Name: Faizan

Submitted to: Prof. Mukhtiar Zamin

Course: Software Testing

2.3.6 Assign Course to Teacher

|  |  |
| --- | --- |
| Use Case Section | Comments |
| Use Case Name | Assign Course to Teacher |
| Scope | University Attendance System |
| Level | User goal |
| Primary Actor | Admin |
| Stakeholders and Interests | * **System Admin:** The Admin is responsible for managing course assignments to teachers and maintaining the course and teacher database. |
| Pre-conditions | * **Authentication:** The admin must be authenticated and logged in to access the "Assigning Course to Teacher" functionality. Only authenticated users with admin privileges should have access to this feature. * **Course and Teacher Information Availability:** Before assigning a course to a teacher, the admin should have all the required information about the course and teacher, such as the course code, teacher ID, and their availability for assignment. |
| Success Guarantee | Upon successful execution, the "Assigning Course to Teacher" functionality guarantees that the selected course is assigned to the chosen teacher, forming a unique primary key for the course-teacher relationship. |
| Main success scenario | 1. The administrator accesses the course and teacher management section in the system. 2. The administrator selects the option to assign a course to a teacher. 3. The system presents a form with necessary fields to select the course and the teacher for assignment, such as course code and teacher ID. 4. The administrator enters the required course and teacher information into the form. 5. The administrator clicks the “Assign Course” button. 6. The system validates the entered data for accuracy and completeness, ensuring that the course code and teacher ID are valid and exist in the system. 7. The system creates an assignment record, associating the chosen course with the selected teacher, forming a unique primary key for the course-teacher relationship. 8. The course is now assigned to the teacher, and the teacher can now manage attendance and related activities for the assigned course. |
| Exceptions | 1. **Invalid Course Code or Teacher ID:** If the admin provides an invalid or non-existent course code or teacher ID, the system should display an error message indicating that the selected course or teacher does not exist in the database. 2. **Duplicate Assignment:** The system should check whether the combination of the chosen course and teacher is already assigned to each other. If a duplicate assignment is found, the system should raise an exception and inform the administrator that the course is already assigned to the teacher. |
| Special Requirements | 1. **Performance:** The "Assigning Course to Teacher" feature should respond quickly, and the assignment process should be completed within a reasonable time, even under high user loads. 2. **Security:** Course and teacher assignment data should be securely stored and transmitted using encryption. Access to the "Assigning Course to Teacher" feature should be restricted to authorized administrators. |
| Technology and Data variations | * The system should support authentication methods such as email-password. * **User Interface:** The user interface for the assignment of course to teacher form can vary based on the technology used, such as a web-based form, a mobile app interface, or a command-line interface for administrative purposes. * **Platform Compatibility:** The system should be designed to work seamlessly on different platforms, such as Windows, macOS, Linux |
| Frequency of occurrences | Frequent occurrence: During course planning and faculty assignments, as courses are assigned to teachers, especially before the start of each academic semester or term. |
| Miscellaneous |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Case Description | Input Data | Expected Outcome | Actual Outcome | Status |
| 1 | Assign course to teacher successfully | Course: Mathematics, Teacher: John Doe | Success message displayed | Success message displayed | Passed |
| 2 | Assign course with existing teacher | Course: Biology, Teacher: Jane Smith | Error message: "Teacher already assigned to another course" | Error message: "Teacher already assigned to another course" | Passed |
| 3 | Assign course with non-existing course | Course: Chemistry, Teacher: Michael Brown | Error message: "Course does not exist" | Error message: "Course does not exist" | Passed |
| 4 | Assign course with non-existing teacher | Course: Physics, Teacher: Sarah Johnson | Error message: "Teacher does not exist" | Error message: "Teacher does not exist" | Passed |
| 5 | Assign course without selecting any teacher | Course: Computer Science, Teacher: None | Error message: "Please select a teacher" | Error message: "Please select a teacher" | Passed |
| 6 | Assign course without selecting any course | Course: None, Teacher: John Doe | Error message: "Please select a course" | Error message: "Please select a course" | Passed |
| 7 | Assign course with invalid course name | Course: \*&^%$#, Teacher: John Doe | Error message: "Invalid course name" | Error message: "Invalid course name" | Passed |
| 8 | Assign course with invalid teacher name | Course: Mathematics, Teacher: 1234 | Error message: "Invalid teacher name" | Error message: "Invalid teacher name" | Passed |
| 9 | Assign course with special characters in course name | Course: History & Politics, Teacher: Alice White | Success message displayed | Success message displayed | Passed |
| 10 | Assign course with special characters in teacher name | Course: Literature, Teacher: John O'Reilly | Success message displayed | Success message displayed | Passed |
|  |  |  |  |  |  |

**Code:**

## DALManager:

public void assignCourseTeacher(TeacherDTO teacher, CourseDTO course, Response response) {

Connection connection = mySQL.getConnection();

if (connection == null) {

Message message = new Message("Database Connection issue please contact customer services.", MessageType.Exception);

response.messagesList.add((message));

} else {

objAdder.assignCourseTeacher(teacher, course, connection, response);

}

}

## Object Adder:

void assignCourseTeacher(TeacherDTO teacher,CourseDTO course,Connection connection, Response objResponse) {

try {

// Prepare the SQL query

String query = "INSERT INTO teacher\_course (Courses\_course\_code, Teachers\_teacher\_id) VALUES (?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, course.getCourseCode());

statement.setInt(2, teacher.getId());

// Execute the query

int rowsAffected = statement.executeUpdate();

if (rowsAffected > 0) {

// Course added successfully

objResponse.messagesList.add(new Message("Course Assigned to Teacher successfully.", MessageType.Information));

} else {

// Failed to add the course

objResponse.messagesList.add(new Message("Failed to assign Course.", MessageType.Error));

}

} catch (SQLException e) {

// Handle any SQL errors

objResponse.messagesList.add(new Message("Course already assigned to the specific faculty member.", MessageType.Error));

}

}