

Capstone Project II

# Contractor Connect

System Implementation 1

The Four Group

**Presented by:**

**Adel Alhajhussain** – Backend Developer

**Mehdi Jazi** – Database Management

**Shifa Karnelia** – UI Developer

**Sana Karnelia** – Frontend Developer

**Eric Laudrum** – Quality Assurance

**Instructor:** Prof. Laily Ajellu

## Project Description

**ContractorConnect** is a web-based platform designed to connect **homeowners** with **contractors** for residential projects through a structured and transparent system.

### Project Idea:

The system centralizes how projects are posted, discovered, and managed, replacing informal communication and fragmented hiring methods.

### What the Project Achieves:

- Enables homeowners to easily find and evaluate contractors
- Allows contractors to discover and bid on relevant projects
- Provides a role-based dashboard experience (Admin / Homeowner / Contractor)
- Improves transparency, organization, and workflow efficiency
- Creates a scalable digital solution for project management and service matching

# Technology Used

## Backend / Server-Side

- PHP
- CodeIgniter 4 (CI4)
- MVC Architecture

## Database

- MySQL

## Frontend

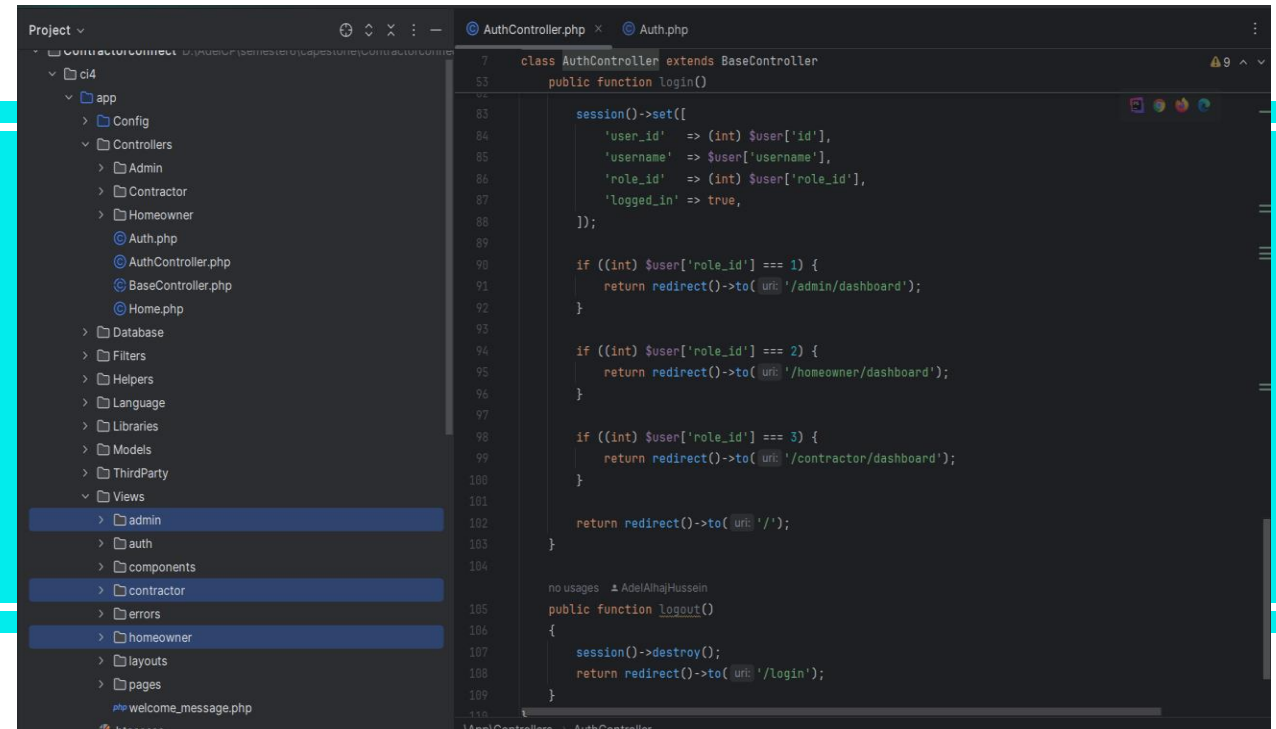
- HTML
- CSS
- Bootstrap

## Version Control

- Git
- GitHub

# Role + CI4 Backend Architecture

- ❖ Backend implemented using CodeIgniter 4 (CI4)
- ❖ Controllers organized by module: Admin / Contractor / Homeowner
- ❖ Central AuthController for authentication (register/login)
- ❖ Clean structure supports maintainability and easier future features



The screenshot displays a code editor with a project structure on the left and the code for AuthController.php on the right. The project structure shows a 'ci4' directory containing 'app', 'Config', 'Controllers', 'Database', 'Filters', 'Helpers', 'Language', 'Libraries', 'Models', 'ThirdParty', and 'Views'. The 'Views' directory is expanded to show 'admin', 'auth', 'contractor', 'homeowner', 'errors', 'layouts', and 'pages'. The 'AuthController.php' file is selected, and its code is visible. The code defines a class 'AuthController' that extends 'BaseController'. It includes a 'login()' method that sets session variables for 'user\_id', 'username', 'role\_id', and 'logged\_in'. It then uses conditional logic to redirect users to their respective dashboards based on their role\_id (1 for admin, 2 for homeowner, 3 for contractor). A 'logout()' method is also shown, which destroys the session and redirects to the login page.

```
class AuthController extends BaseController
{
    public function login()
    {
        session()->set([
            'user_id' => (int) $user['id'],
            'username' => $user['username'],
            'role_id' => (int) $user['role_id'],
            'logged_in' => true,
        ]);

        if ((int) $user['role_id'] === 1) {
            return redirect()->to('uri'.'/admin/dashboard');
        }

        if ((int) $user['role_id'] === 2) {
            return redirect()->to('uri'.'/homeowner/dashboard');
        }

        if ((int) $user['role_id'] === 3) {
            return redirect()->to('uri'.'/contractor/dashboard');
        }

        return redirect()->to('uri'.'/');
    }

    public function logout()
    {
        session()->destroy();
        return redirect()->to('uri'.'/login');
    }
}
```

## Routing Setup

```
php Routes.php x
39 $routes->get( from: 'admin/contractors', to: 'Admin\ContractorsController::index', ['filter' => 'auth']);
40 $routes->get( from: 'admin/contractors/toggle/(:num)', to: 'Admin\ContractorsController::toggle/$1', ['filter' => 'auth']);
41 $routes->get( from: 'admin/contractors/approve/(:num)', to: 'Admin\ContractorsController::approve/$1', ['filter' => 'auth']);
42 $routes->get( from: 'admin/contractors/reject/(:num)', to: 'Admin\ContractorsController::reject/$1', ['filter' => 'auth']);
43
44
45 $routes->get( from: 'admin/homeowners', to: 'Admin\HomeownersController::index');
46 $routes->get( from: 'admin/homeowners/toggle/(:num)', to: 'Admin\HomeownersController::toggle/$1', ['filter' => 'auth']);
47
48
49 $routes->get( from: 'admin/projects', to: 'Admin\ProjectsController::index');
50 $routes->get( from: 'admin/projects', to: 'Admin\ProjectsController::index', ['filter' => 'auth']);
51 $routes->get( from: 'admin/projects/view/(:num)', to: 'Admin\ProjectsController::view/$1', ['filter' => 'auth']);
52 $routes->get( from: 'admin/projects/cancel/(:num)', to: 'Admin\ProjectsController::cancel/$1', ['filter' => 'auth']);
53 $routes->get( from: 'admin/projects/close-bidding/(:num)', to: 'Admin\ProjectsController::closeBidding/$1', ['filter' => 'auth']);
54
55
56 $routes->get( from: 'admin/bids', to: 'Admin\BidsController::index');
57 $routes->get( from: 'admin/bids', to: 'Admin\BidsController::index', ['filter' => 'auth']);
58 $routes->get( from: 'admin/bids/view/(:num)', to: 'Admin\BidsController::view/$1', ['filter' => 'auth']);
59 $routes->get( from: 'admin/bids/withdraw/(:num)', to: 'Admin\BidsController::withdraw/$1', ['filter' => 'auth']);
60
61
62
63 $routes->get( from: 'admin/ratings', to: 'Admin\RatingsController::index');
64 $routes->get( from: 'admin/ratings', to: 'Admin\RatingsController::index', ['filter' => 'auth']);
65 $routes->get( from: 'admin/ratings/view/(:num)', to: 'Admin\RatingsController::view/$1', ['filter' => 'auth']);
66 $routes->get( from: 'admin/ratings/remove/(:num)', to: 'Admin\RatingsController::remove/$1', ['filter' => 'auth']);
67 $routes->get( from: 'admin/ratings/suspicious', to: 'Admin\RatingsController::suspicious', ['filter' => 'auth']);
68
69
```

- All routes are defined in app/Config/Routes.php
- Routes map URLs → controller methods (login, logout, admin pages)
- Dynamic parameters supported (e.g., (:num) for IDs)
- Auth filter protects admin routes from unauthorized access

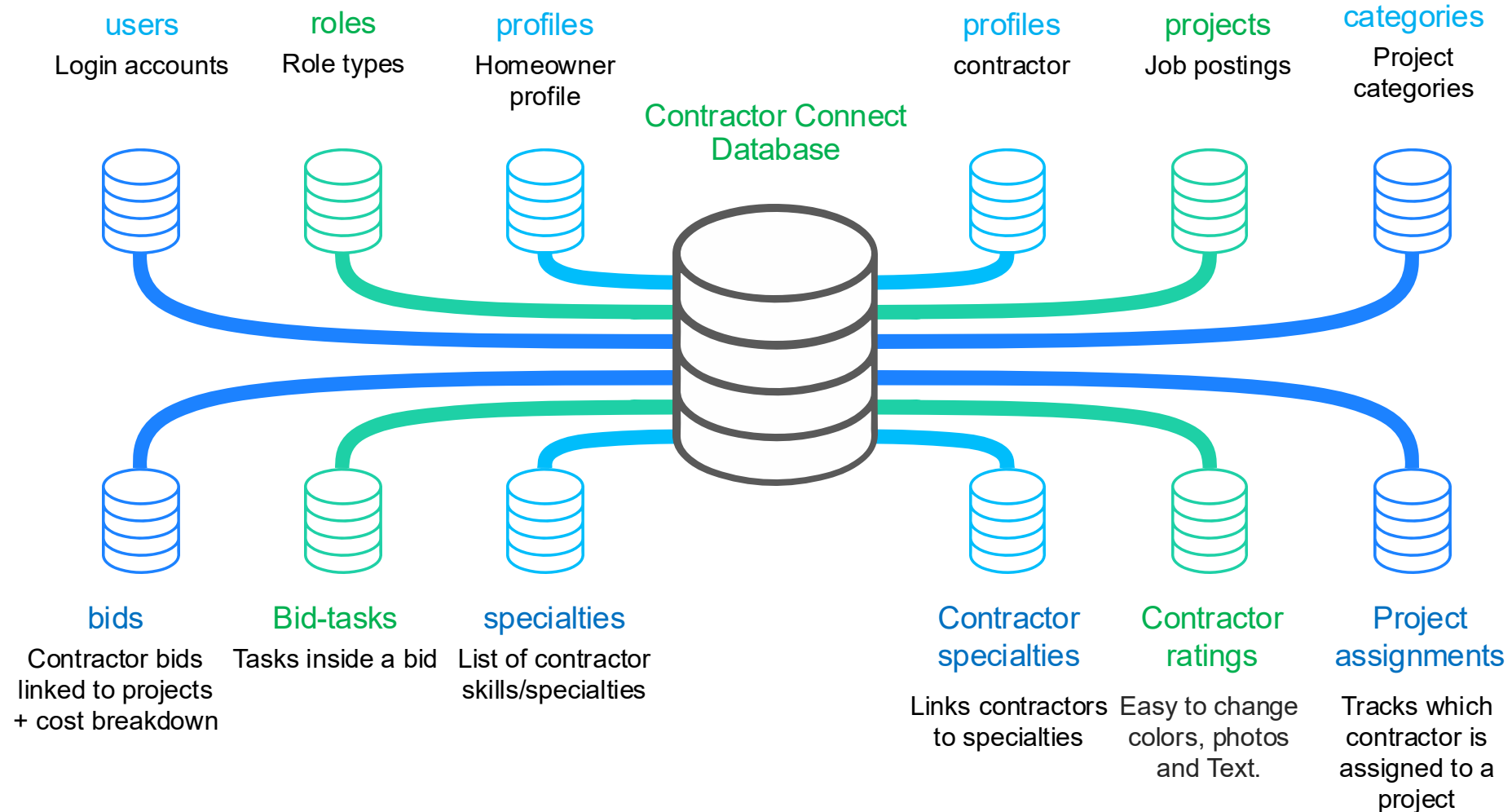
```
1 <?php
2 namespace App\Controllers\Admin;
3 use App\Controllers\BaseController;
4 use App\Models\CategoryModel;
5 class CategoriesController extends BaseController
6 {
7     public function index()
8     {
9         $model = new CategoryModel();
10        $q = trim($this->request->getGet('q') ?? '');
11        $visibility = $this->request->getGet('visibility');
12        $builder = $model;
13        if ($q != '') {
14            $builder = $builder->like('field', 'name', $q);
15        }
16        if ($visibility != null && $visibility != '') {
17            $builder = $builder->where('key', 'is_visible', (int)$visibility);
18        }
19        $data = [
20            'categories' => $builder->orderBy('orderBy', 'id', 'direction' => 'DESC')->findAll(),
21            'q' => $q,
22            'visibility' => $visibility,
23        ];
24        return view('admin/categories/index', $data);
25    }
26    public function create()
27    {
28        return view('admin/categories/create');
29    }
30    public function store()
31    {
32        $model = new CategoryModel();
33
34        $model->insert([
35            'name' => $this->request->getPost('name'),
36            'is_visible' => 1
37        ]);
38        return redirect()->to(site_url('relativePath: 'admin/categories'));
39    }
40    public function edit($id)
41    {
42        $model = new CategoryModel();
43        $data['category'] = $model->find((int)$id);
44    }
45    public function update($id)
46    {
47        $model = new CategoryModel();
48        $model->update((int)$id, [
49            'name' => $this->request->getPost('name')
50        ]);
51        return redirect()->to(site_url('relativePath: 'admin/categories'));
52    }
53    public function delete($id)
54    {
55        $model = new CategoryModel();
56        $model->delete((int)$id);
57        return redirect()->to(site_url('relativePath: 'admin/categories'));
58    }
59    public function toggle($id)
60    {
61        $model = new CategoryModel();
62        $category = $model->find((int)$id);
63        if ($category) {
64            $model->update((int)$id, [
65                'is_visible' => $category['is_visible'] ? 0 : 1
66            ]);
67            audit_log(
68                action: 'category_visibility_toggled',
69                entityType: 'category',
70                (int)$id,
71                details: 'Admin toggled category visibility'
72            );
73        }
74        return redirect()->to(site_url('relativePath: 'admin/categories'));
75    }
76 }
```

```
5 class CategoriesController extends BaseController
6 public function store()
7 {
8 }
9 public function edit($id)
10 {
11     $model = new CategoryModel();
12     $data['category'] = $model->find((int)$id);
13     return view('admin/categories/edit', $data);
14 }
15 public function update($id)
16 {
17     $model = new CategoryModel();
18     $model->update((int)$id, [
19         'name' => $this->request->getPost('name')
20     ]);
21     return redirect()->to(site_url('relativePath: 'admin/categories'));
22 }
23 public function delete($id)
24 {
25     $model = new CategoryModel();
26     $model->delete((int)$id);
27     return redirect()->to(site_url('relativePath: 'admin/categories'));
28 }
29 public function toggle($id)
30 {
31     $model = new CategoryModel();
32     $category = $model->find((int)$id);
33     if ($category) {
34         $model->update((int)$id, [
35             'is_visible' => $category['is_visible'] ? 0 : 1
36         ]);
37         audit_log(
38             action: 'category_visibility_toggled',
39             entityType: 'category',
40             (int)$id,
41             details: 'Admin toggled category visibility'
42         );
43     }
44     return redirect()->to(site_url('relativePath: 'admin/categories'));
45 }
```

## CRUD Implementation

- CategoriesController demonstrates full CRUD workflow
- Create: insert() adds new category records
- Read: findAll() loads category list (with filters/search)
- Update: update(id, data) edits category details or visibility
- Delete: delete(id) removes records when needed

# Database Schema Overview





phpMyAdmin - Server: localhost:3306 - Database: cclogin\_ccdb - Table: bids

Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)

SELECT \* FROM `bids`

	id	project_id	contractor_id	details	bid amount	est total minutes	materials_cost	labour_cost	hst_cost	total_cost	status	created_at	updated_at
<input type="checkbox"/>	1	1	3	Test bid for admin bid management module	1500.00	300	400.00	900.00	195.00	1495.00	submitted	2026-02-12 18:26:25	2026-02-12 18:26:25
<input type="checkbox"/>	2	2	3	Available this weekend.	150.00	120	20.00	110.00	16.90	146.90	submitted	2026-02-13 13:44:15	2026-02-13 13:44:15
<input type="checkbox"/>	3	5	6		250.00	0	0.00	0.00	0.00	0.00	NULL	NULL	NULL
<input type="checkbox"/>	4	2	6	I will do the job quick and cheap	100.00	0	0.00	0.00	0.00	100.00	submitted	NULL	NULL

# Main Tables & Sample Records (Users • Projects • Bids)

- Core tables used in the prototype: users, projects, bids

phpMyAdmin - Server: localhost:3306 - Database: cclogin\_ccdb - Table: projects

Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)

SELECT \* FROM `projects`

	id	home_owner_id	category_id	title	description	address	contact_phone	start_date	end_date	deadline_date	budget_min	budget_max	status	completed_at	created_at	updated_at	deleted_at
<input type="checkbox"/>	1	4	1	Test Project	Test description	NULL	NULL	NULL	NULL	NULL	100.00	500.00	in_progress	NULL	2026-02-11 20:42:10	2026-02-12 01:48:14	NULL
<input type="checkbox"/>	2	5	1	Lawn Care	Front yard needs mowing and cleanup.	123 Test Street, Toronto, ON 1234567890	2026-02-13	2026-02-23	2026-02-16	80.00	200.00	bidding_open	NULL	2026-02-13 13:41:58	2026-02-13 13:41:58	NULL	NULL
<input type="checkbox"/>	5	5	2	Roofing test	This is a test project	NULL	NULL	NULL	NULL	NULL	500.00	1500.00	bidding_open	NULL	2026-02-14 00:05:11	2026-02-14 00:05:11	NULL

- users stores login + roles (admin / homeowner / contractor)

- projects created by homeowner (home\_owner\_id → users.id)

- bids submitted by contractor (contractor\_id → users.id, project\_id → projects.id)

phpMyAdmin - Server: localhost:3306 - Database: cclogin\_ccdb - Table: users

Showing rows 0 - 5 (6 total, Query took 0.0003 seconds.)

SELECT \* FROM `users`

	id	username	role_id	first_name	last_name	email	phone	password_hash	is_active	created_at	updated_at	deleted_at
<input type="checkbox"/>	1	admin	1	System	Admin	admin@contractorconnect.cc	0000000000	\$2y\$10\$PqZnDFcR.OVqyew4.Hg7OoIwnRe97or7l.MU9j7F4...	1	2026-02-04 19:47:23	2026-02-04 19:47:23	NULL
<input type="checkbox"/>	3	contractor1	3	Test	Contractor	contractor1@contractorconnect.cc	0000000001	\$2y\$10\$PqZnDFcR.OVqyew4.Hg7OoIwnRe97or7l.MU9j7F4...	1	2026-02-06 14:27:21	2026-02-14 23:01:46	NULL
<input type="checkbox"/>	4	homeowner1	2	Test	Homeowner	homeowner1@contractorconnect.cc	0000000002	\$2y\$10\$PqZnDFcR.OVqyew4.Hg7OoIwnRe97or7l.MU9j7F4...	1	2026-02-11 19:37:36	2026-02-14 22:37:31	NULL
<input type="checkbox"/>	5	homeowner_test	2	Test	Homeowner	homeowner_test@mail.com	1234567890	\$2y\$10\$Nv6iR.akhwFjCERkXde132SHexqLUkYRSiIDIXVR...	1	2026-02-13 12:42:47	NULL	NULL
<input type="checkbox"/>	6	contractor_test	3	Test	Contractor	contractor@test.com	NULL	\$2y\$10\$Nv6iR.akhwFjCERkXde132SHexqLUkYRSiIDIXVR...	1	NULL	NULL	NULL
<input type="checkbox"/>	7	AdelHomeowner	2				NULL	\$2y\$10\$gokQFFKAElAwF3Qj2UJURLXOe7LmpNnwGWGDgY...	1	2026-02-16 23:48:26	2026-02-16 23:48:26	NULL

- Sample data proves full flow: user → post project → submit bid

- Linked with primary keys + foreign keys to keep data consistent

The image shows a screenshot of an IDE with a project structure on the left and a code editor on the right. The project structure is as follows:

- Project
- contractor\_connect-main [Contractorconnect]
- ci4
  - app
  - Database
  - public
  - system
  - tests
  - vendor
  - writable
- .env
- composer.json
- composer.lock
- LICENSE
- phpunit.xml.dist
- preload.php
- README.md
- spark
- vendor.zip
- public\_html
- .gitignore
- README.md
- External Libraries
- Scratches and Consoles

The code editor shows the contents of the .env file:

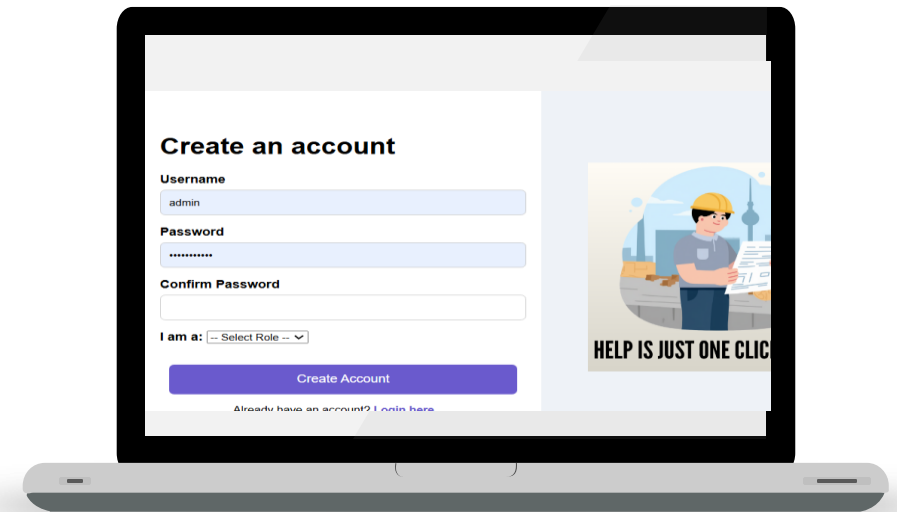
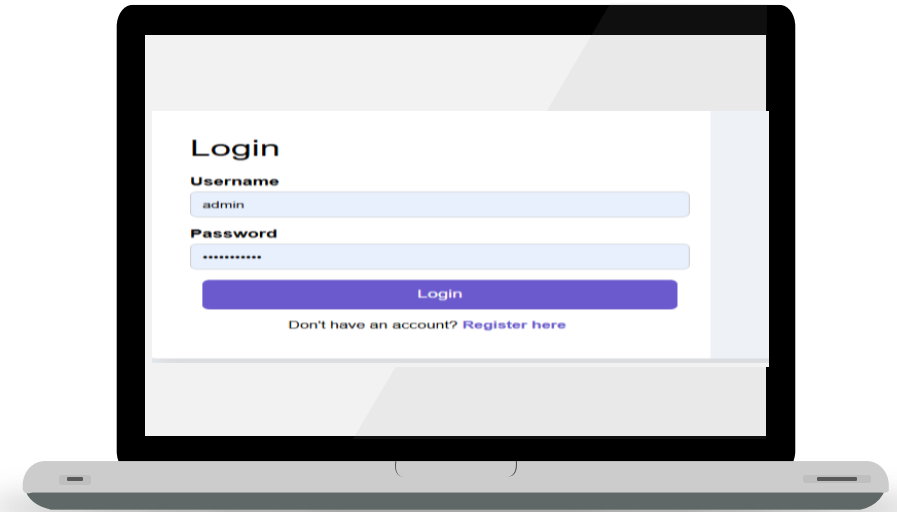
```
1 CI_ENVIRONMENT = development
2 app.baseUrl = 'http://localhost:8081/'
3 database.default.hostname = localhost
4 database.default.database = contractor_connect
5 database.default.username = root
6 database.default.password = ''
7 database.default.DBDriver = MySQLi
8 database.default.port = 3306
9
```

## CI4 Database Connection Deliverables + Next Steps / Risks

- DB config in .env
- CI4 connects to MySQL
- Models: User / Project / Bid
- Flow works: project → bid → records
- Done: ERD + tables + sample data
- Next: indexes + validation + FK rules

# Frontend Architecture & UI Design

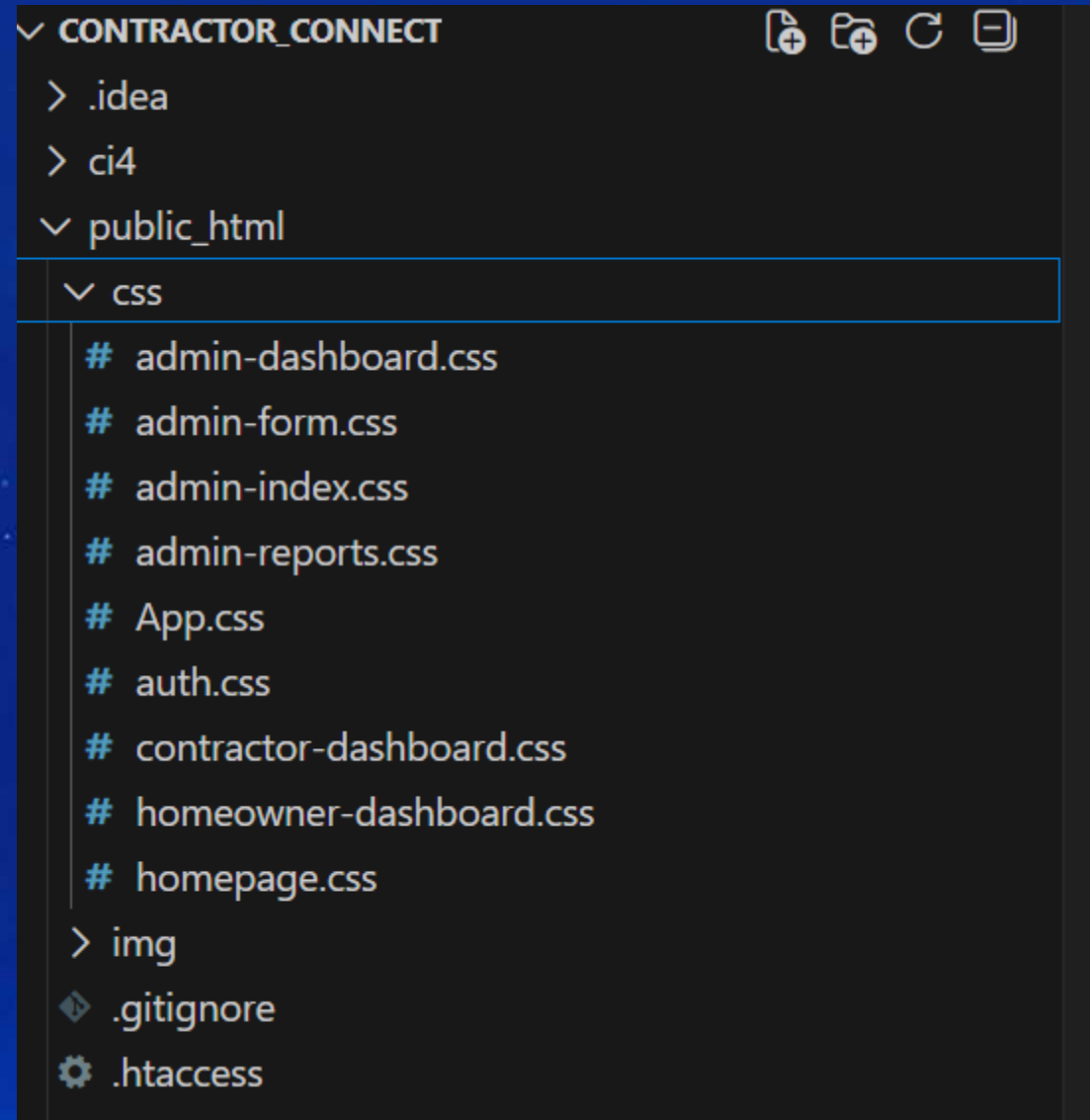
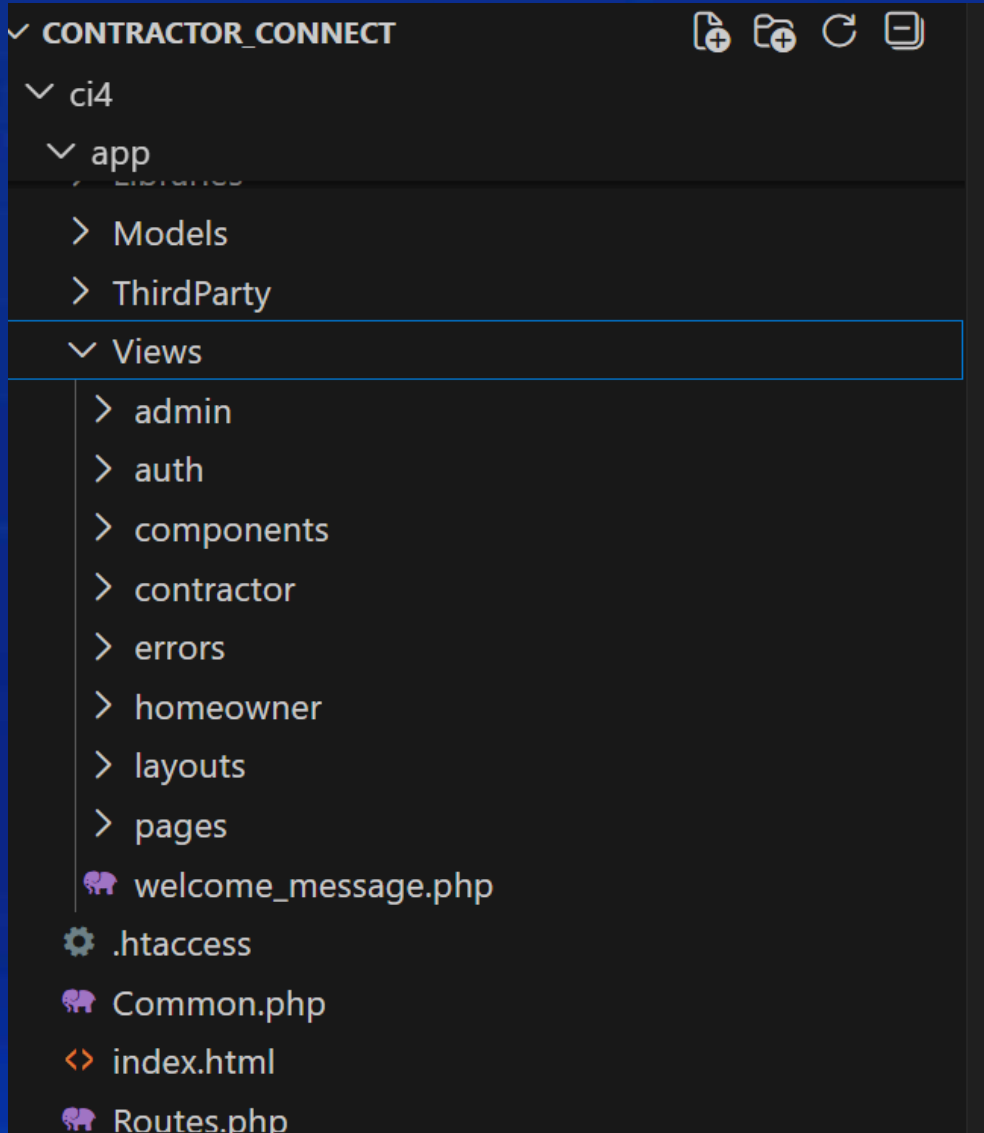
- Designed user interface for all pages
- Implemented responsive layouts using HTML & CSS
- Created dashboard UI components
- Improved navigation and user experience
- Integrated frontend views with backend controllers



# Frontend Architecture & UI Design

## Technologies Used

- HTML5
- CSS3
- PHP (Views in CodeIgniter)
- MVC Architecture
- Bootstrap



# Folder Structure

## *Planned Improvements :-*

- Fix issues in the **User Registration process**
- Add “**Back to Dashboard**” navigation button in all role-based views (Admin, Homeowner, Contractor)
- Implement **Edit Project functionality** for both Contractors and Homeowners
- Enable **View All Project Bids** for Contractors and Homeowners
- Fix and improve **View & Update Profile** functionality for both roles
- Enhance **Contractor Dashboard** by adding more management features
- Adding a consistent footer across all pages for better structure and navigation

## *Technical Challenges:-*

- Adding new features may impact existing APIs and cause:
  - Page loading failures
  - Routing issues
  - Data binding errors
- Maintaining stability while extending features requires:
  - Proper testing before deployment
  - Careful integration between frontend and backend
  - Avoiding breaking existing controller logic

# Quality Assurance

## Learning Plan:

LinkedIn Learning:

Programming Foundations: Software Testing Q/A

## Testing Framework:

CodeIgniter 4 Testing Suite

PHPUnit

## State Management:

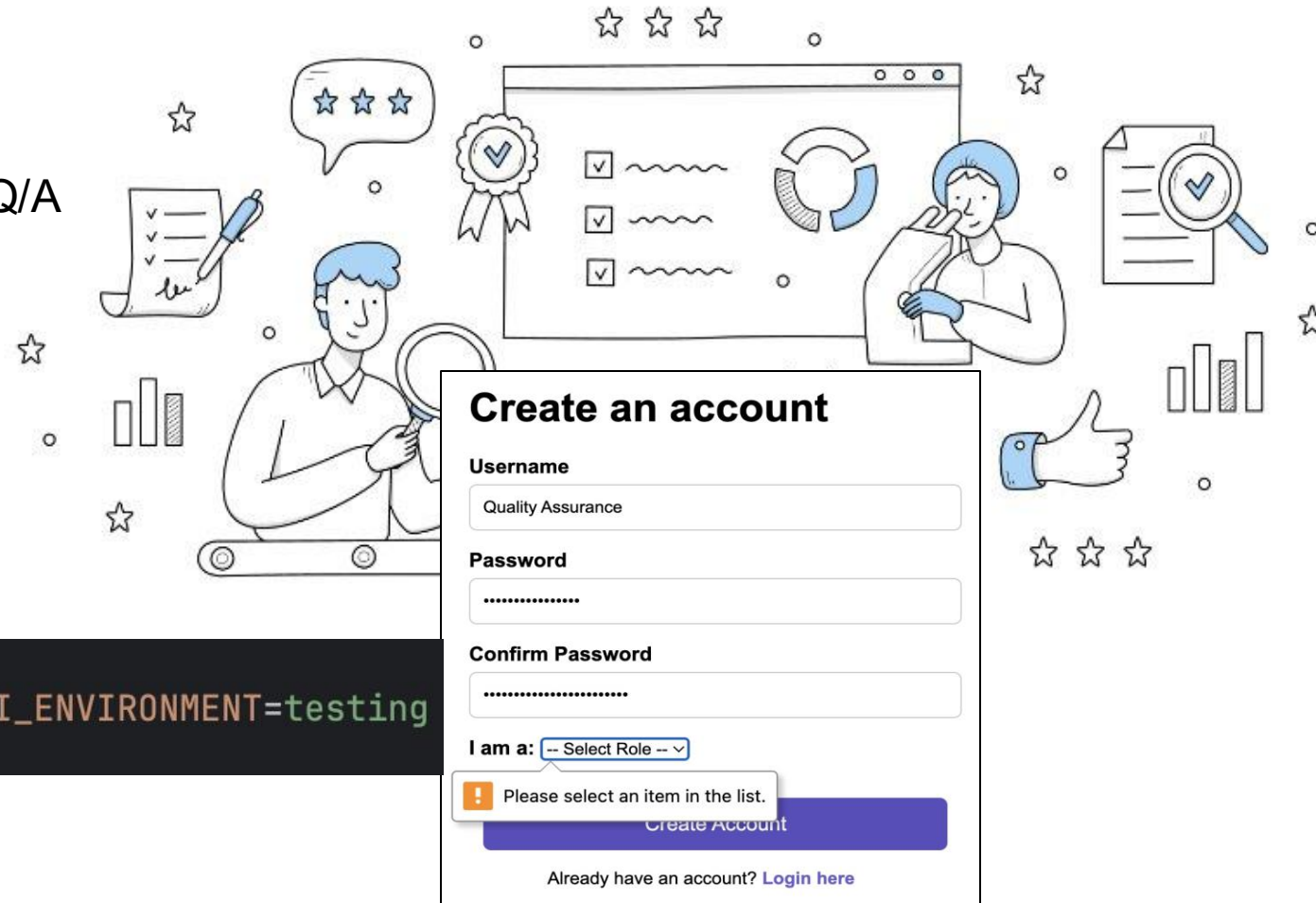
DatabaseTestTrait

Local MySQL

## Data Strategy:

FakerFactory

```
CI_ENVIRONMENT=testing
```



# Unit Testing

## Integration Testing:

- Testing full response cycle

## Model Testing (Business logic):

- Project creation fails without a title
- Verify deadline has not expired
- Budget minimum can't exceed maximum

## Feature Testing (User's path):

- Is the user routed to the appropriate page on login?
- Ex:
  - Simulating user login with POST request
  - Verifying redirects based on roles
  - Verifying Auth filters are protecting restricted routes



# Contractor Connect

## Challenges:

- Normalizing URLs to handle prefixes
- Using Query Builder to isolate authentication logic
- Working with current code

## Next Steps:

- Connect GitHub directly to cPanel
- Integrating GitHub Actions
- Install a Driver (Xdebug) to identify missed edge cases

## Learning Plan:

- LinkedIn Learning Course: GitHub Actions for CI/CD
- Read documentation on CI4 Testing Suite



# Contractor Connect

Thank you

**Adel Alhajhussain** – Backend Developer

**Mehdi Jazi** – Database Management

**Shifa Karnelia** – UI Developer

**Sena Karnelia** – Frontend Developer

**Eric Laudrum** – Quality Assurance