Grade Translator (Group 5):

(Ammar Ahmad Rizvi, Rumaisa Kamran, Hafsa Khurram, Kareemdad Niaz)

User Stories:

The admin member logs in with pre-determined credentials, in case the credentials match the ones determined beforehand the list of classrooms is displayed.

The admin member gets an option to add classes when the admin member clicks on the button, they can add a classroom

Test Cases:

USER NAME	PASSWORD	EXPECTED OUTCOME	POST CONDITION
admin	gradetranslator	Successful	Add a classroom named science 101
Admin	gradetranslator	Successful	Click randomly on the screen. Expected outcome – No change on screen
Admin	GradeTranslator	Failed	Login Page
AdMin	Gradetranslator	Failed	Login Page

Add_a_classroom Page**: Fields**

1. Class title,
2. Class code
3. Instructor code

Developer's Note:

The Authentication Problem

A simple authentication script may work but we discovered that for a user **admin** with password **admin**, a user **ADMIN** with password **ADMIN** can also log in. We need to make this script case sensitive. Here is the solution:

The Authentication Solution

First, we should use hash passwords. A change in capitalization will give us a different hash, so that sorts the issue.

If we want to compare user names in a case sensitive way, we probably also want to store them that way and be able to create different users for the logins 'foo' and 'Foo'. This is also possible but we need to change the database table.

More specifically, in the **table_login**, we need to set the collation of **username** and **password** column. We have changed the collation to **"latin1_swedish_cs"** instead of

"latin1_swedish_ci". (Point to remember, a value which has a trailing "cs" not "ci" like. As "ci" indicates case insensitive and "cs" is for case sensitive.)

Why Encryption is needed on password field?

We are aware of the encryptions that can be done so that the password is not stored as text. There is absolutely no reason that we should not use strong encryption on login credentials especially hash the password.

Hashes produce a layer of security when passing traffic because decoding them needs to be brute forced. It also stores uppercase A and lowercase a as different hashes so it takes care of the case sensitivity issue.

Most commonly use methods are MD5 and SHA-256. We have used MD5 to add a layer of encryption (as it is widely in use).

What is an MD5 hash?

An MD5 hash is created by taking a string of an any length and encoding it into a 128-bit fingerprint. Encoding the same string using the MD5 algorithm will always result in the same 128-bit hash output.

MD5 hashes are commonly used with smaller strings when storing passwords, credit card numbers or other sensitive data in databases such as the popular MySQL. This tool provides a quick and easy way to encode an MD5 hash from a simple string of up to 256 characters in length.

MD5 hashes are also used to ensure the data integrity of files. Because the MD5 hash algorithm always produces the same output for the same given input, users can compare a hash of the source file with a newly created hash of the destination file to check that it is intact and unmodified.

An MD5 hash is NOT encryption. It is simply a fingerprint of the given input. However, it is a one-way transaction and as such it is almost impossible to reverse engineer an MD5 hash to retrieve the original string.

Useful Link: You can generate more MD5 hash from the below link.
https://www.md5hashgenerator.com/