clients. It is built using MySQL, with a schema structured for scalability, flexibility, and performance. At its core, the database is focused on supporting a role-based user system, where individuals can register as freelancers or clients, and organizations can onboard as agencies. Each user regardless of their role is represented within a unified data model that ensures easy access control, extensibility, and secure authentication

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
CREATE TABLE _user ( id INT AUTO_INCREMENT PRIMARY KEY, first_name VARCHAR(50) NOT NULL, last_name VARCHAR(50) NOT NULL, email VARCHAR(100) NOT NULL UNIQUE, password VARCHAR(255) NOT NULL, country VARCHAR(100), city VARCHAR(255), role VARCHAR(255) NOT NULL DEFAULT 'User', agency_name VARCHAR(100) NOT NULL, agency_type VARCHAR(255) NOT NULL, agency_logo VARCHAR(255), team_size VARCHAR(255), agency_description TEXT, services LONGTEXT, agency_website VARCHAR(255), agency_title VARCHAR(100), created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	int(11)			No	<code>None</code>		AUTO_INCREMENT	Change  Drop  More
2	<code>first_name</code>	varchar(50)	utf8mb4_general_ci		No	<code>None</code>			Change  Drop  More
3	<code>last_name</code>	varchar(50)	utf8mb4_general_ci		No	<code>None</code>			Change  Drop  More
4	<code>email</code>	varchar(100)	utf8mb4_general_ci		No	<code>None</code>			Change  Drop  More
5	<code>password</code>	varchar(255)	utf8mb4_general_ci		No	<code>None</code>			Change  Drop  More
6	<code>country</code>	varchar(100)	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
7	<code>city</code>	varchar(255)	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
8	<code>role</code>	varchar(255)	utf8mb4_general_ci		No	<code>User</code>			Change  Drop  More
9	<code>agency_name</code>	varchar(100)	utf8mb4_general_ci		No	<code>None</code>			Change  Drop  More
10	<code>agency_type</code>	varchar(255)	utf8mb4_general_ci		No	<code>None</code>			Change  Drop  More
11	<code>agency_logo</code>	varchar(255)	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
12	<code>team_size</code>	varchar(255)	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
13	<code>agency_description</code>	text	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
14	<code>services</code>	longtext	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
15	<code>agency_website</code>	varchar(255)	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
16	<code>agency_title</code>	varchar(100)	utf8mb4_general_ci		Yes	<code>NULL</code>			Change  Drop  More
17	<code>created_at</code>	timestamp			No	<code>current_timestamp()</code>			Change  Drop  More

As the first step in setting up the Profolio database, we created the `_user` table. It is the core table that powers all user interactions on the platform. This table serves as the backbone for managing user data, supporting various roles such as freelancers, clients, and agencies within a unified structure. It captures essential user information including names, email, password, and location data, while also accommodating role-based distinctions through the `role` field. For agency accounts, additional fields such as `agency_name`, `agency_type`, `services`, and `team_size` enable detailed profiles that help users better showcase their brand and capabilities.

## Attributes that make up the `_user` table:

### 1. Primary Identification

- `id` (INT, AUTO\_INCREMENT, PRIMARY KEY):  
A unique numeric identifier for each user. It auto-increments with every new record insertion, ensuring consistent and unambiguous user identification across the platform.

### 2. Personal & Authentication Details

- `first_name` (VARCHAR(50), NOT NULL):  
Stores the user's first name. Required for personalization and account identification.
- `last_name` (VARCHAR(50), NOT NULL):  
Stores the user's last name. Also required.
- `email` (VARCHAR(100), NOT NULL, UNIQUE):  
Functions as the unique login identifier. The UNIQUE constraint ensures no two users can register with the same email.
- `password` (VARCHAR(255), NOT NULL):  
Stores the user's hashed password (usually hashed using algorithms like bcrypt). The extended length supports modern hashing formats.

### **3. Location-Based Information**

- **country** (VARCHAR(100), NULLABLE):  
Stores the country associated with the user. Useful for localizing content, filtering users by region, and platform analytics.
- **city** (VARCHAR(255), NULLABLE):  
Optional city field for more granular geographic categorization.

### **4. User Role & Access Control**

- **role** (VARCHAR(255), NOT NULL, DEFAULT 'User'):  
Defines the user's function or access level within the platform. Supported values may include:
  - Freelancer
  - Agency Owner
  - Client
  - Admin

### **5. Agency-Specific Fields**

- **agency\_name** (VARCHAR(255), NULLABLE):  
Name of the agency the user represents or manages.
- **agency\_type** (VARCHAR(255), NULLABLE):  
Categorizes the agency (e.g., Design, Development, Marketing). Enables filtering and discovery by type.
- **agency\_logo** (VARCHAR(255), NULLABLE):  
Path or URL to the uploaded agency logo image file.
- **team\_size** (VARCHAR(255), NULLABLE):  
Describes the size of the team. Helps clients understand the scale of the agency.
- **agency\_description** (TEXT, NULLABLE):  
A rich-text field for agencies to describe their background, mission, expertise, or values.
- **services** (LONGTEXT, NULLABLE):  
Lists the services offered by the agency. Stored as long text to accommodate detailed descriptions or structured data.
- **agency\_website** (VARCHAR(255), NULLABLE):  
Stores a URL to the agency's website for external verification or portfolio browsing.
- **agency\_title** (VARCHAR(100), NULLABLE):  
A custom title displayed on profiles or dashboards for added personalization.

## Registration Page

Our registration page serves as the primary entry point for new users to join the Profolio platform. It is designed to onboard users efficiently and securely, whether they are freelancers, clients, or agencies. When a user submits the registration form, our backend PHP script takes over to handle the logic required to validate, process, and store the new account data.

This process begins with capturing all submitted data such as the user's **first name**, **last name**, **email**, **password**, **country**, and predefined **role** (`client`, `freelancer`, or `agency_owner`). To ensure security, each of these inputs is sanitized using `mysqli_real_escape_string()` to guard against SQL injection attacks.

This PHP code manages user registration. It first checks if a user's email already exists in the database. If not, it inserts the user's data into the database and redirects the user to the login page. If the email already exists, it alerts the user. However, the original code is vulnerable to SQL injection and should be rewritten using parameterized queries for security.

```
// Check if the email already exists
$checkEmailSql = "SELECT * FROM _user WHERE email = '$email'";
$result = $con->query($checkEmailSql);

if ($result->num_rows > 0) {
    echo "<script>alert('Email already exists. Please use a different email.');" </script>";
} else {
    // Insert the data if email doesn't exist
    $sql = "INSERT INTO _user (first_name, last_name, email, password, country, role)
            VALUES ('$firstName', '$lastName', '$email', '$password', '$country', '$role')";

    if ($con->query($sql)) {
        echo "<script>alert('Registration successful!'); window.location.href='login.php';</script>";
    } else {
        echo "Error: " . $con->error;
    }
}
?>
```

- `$checkEmailSql = "SELECT * FROM _user WHERE email = '$email'"`
  - This line constructs an SQL query to select all columns (SELECT \*) from the `_user` table where the `email` column matches the value stored in the `$email` variable. This variable presumably comes from the user's registration form submission.
- `$result = $con->query($checkEmailSql);`
  - This line executes the SQL query using the database connection object `$con` (which is assumed to be established elsewhere in the code). The result of the query (either a rowset or an error) is stored in the `$result` variable.
- `if ($result->num_rows > 0) { ... }`
  - This condition checks if the query returned any rows. If `$result->num_rows` is greater than 0, it means an email address matching the user's input already exists in the database.
- `echo "<script>alert('Email already exists. Please use a different email.');" </script>"`
  - If the email already exists, this line uses JavaScript to display an alert message to the user, prompting them to use a different email address.
- `} else { ... }`
  - This block executes only if the email address is unique (no rows were returned from the previous query).

- \$sql = "INSERT INTO \_user (first\_name, last\_name, email, password, country, role) VALUES ('\$firstName', '\$lastName', '\$email', '\$password', '\$country', '\$role')";
  - This line constructs an SQL query to insert a new user record into the \_user table. It inserts values for first\_name, last\_name, email, password, country, and role, all taken from variables presumably populated from the user's registration form. Important Security Note: This code is vulnerable to SQL injection. Using parameterized queries is crucial to prevent this vulnerability.
- if (\$con->query(\$sql)) { ... }
  - This line executes the insertion query.
- echo "<script>alert('Registration successful!'); window.location.href='login.php';</script>";
  - If the insertion was successful, this line displays a success message using JavaScript and redirects the user to the login.php page.
- } else { echo "Error: " . \$con->error; }
  - If the insertion fails, this line displays an error message including the database error message from \$con->error.

This PHP code handles user registration via Google Sign-In. It checks if the user's email already exists; if not, it adds the user to the database (leaving the password field blank as Google handles authentication). If the email exists, it redirects the user to the login page. Crucially, the original code needs improvement to prevent SQL injection vulnerabilities. The improved version uses prepared statements for enhanced security.

```

8 // Handle Google Sign-Up
9
10 if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['google_signup'])) {
11     $firstName = isset($_POST['first_name']) ? mysqli_real_escape_string($con, $_POST['first_name']) : '';
12     $lastName = isset($_POST['last_name']) ? mysqli_real_escape_string($con, $_POST['last_name']) : '';
13     $email = isset($_POST['email']) ? mysqli_real_escape_string($con, $_POST['email']) : '';
14     $role = 'client';
15
16     if (!empty($email)) {
17         $checkUserSql = "SELECT * FROM _user WHERE email = '$email'";
18         $result = $con->query($checkUserSql);
19
20         if ($result->num_rows === 0) {
21             // No user yet - proceed with Google sign-up
22             $sql = "INSERT INTO _user (first_name, last_name, email, password, country, role)
23                   VALUES ('$firstName', '$lastName', '$email', '', '', '$role')";
24             $con->query($sql);
25             echo json_encode(['status' => 'success', 'message' => 'Signed in with Google!']);
26             exit;
27         } else {
28             // Email already exists - do NOT allow Google sign-up again
29             echo json_encode([
30                 'status' => 'redirect',
31                 'message' => 'Email already registered. Redirecting to login...',
32                 'redirect_url' => 'login.php'
33             ]);
34             exit;
35         }
36     }
37
38     // Final fallback to ensure this block does NOT fall through to regular registration
39     exit;
40 }

```

- if (\$\_SERVER["REQUEST\_METHOD"] == "POST" && isset(\$\_POST['google\_signup'])) { ... }
  - This checks if the request method is POST and if the google\_signup key exists in the \$\_POST array (Google sign-up form was submitted).
- \$firstName = isset(\$\_POST['first\_name']) ? mysqli\_real\_escape\_string(\$con, \$\_POST['first\_name']) : "";
- \$lastName = isset(\$\_POST['last\_name']) ? mysqli\_real\_escape\_string(\$con, \$\_POST['last\_name']) : "";
- \$email = isset(\$\_POST['email']) ? mysqli\_real\_escape\_string(\$con, \$\_POST['email']) : "";
  - These lines retrieve values from the POST data (\$\_POST), representing user information (first name, last name, email), and sanitize them using mysqli\_real\_escape\_string(\$con, ...) to prevent SQL injection. \$con is assumed to be a database connection object. Important: While this sanitization helps, it's still not as robust as prepared statements.

- `if(empty($email)) { ... }`
  - This checks if the email is empty. If so, it skips the database check (although this is unusual behavior and might indicate a flaw in the input handling).
- `$checkUserSql = "SELECT * FROM _user WHERE email = '$email';"`
  - This constructs an SQL query to check if a user with the given email already exists.
- `$result = $con->query($checkUserSql);`
  - This executes the query.
- `if ($result->num_rows == 0) { ... }`
  - This checks if no user with the email was found.
- `$sql = "INSERT INTO _user (first_name, last_name, email, password, country, role) VALUES ('$firstName', '$lastName', '$email', '$password', '$country', '$role')";`
  - This inserts a new user into the `_user` table. Note that the `password` field is left empty, which is appropriate for Google sign-up since authentication happens through Google.
- `$con->query($sql);`
  - This executes the insertion query.
- `echo json_encode(['status' => 'success', 'message' => 'Signed in with Google!']);`
  - This sends a JSON response indicating successful sign-up.
- `exit;`
  - This terminates the script execution.
- `} else { ... }`
  - This block executes if a user with the given email already exists.
- `echo json_encode(['status' => 'redirect', 'message' => 'Email already registered. Redirecting to login...', 'redirect_url' => 'login.php']);`
  - This sends a JSON response indicating that the user is already registered and should be redirected to the login page.
- `exit;`
  - This terminates the script execution.

## Login Page

The ProFolio login page features a clean and modern design with two main authentication options: traditional email/password login and Google Sign-In. Users can enter their credentials manually or opt to sign in quickly using their Google account. The "Sign in with Google" button triggers a Google account selection popup, making the login process faster and more convenient. This integration enhances user experience by providing a seamless and secure way to access the platform without the need to remember additional login details.

The screenshot shows the ProFolio login interface on the left and a Google sign-in modal on the right.

**ProFolio Login Page:**

- The title bar says "Log in to ProFolio".
- A red error message box contains the text "Invalid email or password."
- Input fields for "Email" and "Password" are present.
- A "Remember me" checkbox and a "Forgot" link are below the input fields.
- A blue "Continue" button is centered at the bottom.
- A "Sign in with Google" button is located at the bottom.

**Google Sign-in Modal:**

- The title bar says "Mag-sign in - Google Accounts - Google Chrome".
- The URL in the address bar is "accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount".
- The main heading is "Pumili ng account".
- The subtext is "upang magpatuloy sa ProFolio".
- Two accounts are listed:
  - Victor Ronel Lumayad** (vrl2023-3281-98217@bicol-u.edu.ph)
  - Victor Ronel Lumayad** (vplumayad@gmail.com)
- An account for **Aguilar Raizhel** (raizhelaguilar@gmail.com) is shown with a "Nag-sign out" link.
- A link for "Gumamit ng isa pang account" is available.

## Code:

```
PHP > login.php
 1  <?php
 2  session_start();
 3  include_once("../connection/connection.php");
 4  $con = connection();
 5
 6  ✓ if ($_SERVER["REQUEST_METHOD"] === "POST" && !isset($_POST['google_signup'])) {
 7      $email = $_POST['email'];
 8      $password = $_POST['password'];
 9
10      $stmt = $con->prepare("SELECT * FROM _user WHERE email = ?");
11      $stmt->bind_param("s", $email);
12      $stmt->execute();
13      $result = $stmt->get_result();
14
15  ✓  if ($result && $result->num_rows > 0) {
16      $user = $result->fetch_assoc();
17
18      // Verify the hashed password
19      ✓ if (password_verify($password, $user['password'])) {
20          $_SESSION['email'] = $user['email'];
21          $_SESSION['role'] = $user['role'];
22
23  ✓      $redirect = match ($user['role']) {
24          'freelancer' => 'freelancer_dashboard.php',
25          'client' => 'client_dashboard.php',
26          'agency' => 'agency_dashboard.php',
27          default => 'login.php'
28      };
29  }
```

```
PHP > login.php
  o   L1 ( ) { if ( (!$_REQUEST['method']) === 'POST' && !isset($_POST['google_signup'])) {
41   }
42   if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST['google_signup'])) {
43     header('Content-Type: application/json');
```

```
44
45     $email = $_POST['email'];
46     $firstName = $_POST['first_name'];
47     $lastName = $_POST['last_name'];
48
49     // Check if user exists
50     $stmt = $con->prepare("SELECT * FROM users WHERE email = ?");
51     $stmt->bind_param("s", $email);
```

```
52     $stmt->execute();
53     $result = $stmt->get_result();
54
55     if ($result->num_rows > 0) {
56         $user = $result->fetch_assoc();
57         $_SESSION['email'] = $user['email'];
58         $_SESSION['role'] = $user['role'];
59
60         // Redirect based on role
61         $redirect = match ($user['role']) {
62             'freelancer' => 'freelancer_da.
63             'client' => 'client_dashboard.
64             'agency' => 'agency_dashboard.
65             default => 'login.php'
```

```
42 if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST['google_signup'])) {
43     if ($result->num_rows > 0) {
44         echo json_encode([
45             'status' => 'redirect',
46             'message' => 'Welcome back!',
47             'redirect_url' => $redirect
48         ]);
49         exit();
50     } else {
51         // New user, redirect to registration
52         $_SESSION['google_email'] = $email;
53         $_SESSION['google_first_name'] = $firstName;
54         $_SESSION['google_last_name'] = $lastName;
55
56         echo json_encode([
57             'status' => 'redirect',
58             'message' => 'Complete your registration.',
59             'redirect_url' => 'register.php'
60         ]);
61         exit();
62     }
63 }
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2290
2291
2292
2293
2294
2295
2296
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2390
2391
2392
2393
2394
2395
2396
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2478
2479
2480
2481
2482
2483
2484
2485
2486
2
```

```
if ($_SERVER["REQUEST_METHOD"] === "POST" && !isset($_POST['google_signup'])) {  
    if ($result && $result->num_rows > 0) {  
        if (password_verify($password, $user['password'])) {
```

```
    };
    header("Location: $redirect");
    exit();
} else {
    $_SESSION['error'] = "Invalid email or password.";
}
else {
```

```
|     header("Location: login.php");
```

This login system is built using PHP and integrates both traditional email/password authentication and Google Sign-In for user flexibility and ease of access. It starts with `session_start()` to initialize a session, allowing user data to persist across pages during their session. The system connects to the database using a custom connection() function from an included file `connection.php`.

When a user submits the login form via POST, the system retrieves the entered email and password. It uses a prepared SQL statement to securely fetch the user's details from the `_user` table based on the email. To prevent SQL injection, the email is bound to the statement using `bind_param()`. After executing, it checks if the user exists, and if so, verifies the submitted password against the hashed password in the database using `password_verify()`.

#### If the login is successful:

- The user's email and role are stored in the session.
- A match statement determines where to redirect the user based on their role.
- A header is used for redirection.

#### If the credentials are invalid:

- An error message is stored in the session (`$_SESSION['error']`), which is later displayed on the login page.

When the Google Sign-In button is clicked, JavaScript initializes Google's OAuth2 client using `google.accounts.oauth2.initTokenClient()`. It requests the user's profile and email and retrieves an access token. With this token, it fetches the user's information from Google's UserInfo API.

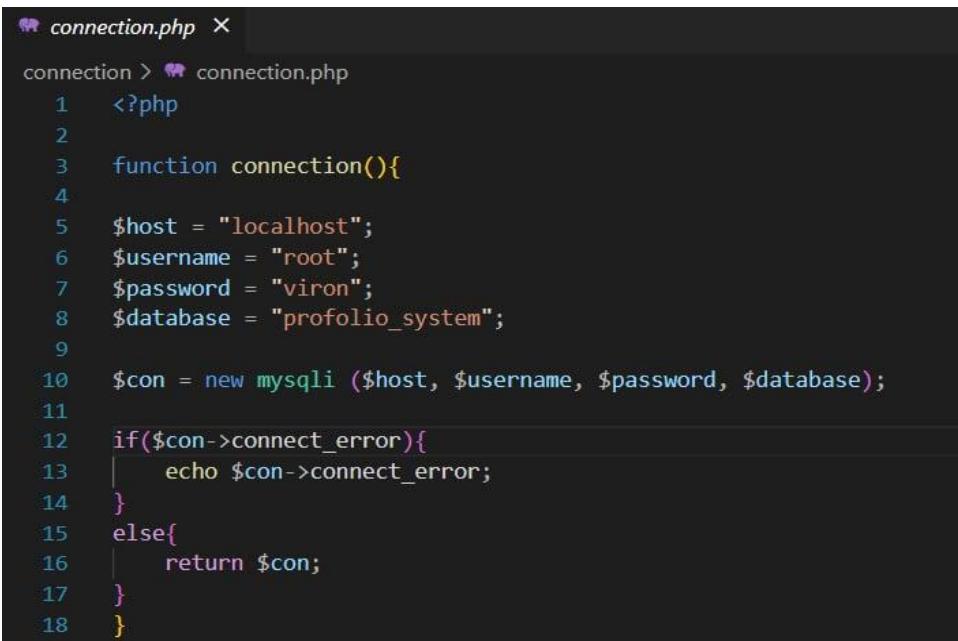
This data is then sent to the backend using an XMLHttpRequest. The PHP backend checks if the email already exists:

- If it does, it logs the user in by storing their email and role in the session and redirects them.
- If it doesn't, the user is assumed to be new. Their email and name are stored in the session temporarily, and they are redirected to a registration page to complete their account setup.

## Code snippets of Server.js, Sample model, and Routes

### Database Connection (connection.php)

The connection.php file is a centralized script that handles the connection to our MySQL database using PHP's mysqli extension. It defines a function that sets the database host, username, password, and database name, and then attempts to establish a connection using these details. If the connection fails, it outputs an error message using mysqli\_connect\_error(), which helps quickly identify issues during development. This file is included in all backend-enabled pages to ensure a consistent and reusable method of connecting to the database. Using a single connection file improves maintainability by allowing changes to be made in one place, keeps the code cleaner by avoiding repetition, and enhances organization by keeping all connection logic in one location. It also supports better security by storing sensitive database credentials in a single file, making them easier to manage and protect. Overall, connection.php plays a key role in structuring the backend of the project and helps maintain a more efficient and reliable development workflow.



A screenshot of a code editor showing the contents of a file named 'connection.php'. The code is written in PHP and defines a function 'connection()' that establishes a MySQL database connection using variables for host, username, password, and database name. It includes an if-statement to check for errors and return the connection object if successful.

```
connection.php ×
connection > connection.php
1 <?php
2
3 function connection(){
4
5     $host = "localhost";
6     $username = "root";
7     $password = "viron";
8     $database = "profolio_system";
9
10    $con = new mysqli ($host, $username, $password, $database);
11
12    if($con->connect_error){
13        echo $con->connect_error;
14    }
15    else{
16        return $con;
17    }
18 }
```

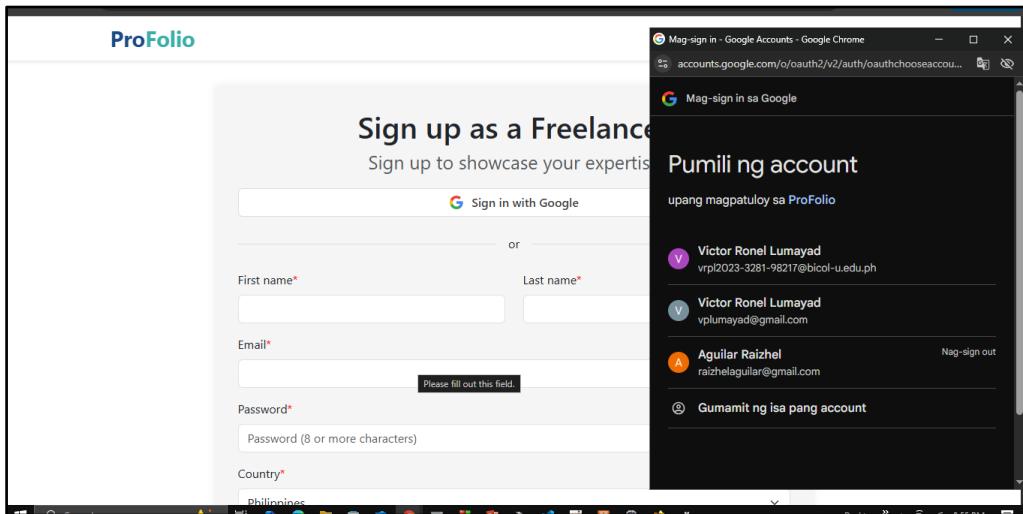
- <?php
  - This line starts the PHP code block.
- function connection(){
  - This line defines a function named connection. Functions in PHP are blocks of reusable code.
- These lines define variables that hold the database connection details:
  - \$host = "localhost";
    - The database server hostname (typically "[localhost](#)" for a local server).
  - \$username = "root";
    - The MySQL username. Important Security Note: Using "root" in a production environment is extremely risky. Create a dedicated user with limited privileges.
  - \$password = "viron";
    - The MySQL password. Important Security Note: Hardcoding passwords directly in code is a major security vulnerability. Consider using environment variables or a more secure method for storing credentials.
  - \$database = "profolio\_system";
    - The name of the database to connect to.

- `$con = new mysqli($host, $username, $password, $database);`
  - This line attempts to create a new MySQLi connection object using the credentials defined in the previous lines. mysqli is the MySQLi extension for PHP.
- This if statement checks for connection errors:
  - `if($con->connect_error) { ... }`
    - This condition checks the connect\_error property of the \$con object. If it's not empty, it means there was an error connecting to the database.
  - `echo $con->connect_error;`
    - If there was an error, this line prints the error message to the output.
- `else { return $con; }`
  - If the connection was successful, the function returns the \$con object, which can then be used to interact with the database.
- `}`
  - This closing brace ends the function definition.

# API Testing Screenshots

## registerFreelancer.php

The Google Sign-In integration was implemented on the freelancer registration form to enhance user experience by streamlining the sign-up process. During API testing, we verified the full OAuth 2.0 flow from initiating the sign-in via the "Sign in with Google" button to handling the authentication response on the server side.



- The process begins when a user clicks the "Sign in with Google" button on the freelancer registration form. This triggers a client-side redirect to Google's OAuth 2.0 authorization server.
  - A Google authentication popup window appears, prompting the user to select their Google account and grant permission for the application to access specific user data (defined in the OAuth 2.0 scope). This ensures secure authentication without requiring the application to handle user credentials directly.
  - Upon successful authentication, Google redirects the user back to a pre-configured redirect URI. This redirect includes an authorization code.
  - The registerFreelancer.php script receives the authorization code. It then uses this code to exchange it for an access token via a request to Google's OAuth 2.0 token endpoint. This exchange requires the application's client ID and client secret (kept confidential).
  - Using the access token, the script makes a request to Google's People API (or a similar Google API) to retrieve user information, including the user's name, email address, and potentially a profile picture URL.
- 
- **User Registration/Login Logic:**
    - New User
      - If the retrieved email address does not exist in the application's database, a new freelancer record is created, populating the fields with the data received from Google. A new session is created for the user.
    - Existing User
      - If the email address already exists in the database, the system verifies if the user's Google account matches the existing record. If a match is found, the user is logged in directly, creating or updating their session. No password is required in this scenario.

## REST API Functionality - Google Registration for Freelancer

METHOD	ENDPOINT	DESCRIPTION	REQUEST DATA	RESPONSE
POST	registerFreelancer.php	Handles the registration of a freelancer via Google Sign-In. It accepts a tokenId from the Google authentication response, verifies it with Google, and checks if the email already exists in the database. If the email is new, it registers the user with basic info extracted from the token or submitted alongside it.	<b>JSON:</b> {"tokenId": "ya29.A0AV9yItExampleGoogleToken"}	<b>Success:</b> Signed in with Google successfully.  <b>Email Exists:</b> Email already registered. Redirecting to login.

### Code:

```
// Check if the email already exists
$checkEmailSql = "SELECT * FROM _user WHERE email = '$email'";
$result = $con->query($checkEmailSql);

if ($result->num_rows > 0) {
  echo "<script>alert('Email already exists. Please use a different email.');" . "</script>";
} else {
  // Insert the data if email doesn't exist
  $sql = "INSERT INTO _user (first_name, last_name, email, password, country, role)
    VALUES ('$firstName', '$lastName', '$email', '$password', '$country', '$role')";

  if ($con->query($sql)) {
    echo "<script>alert('Registration successful!'); window.location.href='login.php';</script>";
  } else {
    echo "Error: " . $con->error;
  }
}
```

- \$checkEmailSql = "SELECT \* FROM \_user WHERE email = '\$email'";
  - This line creates an SQL query to check if an email address already exists in the \_user table. The \$email variable presumably comes from user input.
- \$result = \$con->query(\$checkEmailSql);
  - This executes the query.
- if (\$result->num\_rows > 0) { ... }
  - This checks if the query returned any rows (meaning the email already exists). If so, a JavaScript alert is displayed.
- } else { ... }
  - This block executes if the email is unique.
- \$sql = "INSERT INTO \_user (first\_name, last\_name, email, password, country, role) VALUES ('\$firstName', '\$lastName', '\$email', '\$password', '\$country', '\$role')";
  - This creates an SQL query to insert new user data into the \_user table. The values are taken from variables (\$firstName, \$lastName, etc.) presumably obtained from a user registration form.
- if (\$con->query(\$sql)) { ... }
  - This executes the insertion query. If successful, a JavaScript alert and redirect to login.php occurs.
- } else { echo "Error: " . \$con->error; }
  - If the insertion fails, an error message is displayed.

## REST API Functionality - Manual Registration for Freelancer

METHOD	ENDPOINT	DESCRIPTION	REQUEST DATA	RESPONSE
POST	registerFreelancer.php	Handles the manual registration of a freelancer by accepting user-submitted form data. It receives the first name, last name, email, password, country, and role. The backend checks whether the email already exists in the database. If not, it inserts the new user data into the database and returns a success response.	<p><b>Manual Registration</b></p> <pre>{"firstName": "John", "lastName": "Doe", "email": "john@example.com", "password": "hashed_password", "country": "Philippines"}</pre>	<p><b>Success:</b> Registration successful</p> <p><b>Email Exists:</b> Email already exists. Please use a different email.</p>

### Code:

```

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve POST data
    $firstName = isset($_POST['first_name']) ? mysqli_real_escape_string($con, $_POST['first_name']) : '';
    $lastName = isset($_POST['last_name']) ? mysqli_real_escape_string($con, $_POST['last_name']) : '';
    $email = isset($_POST['email']) ? mysqli_real_escape_string($con, $_POST['email']) : '';
    $password = isset($_POST['password']) ? password_hash($_POST['password'], PASSWORD_DEFAULT) : '';
    $country = isset($_POST['country']) ? mysqli_real_escape_string($con, $_POST['country']) : '';

    // Check if the email already exists
    $checkEmailSql = "SELECT * FROM user WHERE email = '$email'";
    $result = $con->query($checkEmailSql);

    if ($result->num_rows > 0) {
        echo "<script>alert('Email already exists. Please use a different email.');" . "</script>";
    } else {
        // Insert the data if email doesn't exist
        $sql = "INSERT INTO user (first_name, last_name, email, password, country, role)
                VALUES ('$firstName', '$lastName', '$email', '$password', '$country', '$role')";

        if ($con->query($sql)) {
            echo "<script>alert('Registration successful!'); window.location.href='login.php';</script>";
        } else {
            echo "Error: " . $con->error;
        }
    }
}

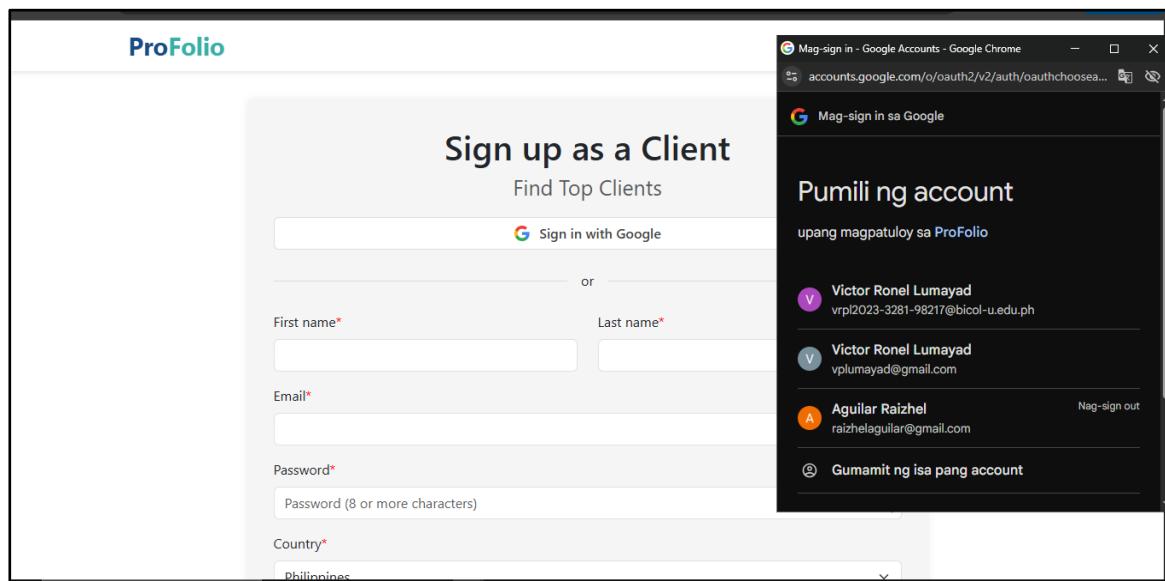
```

- (if(\$\_SERVER["REQUEST\_METHOD"] == "POST"))
  - The code begins by checking if the HTTP request method is . This ensures the code only executes when a form is submitted.
- It then retrieves user data from the \$\_POST superglobal array, which contains data submitted via a form. The data includes first\_name, last\_name, email, password, and country.
- The isset() function checks if each element exists in the \$\_POST array. If an element is missing, its corresponding variable is assigned an empty string. This is a basic form of input validation.
- mysqli\_real\_escape\_string(\$con, ...)
  - this is applied to each field except the password. This function escapes special characters in strings to prevent some SQL injection attacks. However, this method is not sufficient to prevent all SQL injection attacks and is considered outdated and insecure compared to prepared statements. It's crucial to understand that this only provides partial protection and is not a replacement for proper parameterized queries.
- \$checkEmailSql = "SELECT \* FROM user WHERE email = '\$email'";
  - This line constructs an SQL query to check if a user with the given email already exists in the user table. The \$email variable, which comes directly from user input, is directly concatenated into the SQL string. This is the primary source of the SQL injection vulnerability.

- `$result = $con->query($checkEmailSql);`  
This executes the SQL query.
- `if ($result->num_rows > 0) { ... }`
  - This conditional statement checks the number of rows returned by the query. If the number of rows is greater than 0, it means an email with that address already exists in the database. In this case, a JavaScript alert is displayed to inform the user.
- `$sql = "INSERT INTO user (first_name, last_name, email, password, country, role) VALUES ('$firstName', '$lastName', '$email', '$password', '$country', '$role')";`
  - This line constructs the SQL INSERT query to add the new user to the database. The values for the fields are taken directly from the user input variables. This is another point where SQL injection is possible because user-supplied data is directly inserted into the query.
- `password_hash($POST['password'], PASSWORD_DEFAULT)`
  - This line uses the password\_hash() function to securely hash the user's password before storing it in the database. This is a crucial security step that protects against plain-text password storage.
- The INSERT query is then executed using `$con->query($sql)`.
- The conditional statement checks for successful execution: If the query is successful, a JavaScript alert is displayed, and the user is redirected to the login page (login.php). If there's an error, the database error message is displayed.

## registerClient.php

The Google Sign-In feature is integrated into the Client Registration Form to simplify and speed up the account creation process, while also enhancing security. By using this feature, clients can bypass manual data entry for fields like name and email. When a client clicks the “Sign in with Google” button at the top of the form, it triggers Google’s OAuth 2.0 login flow. A popup window appears, prompting the user to choose a Google account, “Pumili ng account upang magpatuloy sa ProFolio”. Once the user selects an account, Google authenticates their identity and requests permission to share basic profile information such as name and email. After successful authentication, this data is securely sent to the backend script (`registerClient.php`). From there, the system can auto-fill the registration form, register the user instantly, or log them in automatically if their account already exists. The integration ensures a faster, smoother user experience with minimal friction during sign-up.



## REST API Functionality - Google Sign-In for Client

METHOD	ENDPOINT	DESCRIPTION	REQUEST DATA	RESPONSE
POST	registerClient.php	Handles the registration of a client using Google Sign-In. The frontend sends a tokenId from Google, which is then verified by the backend to authenticate the user. Upon successful verification, Google provides basic profile information such as first name, last name, and email. The backend checks whether the email is already registered. If it exists, the user is redirected to log in.	<b>JSON:</b> {"tokenId": "ya29.A0AV9yItExampleGoogleToken"}	<p><b>Success:</b> Signed in with Google successfully.</p> <p><b>Email Exists:</b> Email is already registered. Redirecting to login.</p>

### Code:

```
// Retrieve the role from the URL parameter
$role = isset($_GET['role']) ? $_GET['role'] : 'client'; // Default to 'client' if not provided

// Handle Google Sign-Up

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['google_signup'])) {
    $firstName = isset($_POST['first_name']) ? mysqli_real_escape_string($con, $_POST['first_name']) : '';
    $lastName = isset($_POST['last_name']) ? mysqli_real_escape_string($con, $_POST['last_name']) : '';
    $email = isset($_POST['email']) ? mysqli_real_escape_string($con, $_POST['email']) : '';
    $role = 'client';

    if (!empty($email)) {
        $checkUserSql = "SELECT * FROM _user WHERE email = '$email'";
        $result = $con->query($checkUserSql);

        if ($result->num_rows === 0) {
            // No user yet - proceed with Google sign-up
            $sql = "INSERT INTO _user (first_name, last_name, email, password, country, role
                   VALUES ('$firstName', '$lastName', '$email', '', '', '$role')";
            $con->query($sql);
            echo json_encode(['status' => 'success', 'message' => 'Signed in with Google!']);
            exit;
        } else {
            // Email already exists - do NOT allow Google sign-up again
            echo json_encode([
                'status' => 'redirect',
                'message' => 'Email already registered. Redirecting to login...',
                'redirect_url' => 'login.php'
            ]);
            exit;
        }
    }
}

```

- <?php
  - This line starts the PHP code block.
- if (\$\_SERVER["REQUEST\_METHOD"] == "POST" && isset(\$\_POST['google\_signup'])) {
 

This line checks two conditions:

  - \$\_SERVER["REQUEST\_METHOD"] == "POST"
    - Verifies that the HTTP request method is POST (meaning data was submitted from a form).
  - isset(\$\_POST['google\_signup'])
    - Checks if the google\_signup key exists in the \$\_POST array. This likely indicates that the user initiated Google Sign-In. The entire block within the if statement only executes if both conditions are true.
- These lines 4-8 retrieve values from the \$\_POST array, which contains data submitted from the Google Sign-In form. isset() checks if each key exists; if not, an empty string is used as a default. No sanitization is needed here because prepared statements handle data security.
- \$role = isset(\$\_POST['role']) ? \$\_POST['role'] : 'client';
  - This line retrieves the user's role from the \$\_POST array. If the role key is not set, it defaults to 'client'.
- \$stmt = \$con->prepare("SELECT 1 FROM \_user WHERE email = ?");
  - This line prepares an SQL statement to check for an existing email address. \$con is assumed to be a database connection object established elsewhere. SELECT 1 is an optimized query; it only retrieves a single column (1) if a match is found, making it more efficient than SELECT \*. The ? is a placeholder for the email address.

- `$stmt->bind_param("s", $email);`
  - This line binds the \$email variable to the placeholder in the prepared statement. "s" specifies that the variable is a string. This is crucial for security; it prevents SQL injection.
- `$stmt->execute();`
  - This line executes the prepared statement.
- `$result = $stmt->get_result();`
  - This line gets the result set from the executed query.
- `if ($result->num_rows > 0) { ... }`
  - This line checks if the query returned any rows (meaning the email already exists).
- `echo json_encode(['status' => 'redirect', ...]);`
  - If the email already exists, this line sends a JSON response indicating a redirect to the login page.
- `} else { ... }`
  - This block executes if the email is unique (not found in the database).
- `$stmt = $con->prepare("INSERT INTO _user ... VALUES (?, ?, ?, ?, ?, ?)");`
  - This line prepares an SQL INSERT statement to add a new user to the database. The ? placeholders represent the values for the respective columns.
- `$stmt->bind_param("ssssss", $firstName, $lastName, $email, "", $country, $role);`
  - This line binds the user data to the placeholders in the prepared statement. "ssssss" specifies that all variables are strings. The password is set to an empty string ("") because Google handles authentication; the application doesn't need to store a password in this scenario.
- `if ($stmt->execute()) { ... }`
  - This line executes the prepared INSERT statement.
- `echo json_encode(['status' => 'success', ...]);`
  - If the insertion was successful, this line sends a JSON response indicating success.
- `} else { echo json_encode(['status' => 'error', 'message' => $stmt->error]); }`
  - If the insertion failed, this line sends a JSON response with an error message from the database.
- `$stmt->close();`
  - This line closes the prepared statement to release database resources.
- `$stmt->close();`
  - This line closes the prepared statement used for the email check.
- `}`
  - This closing brace ends the if statement.
- `?>`
  - This line closes the PHP code block.

## REST API Functionality - Manual Registration for Client

METHOD	ENDPOINT	DESCRIPTION	REQUEST DATA	RESPONSE
POST	registerClient.php	Handles the manual registration of a client. This endpoint receives form data directly from the user. It checks if the provided email already exists in the database. If the email is unique, the client is registered with the provided information.	<p><b>Manual Registration</b></p> <pre>{ "firstName": "John", "lastName": "Doe", "email": "john@example.com", "password": "hashed_password", "country": "Philippines"}</pre>	<p><b>Success:</b> Registration successful</p> <p><b>Email Exists:</b> Email is already exists. Please use a different email.</p>

### Code:

```

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve POST data
    $firstName = isset($_POST['first_name']) ? mysqli_real_escape_string($con, $_POST['first_name']) : '';
    $lastName = isset($_POST['last_name']) ? mysqli_real_escape_string($con, $_POST['last_name']) : '';
    $email = isset($_POST['email']) ? mysqli_real_escape_string($con, $_POST['email']) : '';
    $password = isset($_POST['password']) ? password_hash($_POST['password'], PASSWORD_DEFAULT) : '';
    $country = isset($_POST['country']) ? mysqli_real_escape_string($con, $_POST['country']) : '';

    // Check if the email already exists
    $checkEmailSql = "SELECT * FROM _user WHERE email = '$email'";
    $result = $con->query($checkEmailSql);

    if ($result->num_rows > 0) {
        echo "<script>alert('Email already exists. Please use a different email.');" . "</script>";
    } else {
        // Insert the data if email doesn't exist
        $sql = "INSERT INTO _user (first_name, last_name, email, password, country, role)
                VALUES ('$firstName', '$lastName', '$email', '$password', '$country', '$role')";

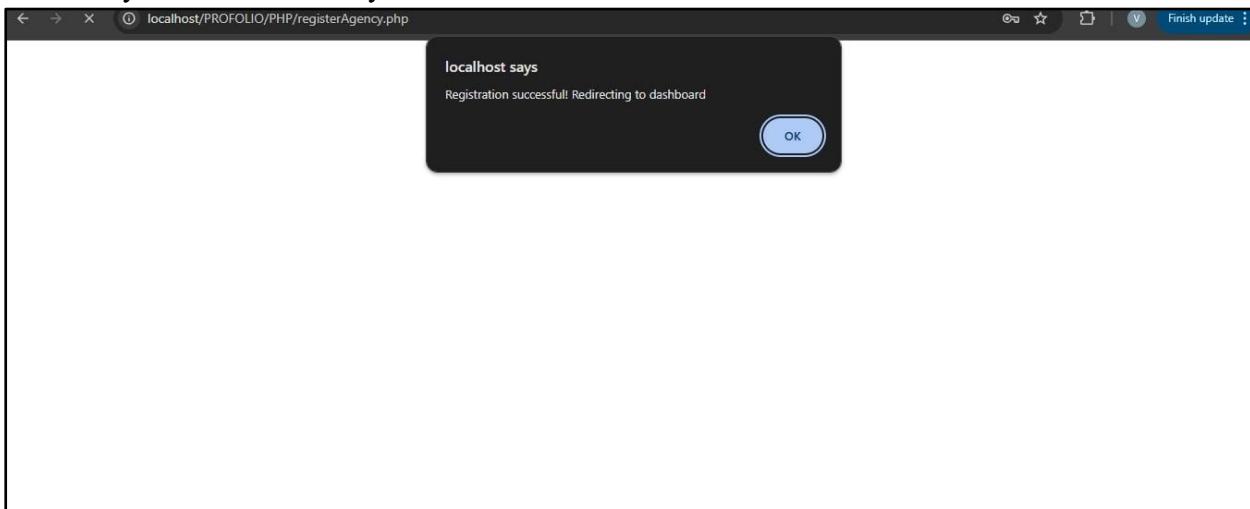
        if ($con->query($sql)) {
            echo "<script>alert('Registration successful!'); window.location.href='login.php';</script>";
        } else {
            echo "Error: " . $con->error;
        }
    }
}
?>
```

- if(\$\_SERVER["REQUEST\_METHOD"]=="POST") { ... }
  - This condition ensures the code executes only if the request is a POST request. POST requests are typically used to submit data from a form to a server.
- \$firstName = isset(\$\_POST['first\_name']) ? mysqli\_real\_escape\_string(\$con, \$\_POST['first\_name']) : '';
  - This line retrieves the first\_name from the POST data. isset() checks if the variable is set. mysqli\_real\_escape\_string(\$con, \$\_POST['first\_name']) sanitizes the input to prevent SQL injection vulnerabilities. If first\_name is not set, it defaults to an empty string. Similar lines handle lastName, email, password, and country.
- \$password = isset(\$\_POST['password']) ? password\_hash(\$\_POST['password'], PASSWORD\_DEFAULT) : '';
  - Crucially, the password is not stored in plain text. password\_hash() securely hashes the password using the PASSWORD\_DEFAULT algorithm (recommended for security).
- \$checkEmailSql = "SELECT \* FROM \_user WHERE email = '\$email'";
  - This SQL query checks if an email address already exists in the \_user table.
- \$result = \$con->query(\$checkEmailSql);
  - The query is executed, and the result is stored in \$result.
- if (\$result->num\_rows > 0) { ... }
  - If the query returns one or more rows (meaning the email exists), an alert message is displayed informing the user.

- \$sql = "INSERT INTO \_user (first\_name, last\_name, email, password, country, role) VALUES ('\$firstName', '\$lastName', '\$email', '\$password', '\$country', '\$role')";  
 - If the email is unique, this SQL query inserts the user's data into the \_user table. Note that \$role is used, suggesting different user roles might exist in the system. The values are inserted using placeholders, which are vulnerable to SQL injection if not properly sanitized (as done earlier).
- if (\$con->query(\$sql)) { ... }  
 - If the insertion is successful, a success message is displayed, and the user is redirected to login.php.
- else { echo "Error: " . \$con->error; }  
 - If an error occurs during insertion, an error message is displayed.

## registerAgency.php

Manual registration requires agencies to fill out a form with details such as agency name, logo, type, team size, description, services, website (optional), and admin account information (name, job title, email, password, country, city). The server-side code then validates, sanitizes, and securely hashes the password before storing the data in the database. Google Sign-In streamlines the process by allowing agencies to click the "Sign in with Google" button, authenticate with their Google account, and grant permission to share their name and email. The server retrieves this data, automatically populates the form fields, and either creates a new agency account or logs the user in directly if the email already exists.



## REST API Functionality - Manual Registration for Agency

METHOD	ENDPOINT	DESCRIPTION	REQUEST DATA	RESPONSE
POST	registerAgency.php	Handles the manual registration of an agency. This endpoint receives form data directly from the user. It checks if the provided email already exists in the database. If the email is unique, the client is registered with the provided information; otherwise, an appropriate error message is returned. The endpoint also validates the input data to ensure data integrity and prevent vulnerabilities. Upon successful registration, a confirmation message or response is sent to the client.	<b>Manual Registration:</b> {"agencyName": "Example Agency", "agencyLogo": "data:image/png;base64,VBORw0KGgoAAAANSUhEUgAAAAEAAQCAQAAAC1HawCAAACOIEQVR42mNkYAAAAAYAAjCB0C8AAAASUVORK5CYII=", // Base64 encoded logo (optional)"agencyType": "Design Agency", "teamSize": "2-5", "agencyDescription": "A creative design agency specializing in branding and web development.", "services": "Web Design, Web Development, UI/UX Design", // Comma-separated list"agencyWebsite": "https://exampleagency.com", "firstName": "John", "lastName": "Doe", "jobTitle": "Creative Director", "email": "john.doe@exampleagency.com", "password": "securepassword", "country": "Philippines", "city": "Manila"}  <b>Success:</b> Registration successful! Redirecting to dashboard	<b>Success:</b> Registration successful! Redirecting to dashboard

## Code:

```
PHP > registerAgency.php
 8  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
13    if (isset($_FILES['agencyLogo']) && $_FILES['agencyLogo']['error'] == 0) {
40      }
41    } else {
42      $agencyLogo = ''; // If no file uploaded
43    }
44
45    $agencyType = $_POST['agencyType'];
46    $teamSize = $_POST['teamSize'];
47    $agencyDescription = $_POST['agencyDescription'];
48
49    // Handle services selection
50    $services = isset($_POST['services']) && is_array($_POST['services']) ? implode(',', $_POST['services']);
51
52    $agencyWebsite = $_POST['agencyWebsite'];
53    $firstName = $_POST['firstName'];
54    $lastName = $_POST['lastName'];
55    $jobTitle = $_POST['jobTitle'];
56    $email = $_POST['email'];
57    $password = password_hash($_POST['password'], PASSWORD_DEFAULT); // Hash password
58    $country = $_POST['country'];
59    $city = $_POST['city'];
60
61    // Check if the email already exists
62    $emailCheck = $con->prepare("SELECT email FROM _user WHERE email = ?");
63    $emailCheck->bind_param("s", $email);
64    $emailCheck->execute();
65    $emailCheck->store_result();
66
67    if ($emailCheck->num_rows > 0) {
```

```
<?php
include_once("../connection/connection.php");
$con = connection();

$role = isset($_GET['role']) ? $_GET['role'] : 'agency'; // Default to 'Agency' if not provided

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Get form data
    $agencyName = $_POST['agencyName'];

    // Handle file upload for logo
    if (isset($_FILES['agencyLogo']) && $_FILES['agencyLogo']['error'] == 0) {
        $logoTmp = $_FILES['agencyLogo']['tmp_name'];
        $logoName = $_FILES['agencyLogo']['name'];
        $logoExtension = strtolower(pathinfo($logoName, PATHINFO_EXTENSION));

        // Allowed file types
        $allowedExtensions = ['jpg', 'jpeg', 'png', 'gif'];

        // Check if the file is of a valid type
        if (in_array($logoExtension, $allowedExtensions)) {
            $logoDestination = '../uploads/' . $logoName;

            // Check if file already exists
            if (file_exists($logoDestination)) {
                echo "Error: File already exists.";
                exit();
            } else {
                if (move_uploaded_file($logoTmp, $logoDestination)) {
```

```
    if ($stmt->execute()) {
        echo "<script>alert('Registration successful! Redirecting to dashboard'); window.location.href='login.php';</script>";
    } else {
        echo "Error: " . $stmt->error;
    }
    $stmt->close();
    $con->close();
}
?>
```