

Simple CRUD with Codeigniter and MySQL

Name: Rhazzhamir I. Aguilar **BSIT-3B**

Barangay Information and Management System (BIMS)

What is CRUD?

CRUD stands for:

- **C - Create (Add new data)**
- **R - Read (Display or view data)**
- **U - Update (Edit or modify data)**
- **D - Delete (Remove data)**

In this tutorial, we will build a simple student list web app that allows you to Create, Read, and Delete student information using CodeIgniter 4.

What is CodeIgniter?

CodeIgniter is a lightweight PHP framework used to build web applications quickly and easily. It's beginner-friendly and works well for simple systems like inventory, student records, etc.

Recap of Activity 5

Before continuing with this tutorial (Activity 6), let's quickly review what was done in **Activity 5:**

- **✓ Step 1:** Searched and explored the official **CodeIgniter Documentation**
- **✓ Step 2:** Installed **Composer** as a PHP dependency manager
- **✓ Step 3:** Created a new **CodeIgniter Project** using Composer
- **✓ Step 4:** Built a simple **User Interface (UI)** layout for the CRUD system
- **✓ Step 5:** Set up a **Controller** to manage logic and flow
- **✓ Step 6:** Configured the **Routes** to connect URLs to controller actions
- **✓ Step 7:** Creating User Dashboard

You should now have a basic CodeIgniter setup with UI, controller, and routing. If not, review Activity 5 before starting this tutorial.



Let's begin with Activity 6: Creating CRUD Operations

Step 1: Install Requirements

You'll need:

- A computer or laptop
- **XAMPP** (includes Apache + MySQL)

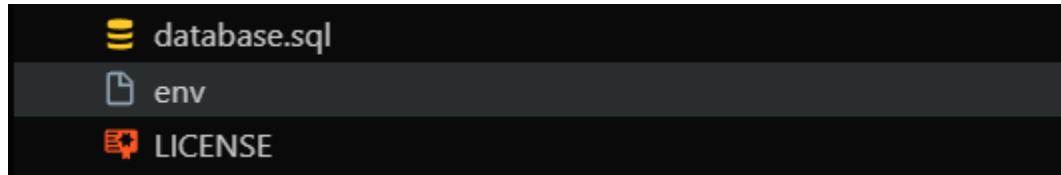
- **CodeIgniter 4**
- A browser (like Chrome)

Step 2: Open XAMPP and Start Services

- Open **XAMPP Control Panel**
- Start both **Apache** and **MySQL**
- Open vscode

Step 3: Connect CodeIgniter to the Database

1. Inside your CodeIgniter project folder, look for a file named env



2. **Copy it, paste** it in the same folder, and rename it to .env



3. Open .env and **uncomment and edit** the following settings:

```

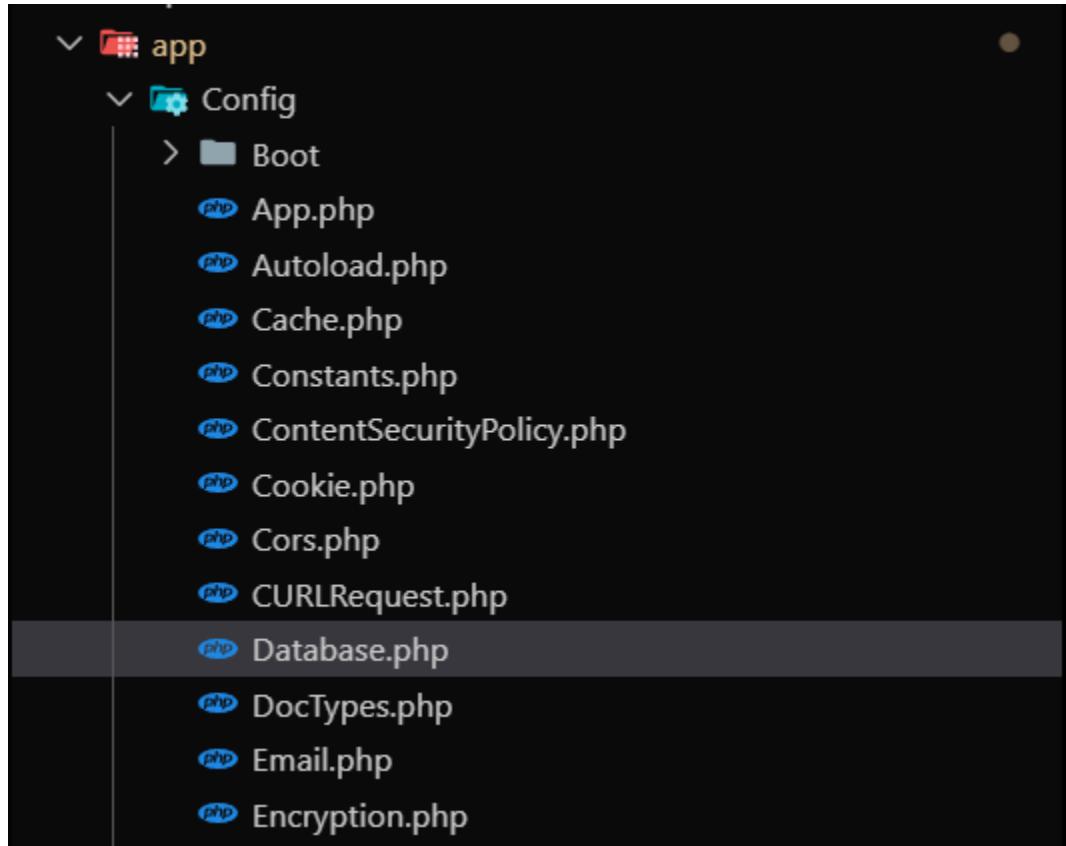
15 #-----
16
17 CI_ENVIRONMENT = development
18
19 #-----
20 # APP
21 #
22
23 # app.baseURL = ''
24 # If you have trouble with `.` , you could also use `__` .
25 # app_baseURL = ''
26 # app.forceGlobalSecureRequests = false
27 # app.CSPEnabled = false
28
29 #-----
30 # DATABASE
31 #
32
33 database.default.hostname = localhost
34 database.default.database = BarangayIMS|root
35 database.default.username = root
36 database.default.password = [REDACTED]
37 database.default.DBDriver = MySQLi
38 database.default.DBPrefix =
39 database.default.port = 3306
40

```

The code editor shows the .env file with several commented-out lines. Line 34 has 'BarangayIMS' typed in, and line 36 has '[REDACTED]' in a redacted box.

Alternatively, you can also configure the database in the app/Config/Database.php file:

1. Open the file app/Config/Database.php.



2. Look for the \$default array and edit it like this:

A screenshot of a code editor showing the 'Database.php' file. The code is as follows:

```
10 class Database extends config
11 {
12     /*
13      * @var array<string, mixed>
14     */
15     public array $default = [
16         'DSN' => '',
17         'hostname' => 'localhost',
18         'username' => 'root',
19         'password' => '',
20         'database' => 'BarangayIMS',
21         'DBDriver' => 'MySQLi',
22         'DBPrefix' => '',
23         'pConnect' => false,
24         'DBDebug' => true,
25         'charset' => 'utf8mb4',
26         'DBCollat' => 'utf8mb4_general_ci',
27         'swapPre' => '',
28         'encrypt' => false,
29         'compress' => false,
30         'strictOn' => false,
31         'failover' => [],
32         'port' => 3306,
33         'numberNative' => false,
34         'foundRows' => false,
35         'dateFormat' => [
36             'date' => 'Y-m-d',
37             'datetime' => 'Y-m-d H:i:s'
38         ]
39     ];
40 }
```

The '\$default' array is highlighted with a light blue selection.

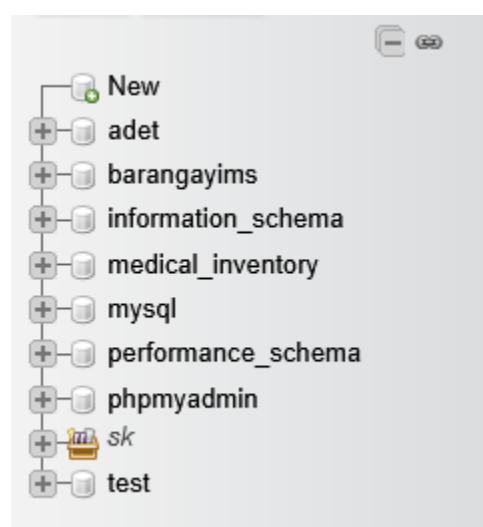
Step 5: Create the Database and Table

1. Create the Database:

- Go to <http://localhost/phpmyadmin> (make sure XAMPP is running).

The screenshot shows the phpMyAdmin configuration page. It includes sections for General settings (server connection collation set to utf8mb4_unicode_ci), Database server (server details like IP, type, version, and user), Web server (Apache, PHP, and MySQL details), and phpMyAdmin (version information). A message at the bottom indicates an available update to version 5.2.2.

- Click on New to create a database.



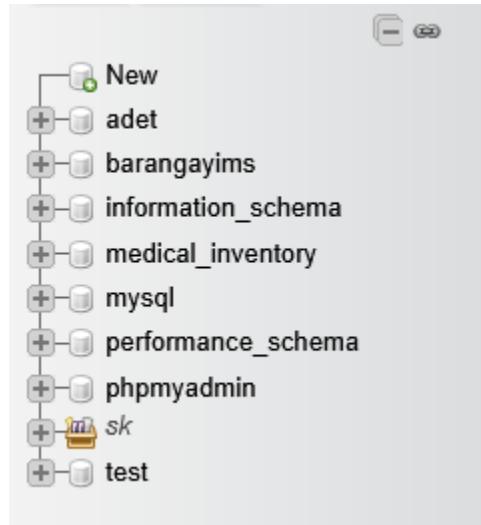
- Name the database BarangayIMS (or whatever name you chose in the .env file) and click Create.

The screenshot shows the 'Create database' dialog box. It has a text input field containing 'BarangayIMS' and a dropdown menu set to 'utf8mb4_general_ci'. A 'Create' button is visible on the right.

2. Create a Table:

After creating the database, you'll need to create a table for storing student data. Follow these steps:

- In phpMyAdmin, select the BarangayIMS database.



- Click on the SQL tab at the top to run an SQL query.



- Paste the following SQL code to create a residents table:

The screenshot shows the SQL editor of phpMyAdmin with the following code pasted into the query box:

```
1 CREATE TABLE students (
2     id INT(11) PRIMARY KEY AUTO_INCREMENT,
3     photo VARCHAR(255) DEFAULT NULL,
4     full_name VARCHAR(255) NOT NULL,
5     date_of_birth DATE NOT NULL,
6     age INT(11) NOT NULL,
7     gender ENUM('Male', 'Female') NOT NULL,
8     civil_status VARCHAR(50) NOT NULL,
9     address TEXT NOT NULL,
10    contact VARCHAR(15) NOT NULL,
11    occupation VARCHAR(255) DEFAULT NULL,
12    date_registered DATETIME NOT NULL,
13    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
14    updated_at DATETIME DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP
15 );
16
```

A yellow callout box in the upper right corner of the editor area says: "You have a previously saved query. Click Get auto-saved query to load the query."

Below the code, there are several buttons: 'SELECT *', 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'Clear', 'Format', 'Get auto-saved query', 'Bind parameters', 'Delimiter :', 'Show this query here again', 'Retain query box', 'Rollback when finished', 'Enable foreign key checks', and a 'Go' button.

Now that the **database** and **table** are set up, you can move on to **creating models, controllers, routes** and **views** for the CRUD operations.

Step 5 : Create a Model in CodeIgniter

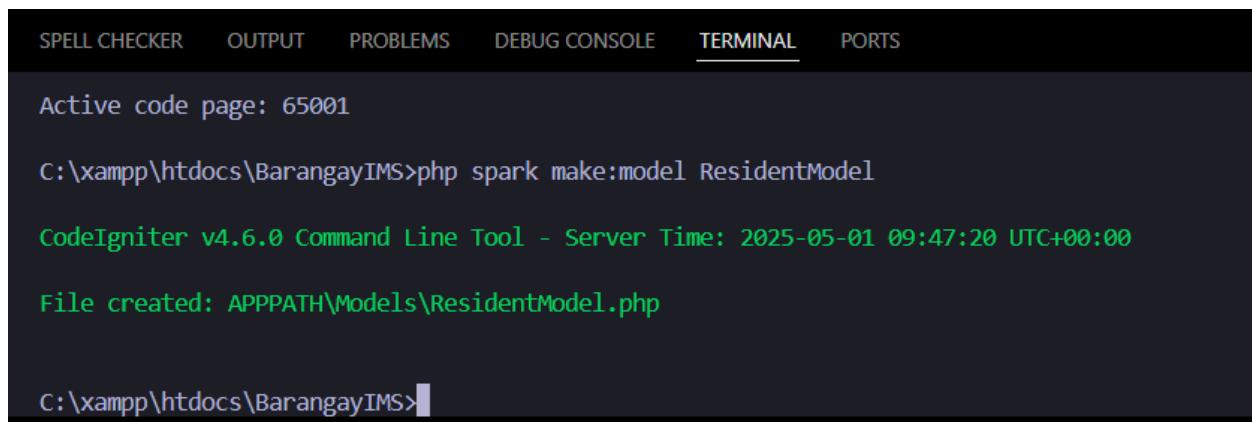
What is a Model?

A **Model** in CodeIgniter is a file that connects your application to the **database**.

- It is responsible for **retrieving, inserting, updating, and deleting** data.
- Think of it as the **bridge** between your database and your website.
- Models help keep your code organized by separating the database logic from the rest of your application (like the user interface and page structure).

Simple Example: If you want to **add a student** to your system, the model is the part that actually saves that student's info into the database.

1. Open your terminal in the project root



The screenshot shows a terminal window with the following output:

```
SPELL CHECKER    OUTPUT    PROBLEMS    DEBUG CONSOLE    TERMINAL    PORTS

Active code page: 65001
C:\xampp\htdocs\BarangayIMS>php spark make:model ResidentModel
CodeIgniter v4.6.0 Command Line Tool - Server Time: 2025-05-01 09:47:20 UTC+00:00
File created: APPPATH\Models\ResidentModel.php

C:\xampp\htdocs\BarangayIMS>
```

Use this command to generate a model:

Here's how your ResidentModel should look based on your table structure:

```
app > Models > ResidentModel.php > PHP > ResidentModel > table
1 <?php
2
3 namespace App\Models;
4
5 use CodeIgniter\Model;
6
7 class ResidentModel extends Model
8 {
9     protected $table = 'residents';
10    protected $primaryKey = 'id';
11
12    protected $allowedFields = [
13        'photo',
14        'full_name',
15        'date_of_birth',
16        'age',
17        'gender',
18        'civil_status',
19        'address',
20        'contact',
21        'occupation',
22        'date_registered',
23        'created_at',
24        'updated_at',
25    ];
26
27    protected $useTimestamps = true;
28    protected $createdField = 'created_at';
29    protected $updatedField = 'updated_at';
30 }
31
```

Notes: ⚠ Make sure the value of \$table matches your actual table name in the database (e.g., students or residents).

Step 6: Creating a Controller (Part of Activity 5)

*note: Creating a controller was already introduced in **Activity 5**, but here we will go deeper as part of our full CRUD setup.*

What is a Controller?

A **Controller** is the part of CodeIgniter that **handles user requests** and **controls what to do with them**.

- It acts like a traffic manager: when the user clicks a button or visits a page, the controller decides **which data to get**, **which page to show**, and **what action to perform** (like add, edit, or delete).
- It **talks to the Model** for data and **shows it using the View**.

How to Create a Controller

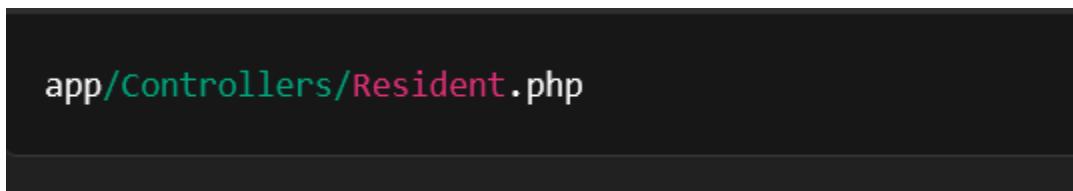
Use the terminal in your project directory and run:



```
SPELL CHECKER (41) OUTPUT PROBLEMS DEBUG CONSOLE TERMINAL PORTS JavaSE-2

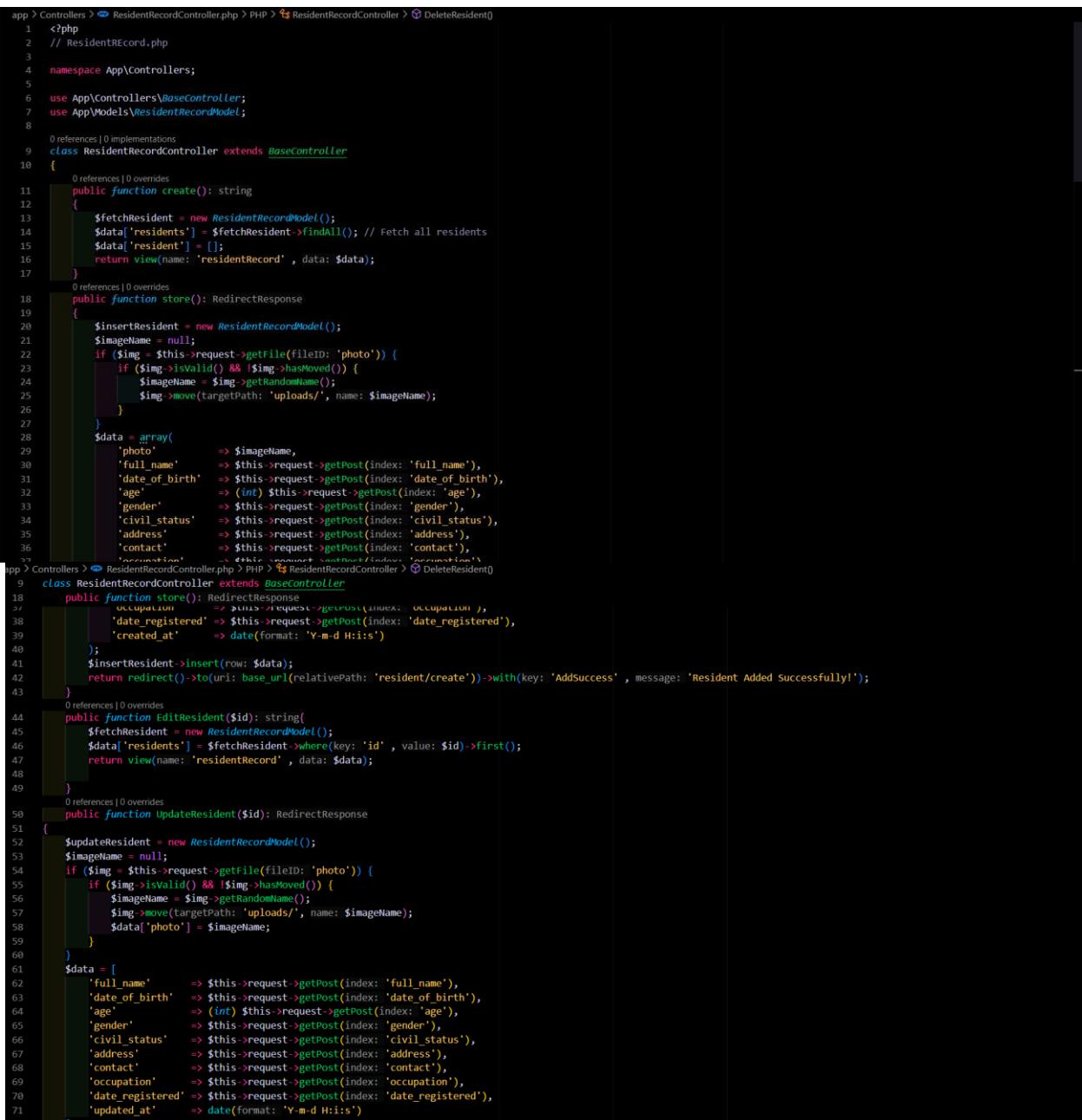
C:\xampp\htdocs\BarangayIMS>php spark make:controller ResidentController
```

This will create a file:



Edit the Controller

Open the Resident.php file and start with this basic setup:



```
app > Controllers > ResidentRecordController.php > PHP > ResidentRecordController > DeleteResident()
1 <?php
2 // ResidentRecordController
3
4 namespace App\Controllers;
5
6 use App\Controllers\BaseController;
7 use App\Models\ResidentRecordModel;
8
9 class ResidentRecordController extends BaseController
{
10     public function create(): string
11     {
12         $fetchResident = new ResidentRecordModel();
13         $data['residents'] = $fetchResident->findAll(); // Fetch all residents
14         $data['resident'] = [];
15         return view('residentRecord', $data);
16     }
17 }
18
19 public function store(): RedirectResponse
20 {
21     $insertResident = new ResidentRecordModel();
22     $imageName = null;
23     if ($img = $this->request->getFile('photo')) {
24         if ($img->isValid() && !$img->hasMoved()) {
25             $imageName = $img->getRandomName();
26             $img->move('uploads/', $imageName);
27         }
28     }
29     $data = array(
30         'photo' => $imageName,
31         'full_name' => $this->request->getPost('full_name'),
32         'date_of_birth' => $this->request->getPost('date_of_birth'),
33         'age' => (int) $this->request->getPost('age'),
34         'gender' => $this->request->getPost('gender'),
35         'civil_status' => $this->request->getPost('civil_status'),
36         'address' => $this->request->getPost('address'),
37         'contact' => $this->request->getPost('contact'),
38         'occupation' => $this->request->getPost('occupation'),
39         'date_registered' => $this->request->getPost('date_registered'),
40         'created_at' => date('Y-m-d H:i:s')
41     );
42     $insertResident->insert($data);
43     return redirect()->to($base_url . 'resident/create')->with('key', 'AddSuccess', 'message', 'Resident Added Successfully!');
44 }
45
46 public function EditResident($id): string
47 {
48     $fetchResident = new ResidentRecordModel();
49     $data['residents'] = $fetchResident->where('id', $id)->first();
50     return view('residentRecord', $data);
51 }
52
53 public function UpdateResident($id): RedirectResponse
54 {
55     $updateResident = new ResidentRecordModel();
56     $imageName = null;
57     if ($img = $this->request->getFile('photo')) {
58         if ($img->isValid() && !$img->hasMoved()) {
59             $imageName = $img->getRandomName();
60             $img->move('uploads/', $imageName);
61             $data['photo'] = $imageName;
62         }
63     }
64     $data = [
65         'full_name' => $this->request->getPost('full_name'),
66         'date_of_birth' => $this->request->getPost('date_of_birth'),
67         'age' => (int) $this->request->getPost('age'),
68         'gender' => $this->request->getPost('gender'),
69         'civil_status' => $this->request->getPost('civil_status'),
70         'address' => $this->request->getPost('address'),
71         'contact' => $this->request->getPost('contact'),
72         'occupation' => $this->request->getPost('occupation'),
73         'date_registered' => $this->request->getPost('date_registered'),
74         'updated_at' => date('Y-m-d H:i:s')
75     ];
76 }
```

```

app > Controllers > ResidentRecordController.php > PHP > ResidentRecordController > DeleteResident()
9  v class ResidentRecordController extends BaseController
10 < public function updateResident($id): RedirectResponse
11   {
12     $img->move('uploads/', name: $imageName);
13     $data['photo'] = $imageName;
14   }
15   $data = [
16     'full_name' => $this->request->getPost(index: 'full_name'),
17     'date_of_birth' => $this->request->getPost(index: 'date_of_birth'),
18     'age' => ($int) $this->request->getPost(index: 'age'),
19     'gender' => $this->request->getPost(index: 'gender'),
20     'civil_status' => $this->request->getPost(index: 'civil_status'),
21     'address' => $this->request->getPost(index: 'address'),
22     'contact' => $this->request->getPost(index: 'contact'),
23     'occupation' => $this->request->getPost(index: 'occupation'),
24     'date_registered' => $this->request->getPost(index: 'date_registered'),
25     'updated_at' => date(format: 'Y-m-d H:i:s')
26   ];
27
28   if (!$imageName) {
29     unset($data['photo']);
30   }
31   $updateResident->update(id: $id, row: $data);
32   return redirect()->to(uri: base_url(relativePath: 'resident/create'))->with(key: 'UpdateSuccess', message: 'Resident Updated Successfully!');
33 }
34
35 0 references | 0 overrides
36 public function deleteResident($id): RedirectResponse
37 {
38   $deleteResident = new ResidentRecordModel();
39   $resident = $deleteResident->find(id: $id);
40   if (!$resident) {
41     return redirect()->to(uri: base_url(relativePath: 'resident/create'))->with(key: 'error', message: 'Resident not found!');
42   }
43   $deleteResident->delete(id: $id);
44   return redirect()->to(uri: base_url(relativePath: 'resident/create'))->with(key: 'DeleteSuccess', message: 'Resident Deleted Successfully!');
45 }
46
47 }

```

What this Controller does:

- index() – show all residents
- create() – show form to add a resident
- store() – save new resident to database
- edit() – show form to edit an existing resident
- update() – update resident info
- delete() – delete a resident

Controller is now ready!

Step 7: Setting Up Routes (Part of Activity 5)

*note: Setting up routes was already introduced in **Activity 5**, but here we'll apply it in our actual CRUD system.*

What is a Route?

A **Route** tells CodeIgniter which **controller function** to run when a user visits a certain URL.

Think of it as a **map** that connects website links (like /resident/create) to functions inside your controller (like Resident::create()).

How to Add Routes

Open this file in your project:

```
app/Config/Routes.php
```

Scroll down to the part where you can define your own routes, and add this:

```
$routes->get('/resident', 'Resident::index');
$routes->get('/resident/create', 'Resident::create');
$routes->post('/resident/store', 'Resident::store');
$routes->get('/resident/edit/(:num)', 'Resident::edit/$1');
$routes->post('/resident/update/(:num)', 'Resident::update/$1');
$routes->get('/resident/delete/(:num)', 'Resident::delete/$1');
```

What These Routes Do:

URL	Action
/resident	Show list of residents
/resident/create	Show form to add new resident
/resident/store	Save new resident (via POST)
/resident/edit/{id}	Show form to edit resident
/resident/update/{id}	Update resident (via POST)
/resident/delete/{id}	Delete resident

Your routes are now connected!

Step 8: Creating Views (Part of Activity 5)

note: Creating the UI (views) was already part of **Activity 5**. If you already designed the interface there, we would now connect it to make it functional.

What is a View?

A **View** in CodeIgniter is a file that contains **HTML code**. It's what the user **sees** on the browser.

- Views are used to display forms, tables, and other visual elements.
- They **receive data from the controller** and show it in a human-friendly way.

Where to Place the Views

Save your view files in this folder:

```
app/Views/residents/
```

Make sure you create the residents folder if it doesn't exist yet.

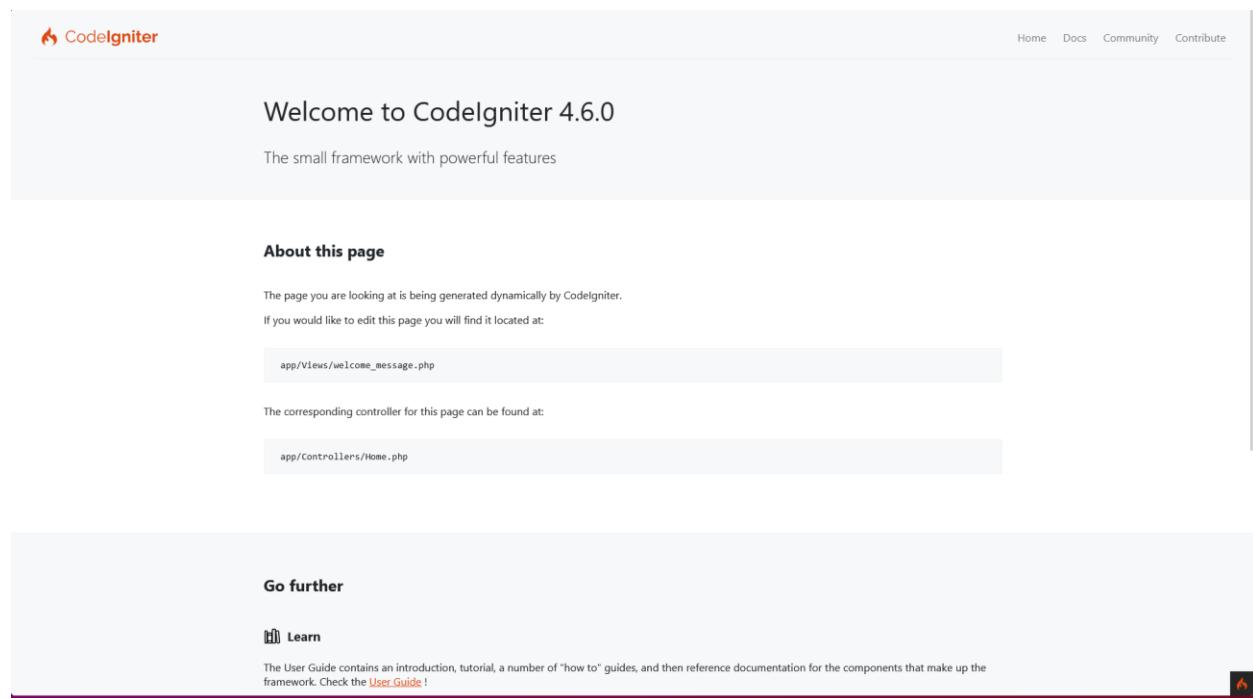
Save the file, restart the server (`php spark serve`), and visit:

```
C:\xampp\htdocs\BarangayIMS>php spark serve

CodeIgniter v4.6.0 Command Line Tool - Server Time: 2025-05-01 10:27:20 UTC+00:00

CodeIgniter development server started on http://localhost:8080
Press Control-C to stop.
[Thu May 1 18:27:21 2025] PHP 8.2.12 Development Server (http://localhost:8080) started
```

Open your browser and go to <http://localhost:8080>. You should see the CodeIgniter welcome page.



The screenshot shows the 'Welcome to CodeIgniter 4.6.0' page. At the top, there's a navigation bar with the CodeIgniter logo, Home, Docs, Community, and Contribute links. Below the header, the main content area has a light gray background. It displays the text "Welcome to CodeIgniter 4.6.0" and "The small framework with powerful features". A section titled "About this page" provides information about the dynamic generation of the page and its location at "app/Views/welcome_message.php". Another section indicates the controller is located at "app/Controllers/Home.php". At the bottom, a "Go further" section includes a "Learn" link, and a note about the User Guide. The footer features a dark red bar with the CodeIgniter logo.

This is the UI from AdminLTE that I provided.

We are now using it as part of our project. Below is the **before and after** modification of the UI:

Before (Default AdminLTE UI)

The screenshot shows the AdminLTE 3 dashboard. On the left is a sidebar with navigation links like Home, Contact, Search, Dashboard, Widgets, Layout Options, Charts, UI Elements, Forms, Tables, Examples, Calendar, Gallery, Kanban Board, and Mailbox. The main area has four large cards: 'New Orders' (150), 'Bounce Rate' (53%), 'User Registrations' (44), and 'Unique Visitors' (65). Below these are two charts: 'Sales' (an area chart showing monthly sales) and 'Visitors' (a map of the United States with state-level visitor data). A 'Direct Chat' sidebar shows a message from Sarah Bullock.

```
<!-- Example AdminLTE card before modification -->


<div class="card-header">
    <h3 class="card-title">Default Title</h3>
  </div>
  <div class="card-body">
    <!-- Default content -->
  </div>
</div>


```

After (Modified to fit our Resident CRUD system)

The screenshot shows the modified dashboard for the Resident CRUD system. The sidebar includes Management Records, Dashboard, Resident Record, Budget, and Officials. The main area features a night photograph of a modern building with illuminated glass walls and a red structural frame. Overlaid on the image are four cards with statistics: 'Budget' (308,000), 'Officials' (10), 'Resident record' (3), and 'Unique Visitors' (65). The URL at the bottom is http://localhost:8080/Admin/Dashboard#.

```
        Copy Edit
```

```
<div class="card">
  <div class="card-header">
    <h3 class="card-title">Resident List</h3>
    <div class="card-tools">
      <a href="/resident/create" class="btn btn-primary btn-sm">Add Resident</a>
    </div>
  </div>
  <div class="card-body">
    <table class="table table-bordered table-hover">
      <thead>
        <tr>
          <th>ID</th>
          <th>Full Name</th>
          <th>Age</th>
          <th>Gender</th>
          <th>Contact</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($residents as $resident): ?>
        <tr>
          <td><?= $resident['id']; ?></td>
          <td><?= $resident['full_name']; ?></td>
          <td><?= $resident['age']; ?></td>
          <td><?= $resident['gender']; ?></td>
          <td><?= $resident['contact']; ?></td>
          <td>
            <a href="/resident/edit/<?= $resident['id']; ?>" class="btn btn-warning btn-sm">Edit<
            <a href="/resident/delete/<?= $resident['id']; ?>" class="btn btn-danger btn-sm" oncl:>
          </td>
        </tr>
      <?php endforeach; ?>
    </tbody>
  </table>
</div>
</div>
```

This modified layout fits nicely with your AdminLTE design and makes your CRUD UI more modern and user-friendly.

After that, Let's proceed to the objective of the proposal with the interaction between the Admin and the User in your Barangay Certificate Request system, and build it based on your CRUD app in CodeIgniter 4.

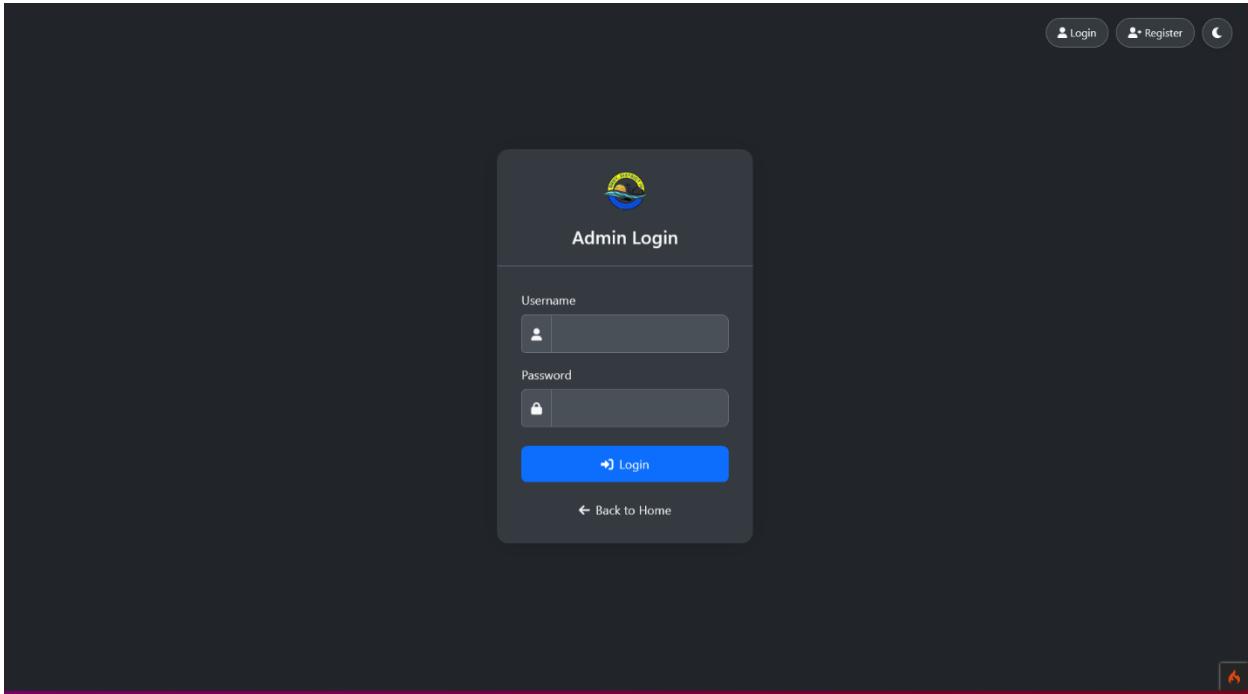
Title Proposal Objective – What You See Inside

1. **Barangay Budget Management** – To provide a structured system for recording, tracking, and managing the barangay's allocated budget, expenses, and financial reports.
2. **Barangay Member and Officials Directory** – To maintain an updated database of barangay officials, staff, and community leaders with their respective roles and responsibilities.
3. **Resident Records Management** – To create a secure and organized digital record-keeping system for all residents, including personal information, household data, and demographic details.

Before we proceed to the Admin dashboard we need to login to the admin form

Admin Log In

This login screen is only for the Barangay official or assigned Barangay members. They need special credentials (admin access) to enter.



After logging in, the admin sees a control panel with many tools. They can manage and update the system.

The screenshot shows the Barangay Dist IV dashboard. At the top, there are three cards: 'Total Budget' (₱1,000,000.00), 'Total Expenses' (₱100,001.00), and 'Remaining Balance' (₱899,999.00). Below these is a section titled 'Resident Records' with a search bar and a table of recent registrations. The table includes columns for 'Recent Registrations' and 'Date Registered'. The data shows five entries:

Recent Registrations	Date Registered
haha hahahaha	May 08, 2025
Nathaniel James Cruz	Apr 17, 2025
ras new new	Apr 19, 2025
Reonadylyn Iglesias Parina	May 11, 2025
rochelle Iglesias Aguilar	Apr 18, 2025

Showing 1 to 5 of 5 entries

Resident Management

- Admin can view resident information, remove inactive users.

The screenshot shows the Resident Management page under the 'Residents' tab. It displays a table of 'Residents List' with columns: #, FirstName, MiddleName, LastName, Phone Number, Created_at, Updated_at, and Actions. The table contains 10 entries from 1 to 10. Each row has a 'View' icon (blue eye) and a 'Delete' icon (red trash bin). The table also includes a search bar at the top and pagination at the bottom.

#	FirstName	MiddleName	LastName	Phone Number	Created_at	Updated_at	Actions
1	Juan	Carlos	Dela Cruz	09171234567	2025-04-17 12:02:14	2025-04-17 12:02:14	
2	Maria	Angelica	Santos	09181234567	2025-04-17 12:02:14	2025-04-17 12:02:14	
3	Pedro	Luis	Reyes	09201234567	2025-04-17 12:02:14	2025-04-17 12:02:14	
4	Ana	Beatriz	Mendoza	09301234567	2025-04-17 12:02:14	2025-04-17 12:02:14	
5	Carlos	Miguel	Fernandez	09451234567	2025-04-17 12:02:14	2025-04-17 12:02:14	
6	Liza	Marie	Gutierrez	09172345678	2025-04-17 12:30:56	2025-04-17 12:30:56	
7	Rafael	Tomas	Navarro	09173456789	2025-04-17 12:30:56	2025-04-17 12:30:56	
8	Camille	Joyce	Herrera	09174567890	2025-04-17 12:30:56	2025-04-17 12:30:56	
9	Ernesto	Juan	Ramirez	09175678901	2025-04-17 12:30:56	2025-04-17 12:30:56	
10	Daphne	Elaine	Soriano	09176789012	2025-04-17 12:30:56	2025-04-17 12:30:56	

Showing 1 to 10 of 18 entries

Official Management

- The admin can add a new Barangay official when the current official's term has ended.

Add official

The screenshot shows the 'Add New Official' form. It includes fields for First Name, Middle Name, Last Name, Position, Term, Email, Phone Number, and Address. There is also a Profile Image input field with a browse button. At the bottom are 'Save Official' and 'Cancel' buttons.

Barangay Dist IV

Add Official

Add New Official

First Name *

Middle Name

Last Name *

Position *

Term *

Email

Phone Number

Address

Profile Image

Recommended size: 300x300 pixels. Max file size: 2MB

Browse

Save Official Cancel

Copyright © 2025 Barangay Information and Management System All rights reserved.

List of official

The screenshot shows a table titled 'Officials List' with columns for #, Image, Name, Position, Term, Phone Number, Created_at, Updated_at, and Actions. The table contains four entries. At the bottom, it shows 'Showing 1 to 4 of 4 entries' and navigation buttons for Previous, Next, and a page number '1'.

Barangay Dist IV

Officials List

Officials List

Show 10 entries Search:

#	Image	Name	Position	Term	Phone Number	Created_at	Updated_at	Actions
1		Rhazzhamir Iglesias Aguilar	SK chairperson	2025	09319697096	2025-04-19 04:09:37	2025-04-19 04:48:41	
2		Reo Iglesias Parina	SK secretary	2025	093196797963	2025-04-19 04:20:28	2025-04-19 04:58:26	
3		sfsdfsdf asdasd aad	adasd	adasd	09319697096	2025-04-19 04:41:09	2025-04-19 04:48:26	
4		asdasd asdasd asdasd	asdasd	asdasd	09319697069	2025-04-19 04:48:14	2025-04-19 04:48:14	

Showing 1 to 4 of 4 entries

Previous **1** Next

Copyright © 2025 Barangay Information and Management System All rights reserved.

Budget Management

- Module for tracking and controlling barangay budget allocations.

Add budget

The screenshot shows a modal window titled "Add New Budget". Inside the modal, there is a dropdown menu for "Year" set to "2025". Below it is a field for "Amount" with a placeholder "₱". At the bottom right of the modal are two buttons: "Close" and "Save Budget". To the right of the modal, there is a sidebar with a green bar icon and a yellow bar icon. The yellow bar has the text "2025 Current Year". Below the sidebar, there is a table header with columns for "Year", "Amount", and "Actions". A single row is visible in the table, showing "2025" under "Year", "₱1,000,000.00" under "Amount", and two icons under "Actions".

Record Budget

The screenshot shows the main "Budget Management" page. At the top, there are summary statistics: "₱1,000,000.00" (Total Budget), "1" (Total Records), and "2025" (Current Year). Below this, there is a table titled "Budget Management" with a "Add New Budget" button at the top right. The table has columns for "Year", "Amount", and "Actions". One entry is listed: "2025" under "Year", "₱1,000,000.00" under "Amount", and two icons under "Actions". At the bottom of the page, there is a footer with the text "Copyright © 2025 Barangay Information and Management System All rights reserved." and a small logo.

Legal

- Explains how your data is collected, used, protected, and your rights within the Barangay Information and Management System.

The screenshot shows the 'Terms & Conditions' page of the Barangay Dist IV system. The left sidebar has a 'Legal' section selected. The main content area is titled 'Terms & Conditions' and contains five sections: 1. Acceptance of Terms, 2. Use of the System, 3. User Responsibilities, 4. System Access, and 5. Intellectual Property. Each section lists specific rules or rights. A note at the bottom states: 'All content and materials on this system are protected by intellectual property rights and are the property of the barangay.'

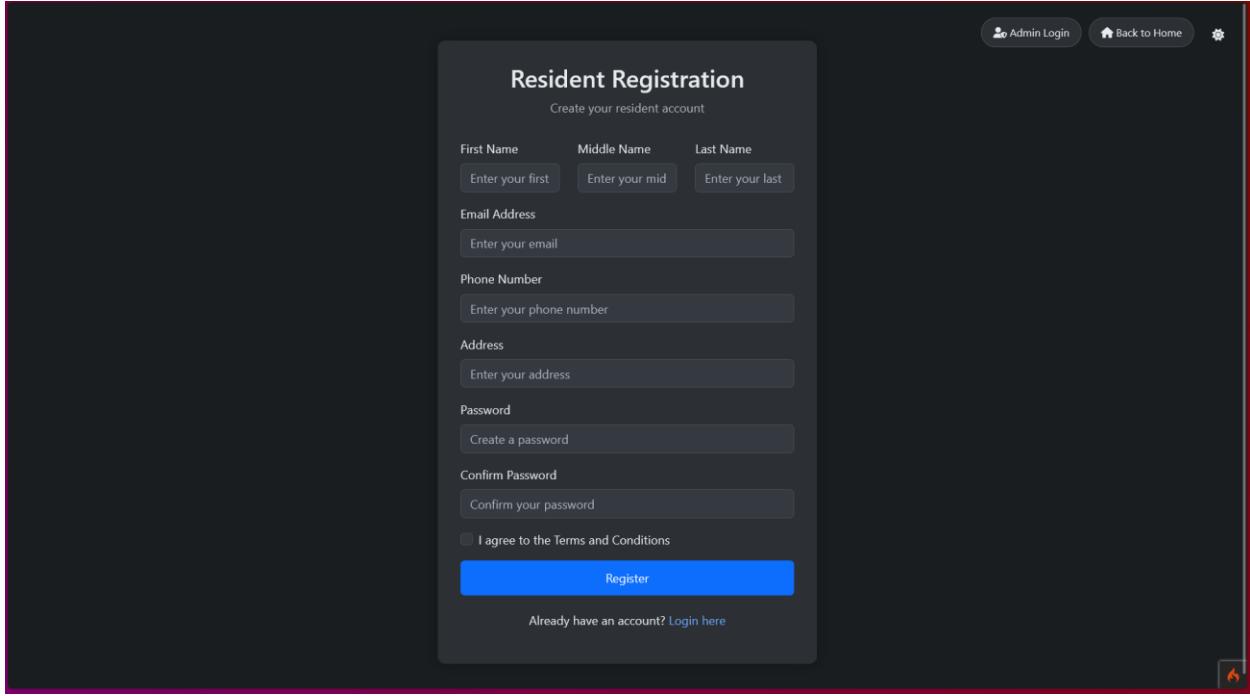
Certificate Management

- Allows residents to request and official barangay can approved the request.

The screenshot shows the 'Certificate Requests' page of the Barangay Dist IV system. The left sidebar has a 'Certificate' section selected. The main content area is titled 'Certificate Requests' and displays a table of 14 entries. The columns are: Request ID, Resident Name, Certificate Type, Purpose, Date Requested, Status, and Actions. The 'Actions' column contains icons for Approve (green), Pending (yellow), and Reject (red). The status column shows 'Approved' for requests 3, 9, 11, and 12, 'Pending' for others, and 'Rejected' for none. A note at the bottom says: 'Showing 1 to 10 of 14 entries'.

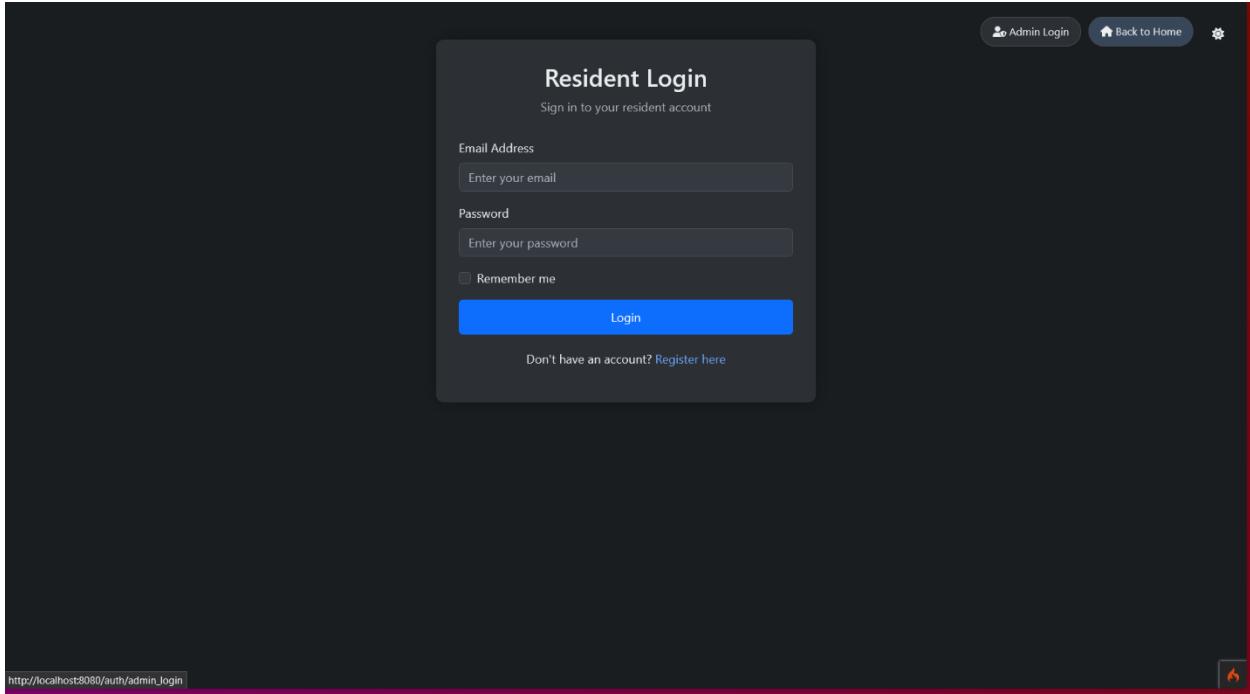
Request ID	Resident Name	Certificate Type	Purpose	Date Requested	Status	Actions
3	ras new new	Indigency	sadasdasdasdas	May 09, 2025	Approved	
4	ras new new	Business	asdadasdasdasd	May 09, 2025	Pending	
5	ras new new	Indigency	sdadadasaddsa	May 09, 2025	Pending	
6	ras new new	Residency	asdsdasdasdas	May 09, 2025	Pending	
7	ras new new	Business	asdadasdasdasdasd	May 09, 2025	Pending	
8	ras new new	Barangay Clearance	adsadasdasdasdasd	May 09, 2025	Pending	
9	Reonadylyn Iglesias Parina	Barangay Clearance	adasdasdasdasd	May 11, 2025	Approved	
10	Reonadylyn Iglesias Parina	Business	asdadasdasdasdasda	May 11, 2025	Pending	
11	Reonadylyn Iglesias Parina	Business	asdadasdasdasdad	May 11, 2025	Approved	
12	Reonadylyn Iglesias Parina	Business	asdadasdasdasdasd	May 11, 2025	Approved	

Before we proceed to the User dashboard we need to register to the registration



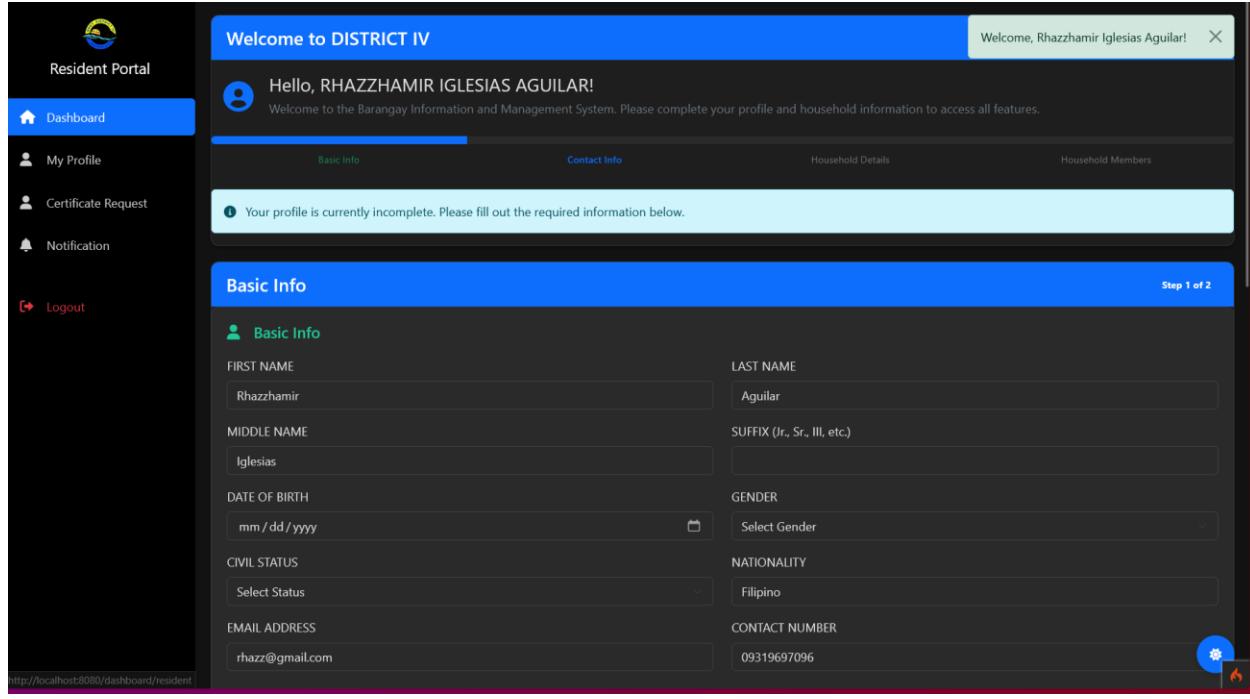
The screenshot shows a Resident Registration form on a dark-themed web page. At the top right are three buttons: 'Admin Login' (with a user icon), 'Back to Home' (with a house icon), and a gear icon. The main title 'Resident Registration' is centered at the top, with the subtitle 'Create your resident account' just below it. The form fields are arranged in rows: 'First Name' (placeholder 'Enter your first'), 'Middle Name' (placeholder 'Enter your mid'), and 'Last Name' (placeholder 'Enter your last'). Below these are 'Email Address' (placeholder 'Enter your email'), 'Phone Number' (placeholder 'Enter your phone number'), 'Address' (placeholder 'Enter your address'), 'Password' (placeholder 'Create a password'), and 'Confirm Password' (placeholder 'Confirm your password'). A checkbox labeled 'I agree to the Terms and Conditions' is followed by a large blue 'Register' button. At the bottom of the form is a link 'Already have an account? Login here'.

After register need to login



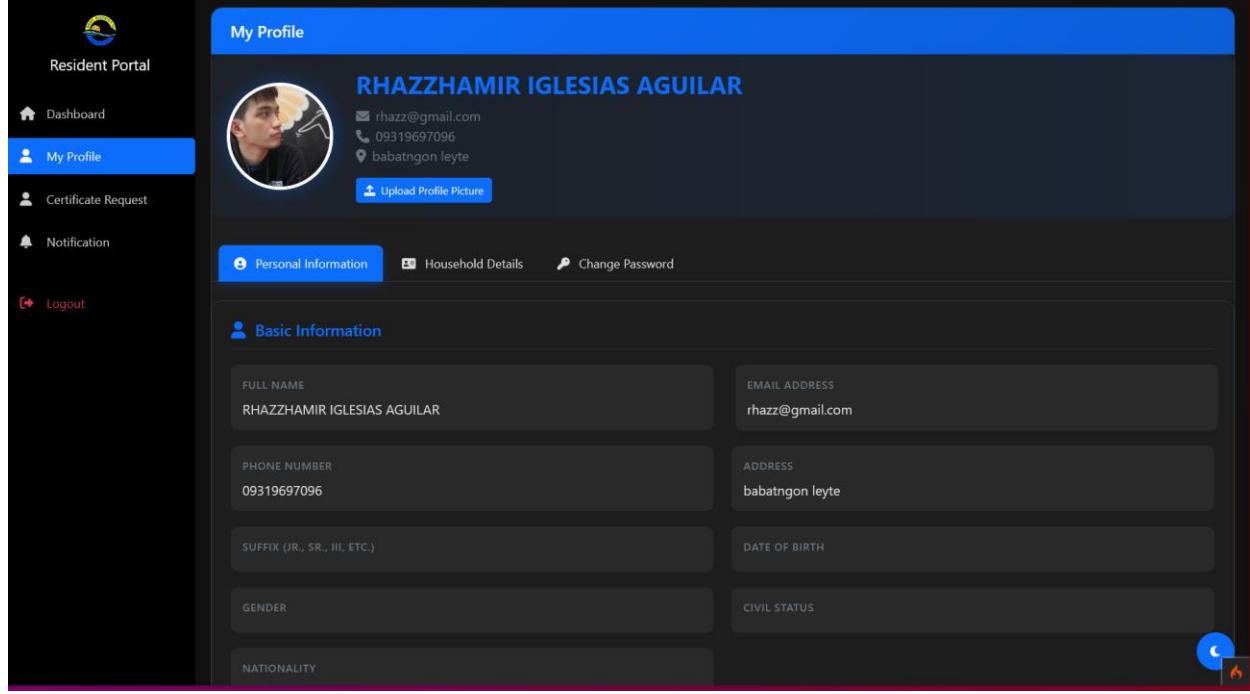
The screenshot shows a Resident Login form on the same dark-themed web page. At the top right are three buttons: 'Admin Login' (with a user icon), 'Back to Home' (with a house icon), and a gear icon. The main title 'Resident Login' is centered at the top, with the subtitle 'Sign in to your resident account' just below it. The form fields are arranged in rows: 'Email Address' (placeholder 'Enter your email') and 'Password' (placeholder 'Enter your password'). Below these is a checkbox labeled 'Remember me'. A large blue 'Login' button is centered at the bottom. At the bottom of the form is a link 'Don't have an account? Register here'.

After logging in, the user sees a control panel with many tools. They can manage and update the information.



Profile Management

- User can add or change their information and also the profile



Certificate Request

- Resident can request a any types certificate and wait to approved from the admin

Certificate form

The screenshot shows the Resident Portal interface. On the left is a sidebar with a logo, 'Resident Portal' title, and links for Dashboard, My Profile, Certificate Request (which is highlighted in blue), and Notification. Below the sidebar is a URL bar with 'http://localhost:8080/dashboard/resident'. The main content area has a blue header 'Certificate Request'. Below it, 'Select Certificate Type' is shown with four options: 'Barangay Clearance' (official document certifying good standing), 'Certificate of Indigency' (document proving financial status for assistance programs), 'Certificate of Residency' (proof of current residence), and 'Business Permit' (required for operating a business). Underneath this is a section for 'Request Details' with fields for 'Purpose of Request' (a text input placeholder 'Please specify the purpose of your certificate request'), 'Preferred Pick-up Date' (a date picker set to 'mm / dd / yyyy'), 'Number of Copies' (a dropdown set to '1'), and a 'Submit Request' button.

Notification when the certificate is approved

The screenshot shows the Resident Portal interface with the 'Notification' link in the sidebar highlighted in blue. The main content area has a blue header 'Notifications'. It lists four notifications:

- Certificate Request Approved**: Your request for Business has been approved. (Approved status, timestamp May 14, 2025 03:08 PM)
- Certificate Request Pending**: Your request for Residency has been pending. (Pending status, timestamp May 14, 2025 03:08 PM)
- Certificate Request Approved**: Your request for Indigency has been approved. (Approved status, timestamp May 14, 2025 03:08 PM)
- Certificate Request Pending**: Your request for Barangay Clearance has been pending. (Pending status, timestamp May 14, 2025 03:08 PM)

Below the notifications is a URL bar with 'http://localhost:8080/dashboard/resident'.

Conclusion

- Display a list of residents
- Allow adding new residents
- Allow editing resident details
- Allow deleting residents
- Allow adding new information
- Title Proposal Objective