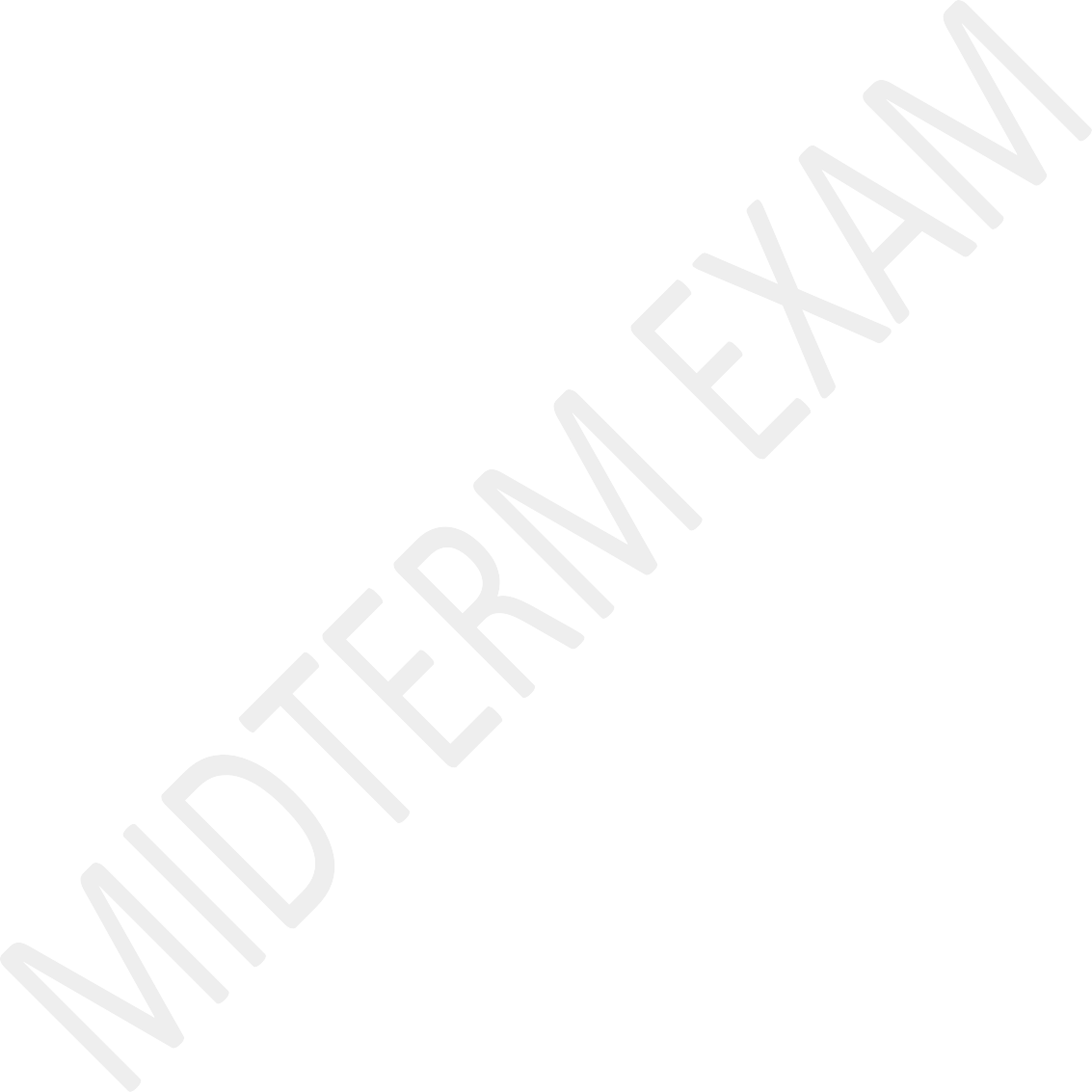
# General Instruction

* You are required to develop features for an Online Student Portal using the CodeIgniter 4 framework.
* Use the existing project structure you have been developing in your laboratory exercises.
* You may use your previous code (e.g., authentication system) as a starting point.
* Version Control is Mandatory. You must make at least 3 meaningful commits during the exam, with descriptive messages. Your commit history will be checked.
* Write clean, readable, and well-structured code. Comments are encouraged for clarity.
* The use of Bootstrap for basic styling is expected, but not the primary focus.

# Problem Scenario

The university wants to enhance its Online Student Portal. Your task is to implement new features and security measures. The system has three user roles: Student, Teacher, and Admin. You will be building upon the authentication system developed in previous labs.

Examination Tasks

# Task 1: Announcements Module Scenario

The admin needs a way to post announcements that are visible to all logged-in users.

# Task:

* + Create a new controller named **Announcement.php**.
  + Inside the controller, create a method named **index()**.
  + The **index()** method should fetch all announcements from a database table (you will create this table in Task 2) and pass them to a view.
  + Create a view file **announcements.php** that displays the announcements in a list (title, content, date posted).
  + Configure a route so that accessing **/announcements** triggers the **index()**

method in the **Announcement** controller.

# Expected Output

* + - An accessible page at **/announcements** that lists announcement data (even if the table is empty).

# Task 2: Database Schema and Data Population (25 points) Scenario

The portal needs a new table to store announcements and a table for student grades.

# Task:

* Generate and write a migration file named **CreateAnnouncementsTable**

that creates an **announcements** table with the following fields:

* + id (Primary Key, Auto Increment)
  + title (VARCHAR)
  + content (TEXT)
  + created\_at (DATETIME)
* Create a model named **AnnouncementModel.php** that is properly configured to interact with the **announcements** table.
* In the **Announcement** controller's **index()** method, use the **AnnouncementModel** to fetch all announcements, ordered by **created\_at** in descending order (newest first).
* Create a seeder that inserts at least two sample announcements into the

**announcements** table. Run this seeder.

# Evaluation Criteria:

* Correct migration schema and successful execution. (10 pts)
* Properly defined Model. (5 pts)
* Correct use of the Model in the Controller to fetch and order data. (5 pts)
* Successful creation and execution of the seeder. (5 pts)

# Task 3 Enhanced Authentication and Role-Based Redirection (30 points) Scenario

The current login system redirects all users to a single dashboard. This needs to be improved for a better user experience and security.

# Task:

1. Modify the **login()** method in your **Auth** controller.
2. After a successful login, instead of redirecting everyone to **/dashboard**, redirect users based on their role:
   * Students should be redirected to **/announcements**.
   * Teachers should be redirected to a **/teacher/dashboard**.
   * Admins should be redirected to a **/admin/dashboard**.
3. Create the **dashboard()** method inside a new **Teacher.php** controller. For now, this method should simply load a view **teacher\_dashboard.php** with the text **Welcome, Teacher!**.
4. Create the **dashboard()** method inside a new **Admin.php** controller. For now, this method should simply load the view **admin\_dashboard.php** with the text **Welcome, Admin!.**
5. Configure routes for **/teacher/dashboard** and **/admin/dashboard**.

# Evaluation Criteria:

* + Successful redirection logic based on session role data. (15 pts)
  + Correct creation of two new role-specific controllers and methods. (10 pts)
  + Proper route configuration for the new dashboards. (5 pts)

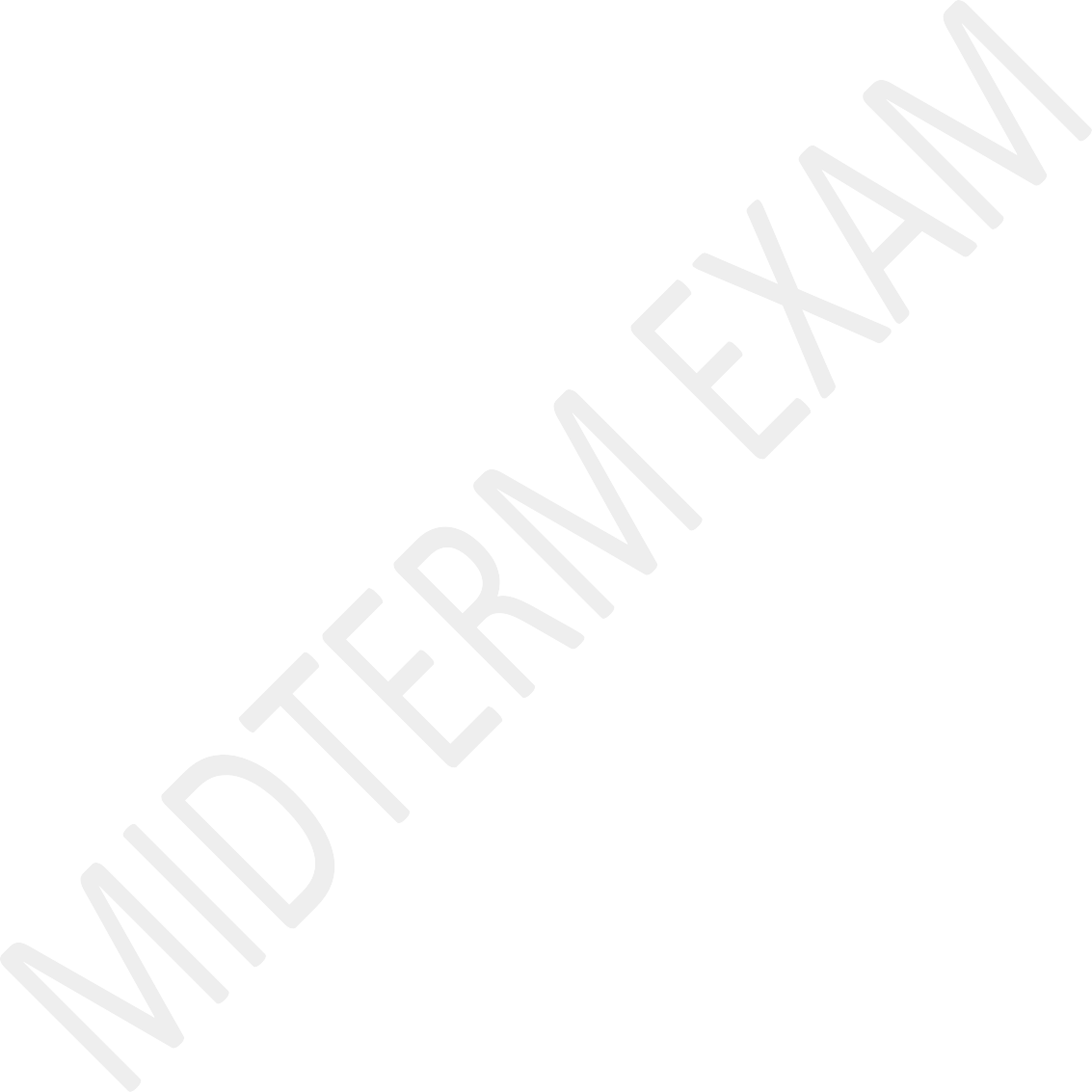
# Task 4 Implementing a Filter for Authorization (30 points) Scenario

Currently, any logged-in user can access any page by typing the URL. This is a critical security flaw.

# Task:

1. Generate a new filter named **RoleAuth**.
2. Inside the filter, write code that checks the user's session.
   * If the user's role is **admin**, they are allowed to access any route starting with /admin.
   * If the user's role is **teacher**, they are only allowed to access routes starting with **/teacher**.
   * If the user's role is **student**, they are only allowed to access routes starting with **/student** and the **/announcements** route.
   * If a user tries to access a route not permitted for their role, redirect them to the **/announcements** page with an error flash message “**Access Denied: Insufficient Permissions**".
3. Register the **RoleAuth** filter in **app/Config/Filters.php**. Then, apply this filter to the appropriate route groups in **app/Config/Routes.php** to protect the **/admin/\*** and **/teacher/\*** routes.

# Evaluation Criteria:

* + Correct filter creation and logical implementation of role checks. (15 pts)
  + Proper registration of the filter in the configuration. (5 pts)
  + Successful application of the filter to route groups effectively restricting access. (10 pts)