***Simple CRUD with Codeigniter and MySQL***

***Name:*** Shawn Michael A. Sudaria ***BSIT-3B***

**EVSU Research Archive**

**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:

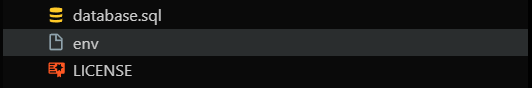
* A computer or laptop
* **Laragon** (includes Apache + MySQL)
* **CodeIgniter 4**
* A browser (like Chrome)

**Step 2: Open Laragon and Start Services**

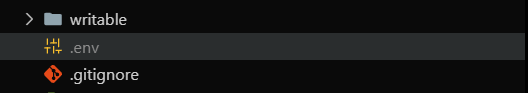
* Open **Laragon Control Panel**
* Start both **Apache** and **MySQL**
* Open vscode

**Step 3: Connect CodeIgniter to the Database**

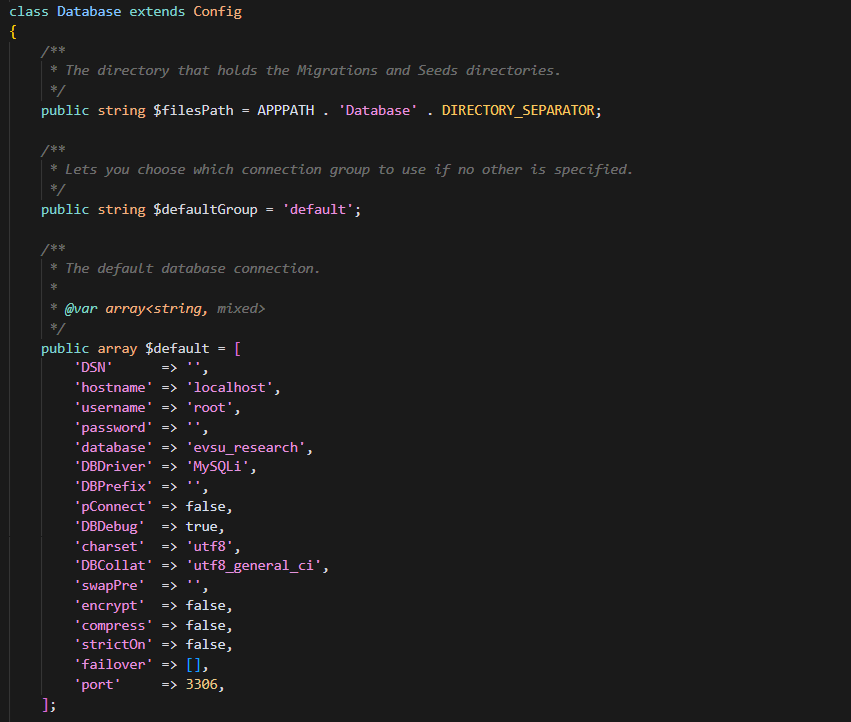
1. Inside your CodeIgniter project folder, look for a file named env



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

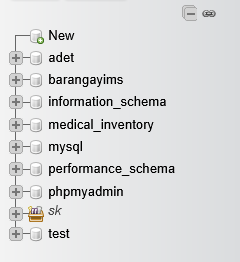


1. Go to the app/Config/Database.php and make sure it follows this structure

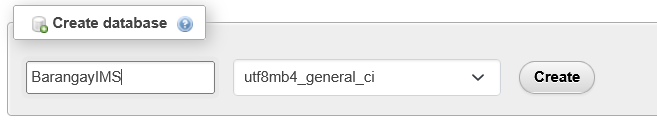


**Step 5: Create the Database and Table**

1. ***Create the Database:***
   * Go to http://localhost/phpmyadmin (make sure XAMPP is running).
   * Click on New to create a database.



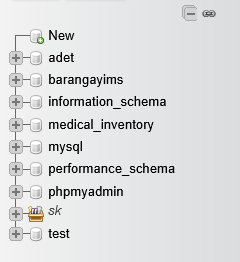
* + Name the database evsu\_research and click Create.



1. ***Create a Table:***

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

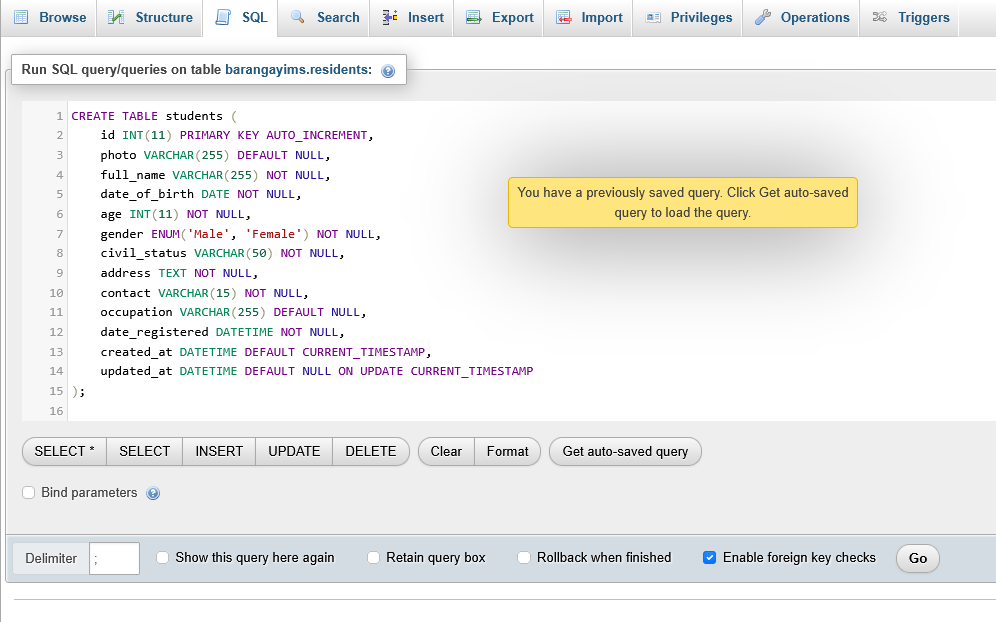
* + In phpMyAdmin, select the evsu\_research database.



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



* + Paste the following SQL code to create a residents table:

****

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

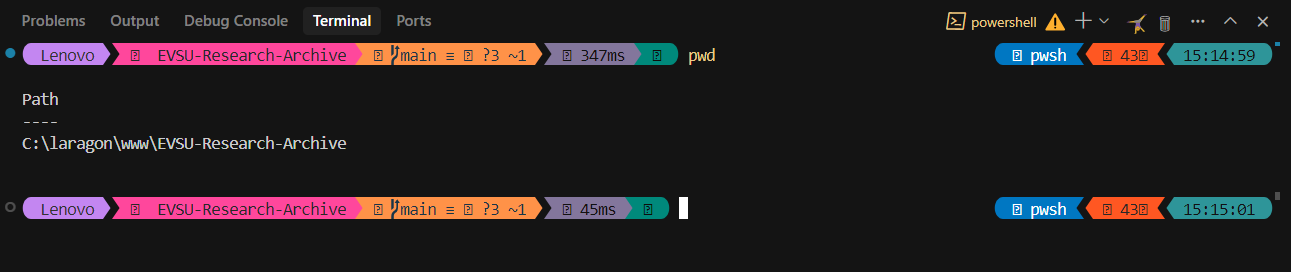
***Step 5 : Create a Model in CodeIgniter***

**What is a Model?**

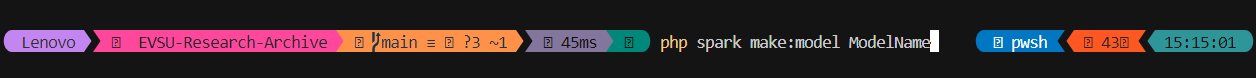
A **Model** in CodeIgniter is a file that connects your application to the **database**.

* 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).

**1. Open your terminal in the project root**



Use this command to generate a model:



Use these commands to make the models for this project:

*php spark make:model ResearchPaperModel*

Here’s how your PublicationModel should look based on your table structure:



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

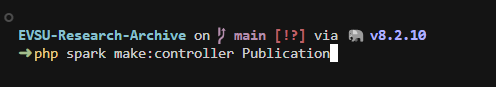
**What is a Controller?**

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**.

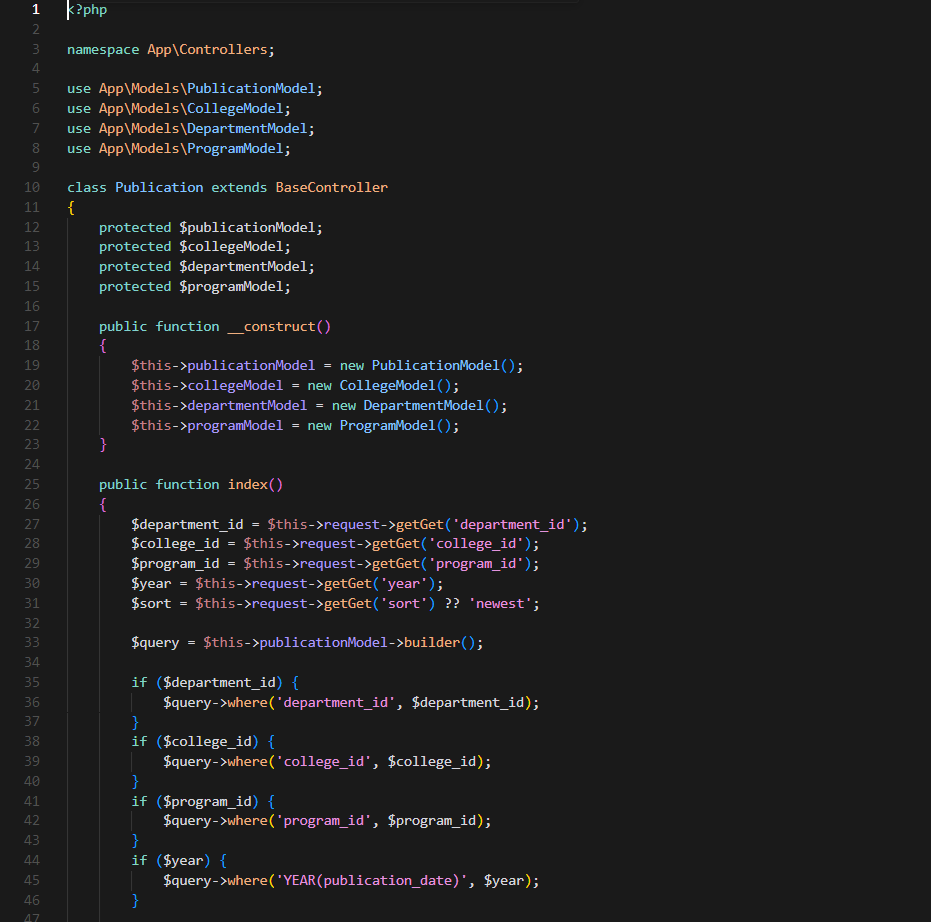
**How to Create a Controller**

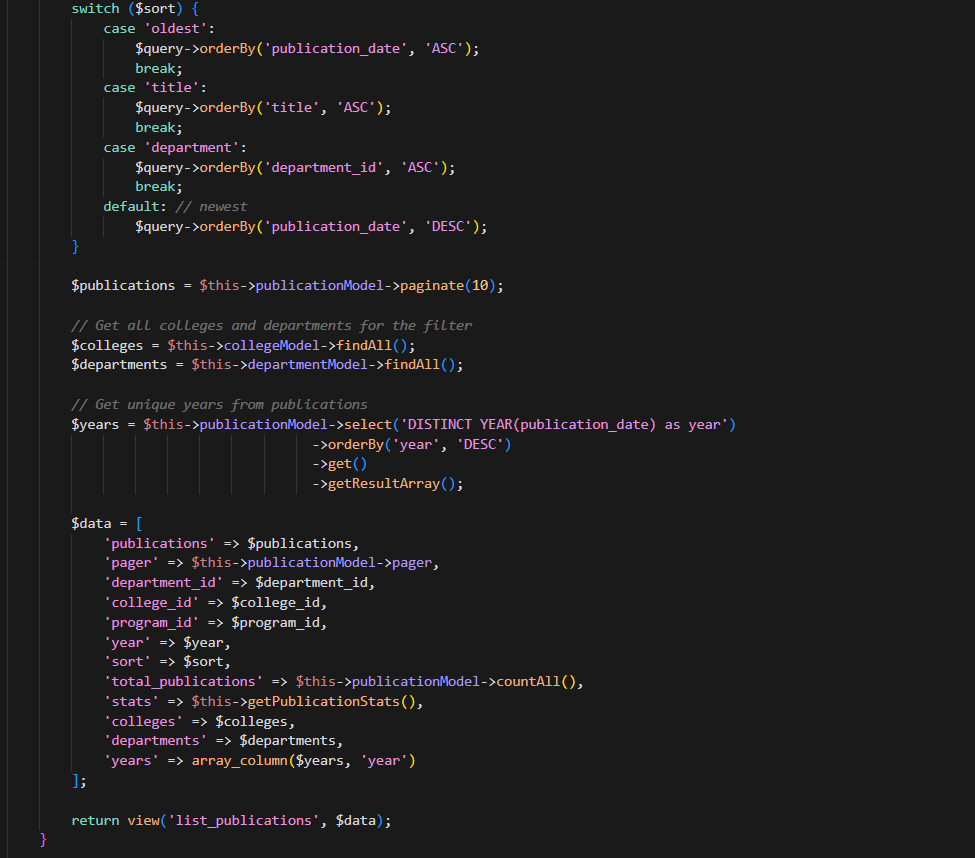
Use the terminal in your project directory and run:

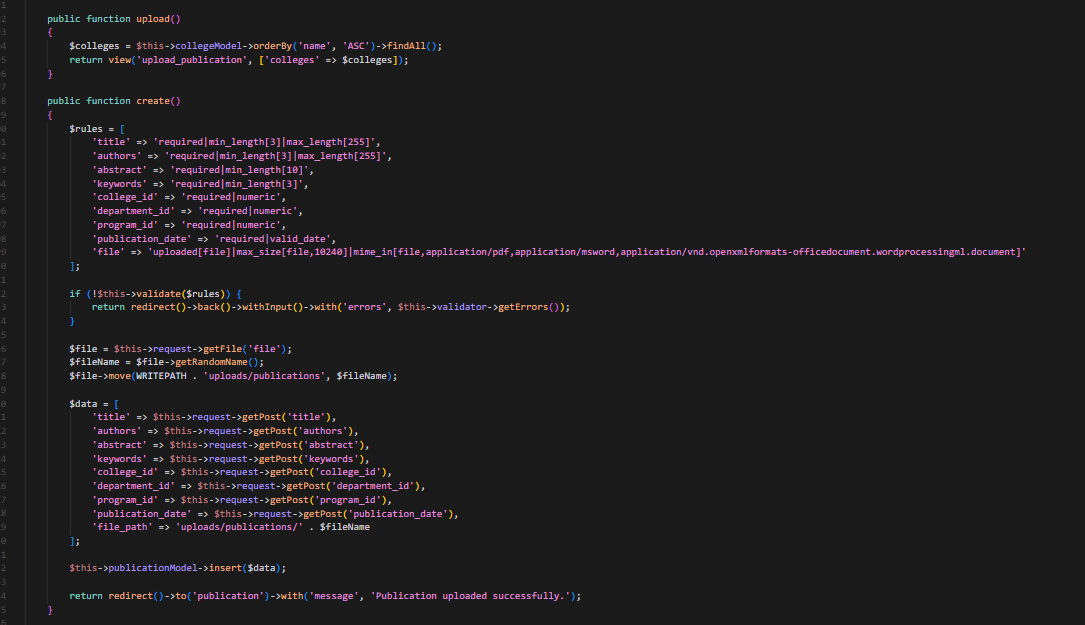


**Edit the Controller**

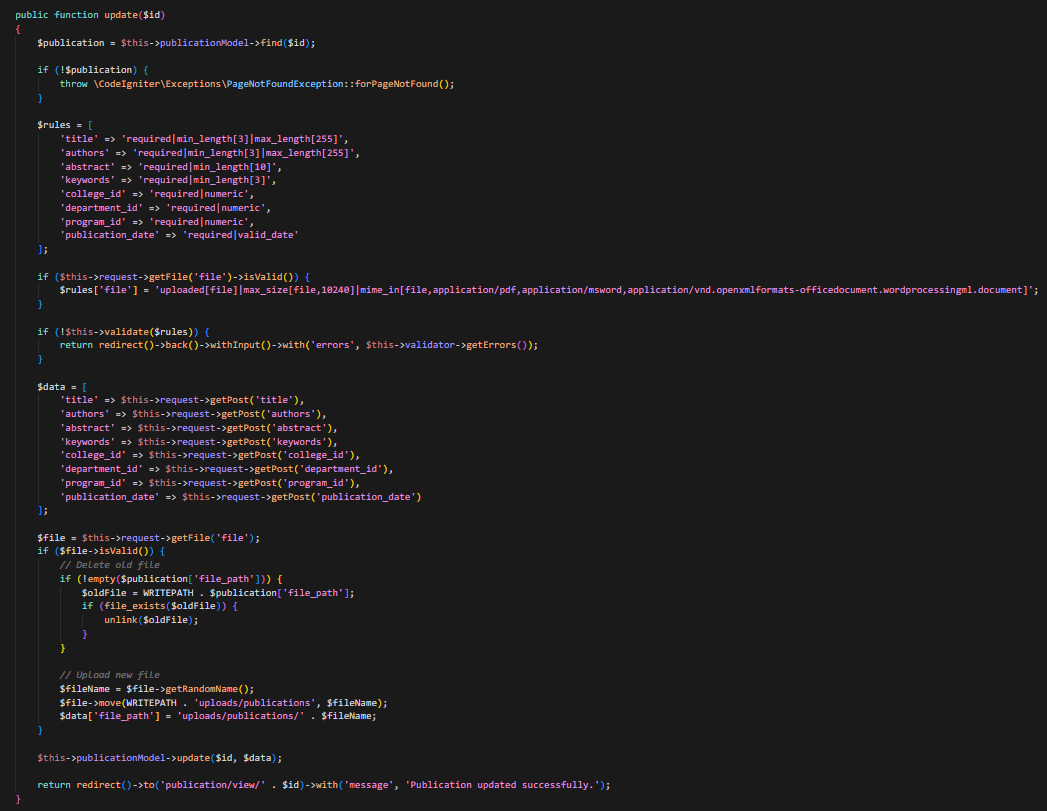
Open the Publication.php file and start with this basic setup:

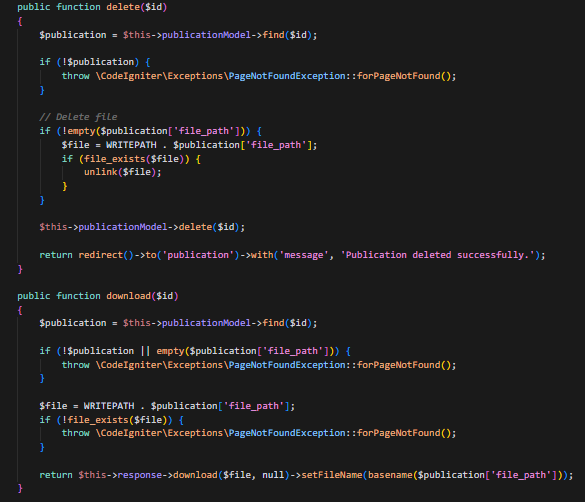












**What this Controller does:**

* upload() – for uploading the document
* create() – creating an entry in the table for the metadata of the file
* view() – viewing the publication
* update() – updating the metadata of the publication
* update() – update publication info
* delete() – delete a publication
* download() – for the ability to download the publication for authorized users

**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.

**How to Add Routes**

Open this file in your project:



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



**What These Routes Do:**

* */publications (GET)* - Displays a list of all publications.
* */publications/upload (GET)* - Shows the form to upload a new publication.
* */publications/create (GET)* - Displays the form to create a new publication record.
* */publications/create (POST)* - Processes data submitted from the new publication form.
* */publications/view/(:num) (GET)* - Views a specific publication by its ID.
* */publications/edit/(:num) (GET)* - Displays the form to edit a specific publication by its ID.
* */publications/edit/(:num) (POST)* - Processes data from the edit publication form for a specific ID.
* /publications/delete/(:num) (GET) - Deletes a specific publication by its ID. (Using GET for deletion is not recommended).
* /publications/download/(:num) (GET) - Downloads a file for a specific publication by its ID.
* /publications/search (GET) - Displays the search form or initial results.
* /publications/search (POST) - Processes search form submissions.

The routes might appear to have duplicates (like /publication vs /publications if the singular route exists) because each unique URI pattern defined creates a distinct access point, even if multiple URIs point to the same controller action. This can be done for flexibility, allowing users to reach the same content through slightly different URLs.

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.

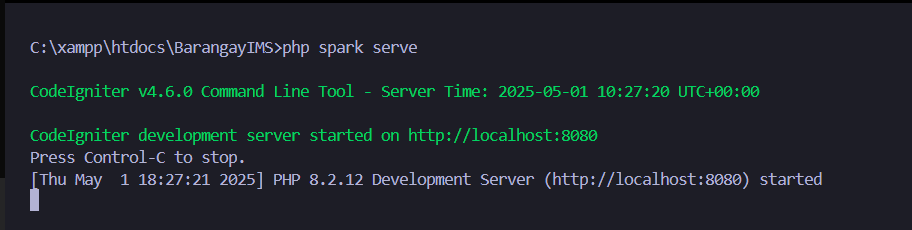
**Where to Place the Views**

Save your view files in this folder:

*apps/views/*

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

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

******

***Open your browser and go to*** [***http://localhost:8080***](http://localhost:8080)***. You should see the CodeIgniter welcome page.***

A screenshot of a computer

AI-generated content may be incorrect.

**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)***

***After (Modified to fit our Resident CRUD system)***

***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.*

***Resident Management***

* *Admin can view resident information, remove inactive users.*

***Official Management***

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

***Add official***

***A screenshot of a computer

AI-generated content may be incorrect.***

***List of official***

***A screenshot of a computer

AI-generated content may be incorrect.***

***Budget Management***

* Module for tracking and controlling barangay budget allocations.

***Add budget***

***Record Budget***

***Legal***

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

***Certificate Management***

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

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

***After register need to login***

*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***

***Notification when the certificate is approved***

***Conclusion***

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