

**Guru Nanak Dev
Engineering
College, Ludhiana**



Department of Information Technology

Mini Project Report

Advanced Web Technologies Laboratory

(AWT) Lpeit-102

(5th Sem 2025 batch)

Student Details

Arshpreet Singh

2104478 (URN)

2121021(CRN)

I.T A1

Submitted to:

Dr. Akshay Girdhar

Abstract

This initiative seeks to transform the way educational institutions handle student information by developing a cutting-edge web-based application using the CodeIgniter framework. The main goal is to establish a highly effective system that seamlessly manages student details through an easy-to-use interface. Focusing on the implementation of CRUD operations (Create, Read, Update, Delete), the project aims to create a secure, scalable, and intuitive platform for storing and managing comprehensive student records.

In the realm of educational information systems, this project aims to reshape the management of student data through the creation of an advanced web-based application. It revolves around leveraging the capabilities of the CodeIgniter framework to devise an innovative solution tailored to the evolving needs of educational institutions. The primary objective is to construct a comprehensive platform capable of efficiently handling student details while ensuring data integrity, security, and user-friendliness.

In today's educational landscape, the effective management of student information plays a crucial role in enhancing administrative efficiency and academic progress. Conventional methods relying on manual record-keeping, disparate spreadsheets, and fragmented databases present significant challenges, including data redundancy, integrity issues, limited accessibility, and a lack of adaptability to evolving educational requirements. Therefore, this project endeavours to create a groundbreaking solution that streamlines the management of student details.

In the culmination of this endeavour, I extend my heartfelt gratitude to my esteemed teacher, whose unwavering guidance and mentorship have been instrumental throughout the development of project. The project has been an enlightening journey under their tutelage, and their invaluable insights have shaped the project's success. I express sincere appreciation to the Head of the Information Technology Department. for fostering an environment conducive to innovation.

A special note of thanks goes to the Principal of Guru Nanak Dev Engineering College, Ludhiana, for providing the wonderful opportunity to engage in this project. The encouragement and support from the leadership have been a source of motivation, empowering us to explore and implement innovative solutions in the realm of Advanced Web Technologies.

This project has not only enriched our technical skills but has also instilled a deep appreciation for the collaborative and visionary approach encouraged by the institution. With sincere thanks, we acknowledge the contributions of our teacher, HOD, and Principal for their pivotal roles in shaping this educational journey.

Table of Contents

1. Abstract	2
2. Introduction.....	4
2.1. Background.....	4
2.2. Objectives	4
2.3. Why CodeIgniter	4
2.4. Project Attributes.....	4
3. Technologies Used.....	5
4. Methodology	5
4.1. System Architecture.....	5
4.1.1.Model (M):.....	5
4.1.2.View (V):.....	5
4.1.3.Controller (C):.....	5
4.1.4.Interaction Flow:	6
4.1.5.Benefits of MVC:	6
4.2. Database Design	6
4.2.1.Course Table.....	6
4.2.2.Log Table	7
4.3. Features	8
4.3.1.CREATE	8
4.3.2.READ	9
4.3.3.UPDATE	9
4.3.4.DELETE	9
4.3.5.Step-wise approach to create a CodeIgniter based Login form:.....	9
5. Output and Results.....	12
5.1. Create Operation (Course Registration).....	12
5.2. Read Operation (Course Listing and Details)	13
5.3. Update Operation (Edit Course Details)	13
5.4. Delete Operation (Delete User)	14
5.5. View Log Table	15
6. The Log table provides a comprehensive overview of modifications that have been implemented. Users can access this table to review and examine alterations made within the system.....	15
7. Conclusion	15
8. References	15
9. GitHub Repository Link.....	15

Introduction

Background

In the rapidly evolving field of education management, efficient systems are essential for handling and organizing course-related information. Our project, the CodeIgniter Study Scheme CRUD Application, addresses this need by providing a robust platform for managing study schemes within an educational institution.

Objectives

The primary objective of this project is to create a user-friendly web application that allows administrators and authorized users to perform CRUD operations on a study scheme table. The study scheme represents a fundamental aspect of course management, encompassing details such as course names, credits, and associated information.

Why CodeIgniter

The choice of CodeIgniter as the web development framework stems from its lightweight nature, excellent documentation, and adherence to the Model-View-Controller (MVC) design pattern. This ensures a structured and modular codebase, promoting code maintainability and scalability.

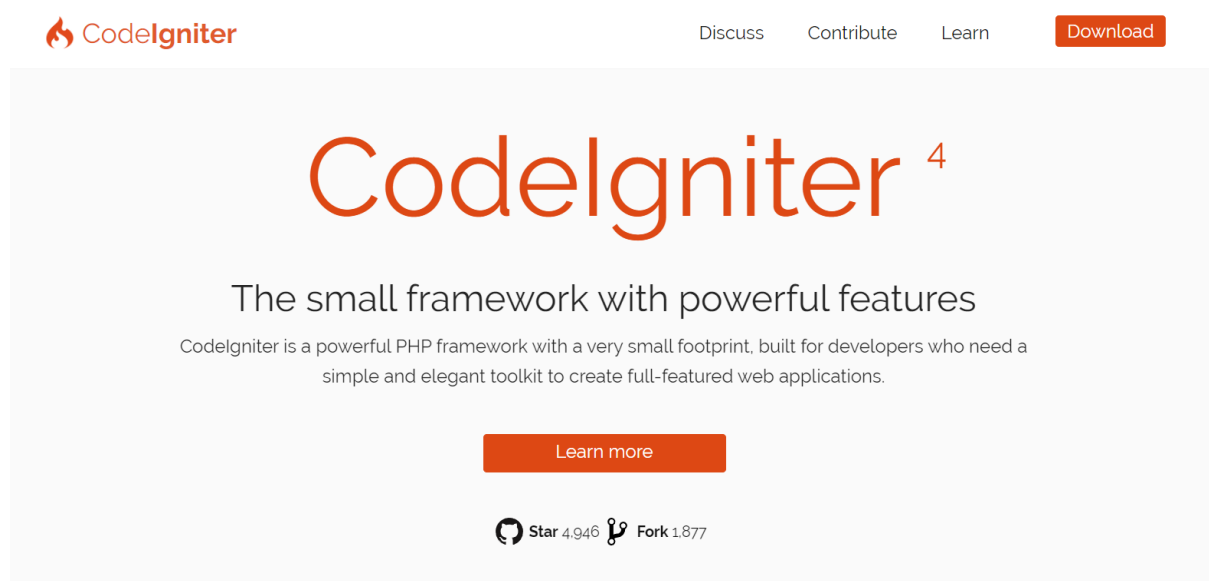


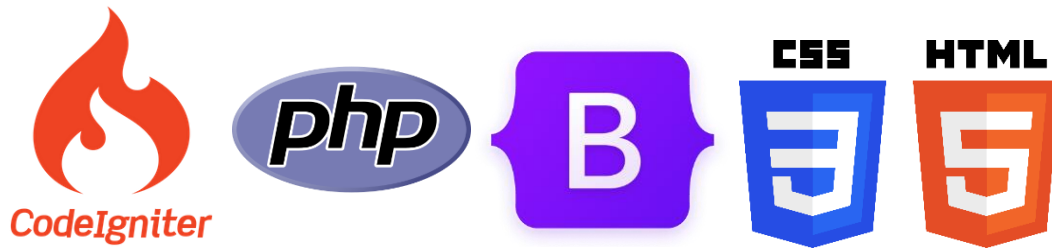
Figure 1 CodeIgniter official website homepage

Project Attributes

- Implementation of CRUD operation
- Log table to store changes made with timestamp
- Easy to Understand Database Design
- User-Friendly interface
- Responsive Design

Technologies Used

- CodeIgniter Framework: Chosen for its MVC architecture, simplicity, and scalability.
- PHP: Server-side scripting language for backend functionality.
- Bootstrap: For responsiveness and mobile-first approach.
- MySQL: Database management system for storing student information.
- HTML, CSS: Frontend development for a user-friendly interface.



Methodology

The development methodology employed in creating the CodeIgniter Study Scheme CRUD Application follows a systematic and iterative approach, ensuring the delivery of a reliable and feature-rich solution. The methodology can be outlined as follows:

System Architecture

Model (M):

The Model component serves as the data layer, responsible for managing the application's data and business logic. In our project, the study scheme table and log table are represented as models. The `student_model` encapsulates the database interactions related to study schemes, while the `log_model` handles logging functionality. This clear separation of concerns ensures that data manipulation and storage are centralized within the model, promoting code reusability and scalability.

View (V):

The View component focuses on the presentation layer, handling the user interface and user experience. In our application, the study scheme views are designed using Bootstrap for a responsive and visually appealing layout. These views interact with the controller to retrieve and display data to the user. By isolating the presentation layer, the application achieves greater flexibility, allowing for easy modification and customization without impacting the underlying logic.

Controller (C):

The Controller acts as an intermediary between the Model and the View, orchestrating the flow of data and user interactions. In our project, the `student_controller` manages the CRUD operations on the study scheme table. It receives user inputs from the view, communicates with the `student_model` to perform database operations, and then updates the view with the results. This division of responsibilities ensures a well-organized and maintainable codebase, as changes in one component do not necessarily affect the others.

Interaction Flow:

The interaction within the MVC architecture follows a systematic flow. User inputs from the view trigger actions in the controller, which then invokes the corresponding methods in the model for data manipulation. The model updates the database and communicates the results back to the controller, which, in turn, updates the view to reflect the changes. This cyclical process ensures a separation of concerns, making the application more modular and adaptable to future enhancements.

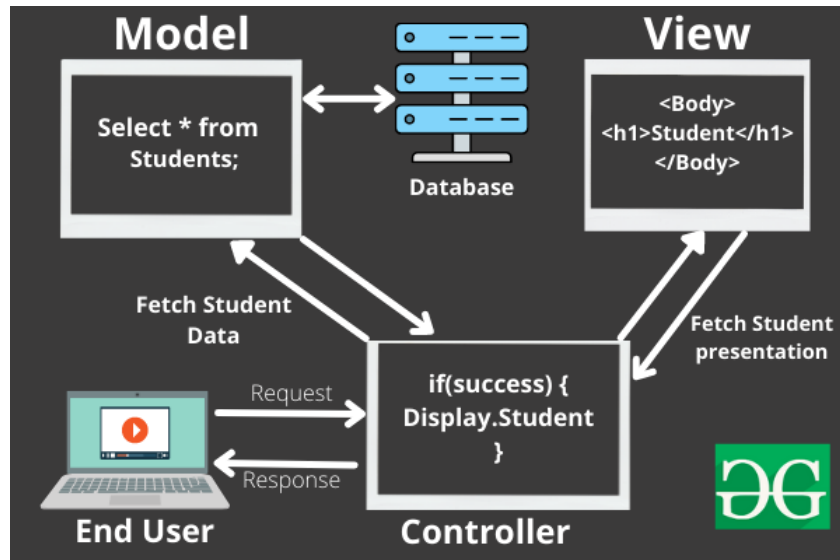


Figure 2 Flow in MVC architecture

Benefits of MVC:

The adoption of MVC in our project offers several advantages. It enhances code maintainability by isolating concerns into distinct components, making it easier to understand, update, and troubleshoot. The modular nature of MVC facilitates collaboration among developers, as different team members can work on separate components simultaneously. Additionally, it promotes code reusability, scalability, and provides a foundation for a systematic testing approach.

In conclusion, the incorporation of the MVC architecture in the CodeIgniter Study Scheme CRUD Application results in a well-organized, flexible, and maintainable system. This design pattern not only facilitates the development process but also ensures the long-term sustainability and adaptability of the application to evolving requirements.

Database Design

Here is the description of the structure of the study scheme table and the log table. Including information about the fields, data types, and relationships.

Course Table

The Course Table serves as a central repository for storing information related to study schemes. Here is an overview of its structure:

```
protected $table = 'course_table';

protected $primaryKey = 'id';

protected $useTimestamps = true; // Enable automatic timestamps

protected $createdField = 'created_at';
```

```
protected $updatedField = 'updated_at';
```

```
protected $allowedFields = ['course_code', 'branch_code', 'scheme_code',  
'leet_status', 'semester', '6_month_training', 'AICTE_RE', 'course_type', 'subject_title', 'm_code', 'theory/  
practical', 'elective_status', 'credits', 'internal_marks', 'external_marks', 'cbs_status', 'created_at', 'update  
d_at'];
```

- **id**: A unique identifier for each course, serving as the primary key.
- **course_name**: Represents the name or title of the course.
- **credits**: Indicates the number of credits associated with the course.
- **description**: A field to provide additional information or a description of the course.
- **created_at**: Automatically records the timestamp when a course entry is created.
- **updated_at**: Automatically updates the timestamp whenever there is a modification to the course entry.


#	Name	Type	Collation	Attribut
<input type="checkbox"/> 1	id 	int(11)		
<input type="checkbox"/> 2	course_code	int(11)		
<input type="checkbox"/> 3	branch_code	int(11)		
<input type="checkbox"/> 4	scheme_code	int(11)		
<input type="checkbox"/> 5	leet_status	enum('leet', 'non-leet')	utf8mb4_general_ci	
<input type="checkbox"/> 6	semester	enum('1', '2', '3', '4', '5', '6', '7', '8')	utf8mb4_general_ci	
<input type="checkbox"/> 7	6_month_training	enum('Yes', 'No')	utf8mb4_general_ci	
<input type="checkbox"/> 8	AICTE_RE	int(11)		
<input type="checkbox"/> 9	course_type	enum('Full-time', 'Part-time')	utf8mb4_general_ci	
<input type="checkbox"/> 10	subject_title	varchar(225)	utf8mb4_general_ci	
<input type="checkbox"/> 11	m_code	int(11)		
<input type="checkbox"/> 12	theory/practical	enum('Theory', 'Practical')	utf8mb4_general_ci	
<input type="checkbox"/> 13	elective_status	enum('Yes', 'No')	utf8mb4_general_ci	
<input type="checkbox"/> 14	credits	enum('0', '2', '3', '4')	utf8mb4_general_ci	
<input type="checkbox"/> 15	internal_marks	int(11)		
<input type="checkbox"/> 16	external_marks	int(11)		
<input type="checkbox"/> 17	cbs_status	enum('Yes', 'No')	utf8mb4_general_ci	
<input type="checkbox"/> 18	created_at	timestamp		
<input type="checkbox"/> 19	updated_at	timestamp		

Figure 3 Course table structure

Log Table

The Log Table is designed to capture changes made to the Course Table, facilitating a comprehensive audit trail. Here is the structure of the Log Table:

```
protected $table = 'log_table';
```

```
protected $primaryKey = 'id';
```

```
protected $updatedField = 'updated_at';
protected $allowedFields = ['old_value', 'updated_at', 'created_at'];
protected $useTimestamps = true;
```


#	Name	Type	Collation
<input type="checkbox"/> 1	id 	int(11)	
<input type="checkbox"/> 2	old_value	mediumtext	utf8mb4_general_ci
<input type="checkbox"/> 3	updated_at	timestamp	
<input type="checkbox"/> 4	created_at	timestamp	

Figure 4 Log Table Structure

Features

CRUD is the acronym for **CREATE**, **READ**, **UPDATE** and **DELETE**. These terms describe the four essential operations for creating and managing persistent data elements, mainly in relational and NoSQL databases. CRUD is extensively used in database applications. This includes Relational Database Management Systems (RDBMS) like Oracle, MySQL, and PostgreSQL. It also includes NoSQL databases like MongoDB, Apache Cassandra, and AWS DynamoDB.



Figure 5 Four basic CRUD operations

CREATE

The Create operation adds a new record to a database. In RDBMS, a database table row is referred to as a record, while columns are called attributes or fields. The CREATE operation adds one or more new records with distinct field values in a table.

READ

Read returns records (or documents or items) from a database table (or collection or bucket) based on some search criteria. The READ operation can return all records and some or all fields.

UPDATE

Update is used to modify existing records in the database. For example, this can be the change of address in a customer database or price change in a product database. Similar to READ, UPDATES can be applied across all records or only a few, based on criteria.

DELETE

Delete operations allow the user to remove records from the database. A hard delete removes the record altogether, while a soft delete flags the record but leaves it in place. For example, this is important in payroll where employment records need to be maintained even after an employee has left the company.

Step-wise approach to create a CodeIgniter based Login form:

Here are the steps related specifically to creating the Model, View, and Controller for a simple login page web application with user authentication and CRUD operations using the CodeIgniter Framework:

Step 1: Install CodeIgniter

Download and extract CodeIgniter into your web server's root directory.

Step 2: Configure Database

Open ``application/config/database.php`` and configure the database settings.

Step 3: Create Database Tables

-Create a ``users`` table in your database with fields like ``id``, ``username``, ``password``, etc.

Step 4: Create Model for Courses

Create a new model file in ``application/models`` (e.g., ``Course_model.php``).

Define functions for CRUD operations (create, read, update, delete) and user authentication.

Course_Model.php

```
<?php
namespace App\Models;
use CodeIgniter\Model;

class student_model extends Model{
    protected $table = 'student_table';
    protected $primaryKey = 'id';
```

```

        protected $useTimestamps = true;

        protected $createdField = 'created_at';

        protected $updatedField = 'updated_at';

        protected $allowedFields = ['course_code', 'branch_code', 'scheme_code',
            'leet_status', 'semester', '6_month_training', 'AICTE_RE', 'course_type',
            'subject_title', 'm_code', 'theory/practical', 'elective_status', 'credits', 'internal_marks',
            'external_marks', 'cbs_status', 'created_at', 'updated_at']; >

```

Step 5: Create Controller for Authentication

- Create a new controller file in `application/controllers` (e.g., `Home.php`).
- Define functions for login, logout, and user registration.
- Load the user model in the constructor for database operations.

Course_controller.php

```

<?php

namespace App\Controllers;

use App\Models\student_model;

use App\Models\log_model;

class student_controller extends BaseController{

    public function index(): string

    {return view('create_student');}
}

```

C - Create

```

        public function insert()

        { $data = [

            'course_code' => $this->request->getVar('course_code'),
            'branch_code' => $this->request->getVar('branch_code'),
            'scheme_code' => $this->request->getVar('scheme_code'),
            'leet_status' => $this->request->getVar('leet_status'),
            'semester' => $this->request->getVar('semester'),.....
        ]
    }

```

R - Read

```
public function show(){  
    $model = new Course_Model();  
    $data['users'] = $model->findAll();  
    return view('show',$data);}
```

U - Update

```
$id = $this->request->getVar('id');  
$model = new student_model();  
$model->update($id,$data);  
return redirect()->to( base_url  
('student_controller/show_student' ) );
```

D - Delete

```
public function delete($id = null){  
    $model = new student_model();  
    $data['course'] = $model->where('id', $id)->delete();  
    return redirect()->to(base_url  
('student_controller/show_student'));
```

Step 6: Create Views

- Create views in `application/views` for login, registration, user listing, user details, etc.
- Design forms using HTML and integrate with CodeIgniter's form helper.
- Utilize CodeIgniter's form helper for form creation, validation, and error handling.

Step 7: Secure Routes

- Utilize CodeIgniter's route configuration to restrict access to certain pages to authenticated users.

Output and Results

Create Operation (Course Registration)

Screenshot 1: Course Registration(study scheme Form) and Success

Study Scheme Form

Course Code:

Branch Code:

Scheme Code:

LEET Status:

Semester:

6-Month Training:

☒ Yes
☐ No

AICTE_RE:


Course Type:

Data sent successfully...

This screenshot captures the course registration process. The registration form is displayed, allowing users to input their details. Upon successful registration, a confirmation message or success page is presented.

Read Operation (Course Listing and Details)

Screenshot 2: Course Listing and Details



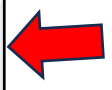
Scheme Code	LEET Status	Semester	6-Month Training	AICTE_RE	Course Type	Subject Title	M Code	Theory/Practical
32	leet	1	Yes	323	Full-time	23fdsfs	23	Theory
34	leet	1	Yes	324	Full-time	dsf	324	Theory
45212	leet	1	Yes	545	Full-time	wadads	232	Theory

This combined screenshot showcases the user listing page along with user details. Users can view a list of registered users, and by clicking on a user, they can access detailed information about that specific user.

Update Operation (Edit Course Details)

Screenshot 3: Edit Course Form and Success

Internal Marks	External Marks	CBS Status	Created At	Updated At	Delete	Edit
323	32	Yes	2023-12-17 08:53:02	2023-12-17 08:53:02	Delete	Edit
432	234	Yes	2023-12-17 09:25:22	2023-12-17 09:33:51	Delete	Edit



Update Form

Course Code:

Branch Code:

Scheme Code:

LEET Status:

Semester:

This screenshot covers the course details editing process. Users with appropriate permissions can access the edit user form, modify details, and receive a success message upon updating.

Delete Operation (Delete User)

Screenshot 4: Delete Confirmation and Success

Before:






















Subject Title	M Code	Theory/Practical	Elective Status	Credits	Internal Marks	External Marks	CBS Status	Created At	Updated At	Delete	Edit
23fdsfs	23	Theory	Yes	3	323	32	Yes	2023-12-17 08:53:02	2023-12-17 08:53:02	Delete	Edit
dsf	324	Theory	Yes	0	432	234	Yes	2023-12-17 09:25:22	2023-12-17 09:33:51	Delete	Edit



After:

Subject Title	M Code	Theory/Practical	Elective Status	Credits	Internal Marks	External Marks	CBS Status	Created At	Updated At	Delete	Edit
23fdsfs	23	Theory	Yes	3	323	32	Yes	2023-12-17 08:53:02	2023-12-17 08:53:02	Delete	Edit

View Log Table

				id	old_value	updated_at	created_at
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	<code>{"id":"2","course_code":"23","br</code>	2023-12-17 09:04:58	2023-12-17 09:04:58
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	<code>{"id":"2","course_code":"23","branch_code":"323","...</code>	2023-12-17 09:29:02	2023-12-17 09:29:02
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	<code>{"id":"3","course_code":"23","branch_code":"213","...</code>	2023-12-17 09:29:07	2023-12-17 09:29:07
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	<code>{"id":"3","course_code":"23","branch_code":"213","...</code>	2023-12-17 09:32:18	2023-12-17 09:32:18
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	<code>{"id":"3","course_code":"23","branch_code":"213","...</code>	2023-12-17 09:33:16	2023-12-17 09:33:16
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	<code>{"id":"3","course_code":"23","branch_code":"213","...</code>	2023-12-17 09:33:49	2023-12-17 09:33:49
<input type="checkbox"/>	 Edit	 Copy	 Delete	7	<code>{"id":"2","course_code":"23","branch_code":"323","...</code>	2023-12-18 01:33:15	2023-12-18 01:33:15

The Log table provides a comprehensive overview of modifications that have been implemented. Users can access this table to review and examine alterations made within the system.

Conclusion

The project serves as a testament to the effective execution of a student management system utilizing the CodeIgniter framework, facilitating the streamlined management of comprehensive student information. This application not only ensures the secure handling of student records but also offers administrators an intuitive interface for seamless control.

Future iterations of the system could incorporate enhancements to augment its functionality. These may include the implementation of robust user authentication mechanisms to fortify data security, the integration of advanced search functionalities to expedite information retrieval, and the addition of report generation capabilities for comprehensive data analysis.

By continually refining and expanding the features of this student management system, the application has the potential to evolve into a comprehensive and versatile tool that meets the dynamic needs of educational institutions. The commitment to ongoing improvement underscores the project's dedication to providing administrators with a powerful and user-friendly platform for effective student data management.

References

- **CodeIgniter Documentation:** https://codeigniter.com/user_guide/
- **Bootstrap Documentation:** <https://getbootstrap.com/>
- **Builtwithphp:** <https://www.builtwithphp.com/how-to-create-codeigniter-3-crud-app>

GitHub Repository Link

GitHub repo : https://github.com/arshpreet8051/AWT_mini_project.git