Assignment 2

Student: Sicobean Alexandra Maria

**Group:30431**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The main objective of this assignment is to help students understand the MVC architectural pattern, services, repository, and unit tests. The application is a web-based system designed to track laboratory activity for the Software Design laboratory. The application has two types of users: students and teachers. Users must register and provide an email and password to access the application. Teachers can perform the following operations: login, CRUD on students, add/edit/delete laboratory classes, CRUD on attendance, CRUD on assignments, and grade assignments individually. Students can perform the following operations: register using the token generated by the teacher, login with the username and password, view a list of laboratory classes, view the assignments for a laboratory class, and create an assignment submission.

# Functional Requirements

• Teachers can login.

• Teachers can perform CRUD operations on students.

• Teachers can add/edit/delete laboratory classes.

• Teachers can create, update, and delete assignments for laboratory classes.

• Teachers can grade individual assignments.

• Students can register using the token generated by the teacher.

• Students can login with a username and password.

• Students can view a list of laboratory classes.

• Students can view the assignments for a laboratory class.

• Students can create an assignment submission.

# Non-functional Requirements

• The application should be easy to use.

• The application should be responsive and fast.

2. Use-Case Model

Use case: Register as a Student

Level: User goal

Primary actor: Student

Main success scenario:

1. Student opens the application
2. Student selects the option to register as a student
3. The system presents a form for the token
4. The system validates the token and erases it from the database
5. The student enters their email, password, name, group and hobbies
6. The system creates a new student account with the provided email and password
7. The student is redirected to the Subjects page

Use Case: Add assignemnt

Level: User Goal

Primary Actor: Teacher

Main Success Scenario:

1. Teacher logs into the application
2. Teacher creates a new subject by providing a title
3. Teacher creates a new laboratory within the subject providing the laboratory number, date, title, curricula, and a long description of the lab text.
4. Teacher adds an assignment to the laboratory class, including the name, deadline, and a long description of the assignment text.
5. Teacher saves the changes
6. The system updates the database with the new laboratory class and assignments.

Use Case: Submit assignment

Level: User Goal

Primary Actor: Student

Main Success Scenario:

1. Student logs into the application.
2. Student views the list of laboratory classes available.
3. Student views the details of the selected laboratory class, including the laboratory number, date, title, curricula, and a long description of the lab text.
4. Student views the list of assignments for the selected laboratory class, including the name, deadline, and a long description of the assignment text.
5. Student submits an assignment for the selected laboratory class by providing a link to a git repository and a short comment (optional) for the teacher.
6. Student saves the submission.
7. The system updates the database with the new submission.
8. The system displays a success message to the student

3. System Architectural Design

**3.1 Architectural Pattern Description**

Building a layered architecture for a .NET web application typically involves separating the application into distinct layers based on their responsibilities. Therefore, in my project I had a presentation, business logic, and data access layer. This approach can help improve code organization, maintainability, and scalability.

Here are the steps I followed to build a layered architecture with .NET web application:

1. Define the layers
2. Create projects for each layer
3. Define interfaces
4. Implement the layers
5. Configure Dependency Injection
6. Define Models: Define models for each layer to represent the data they use
7. Implement Controllers in the presentation layer.
8. Implement Services in business layer
9. Implement Repositories in data layer

**3.2 Diagrams**

4. UML Sequence Diagrams

Diagram

Description automatically generated

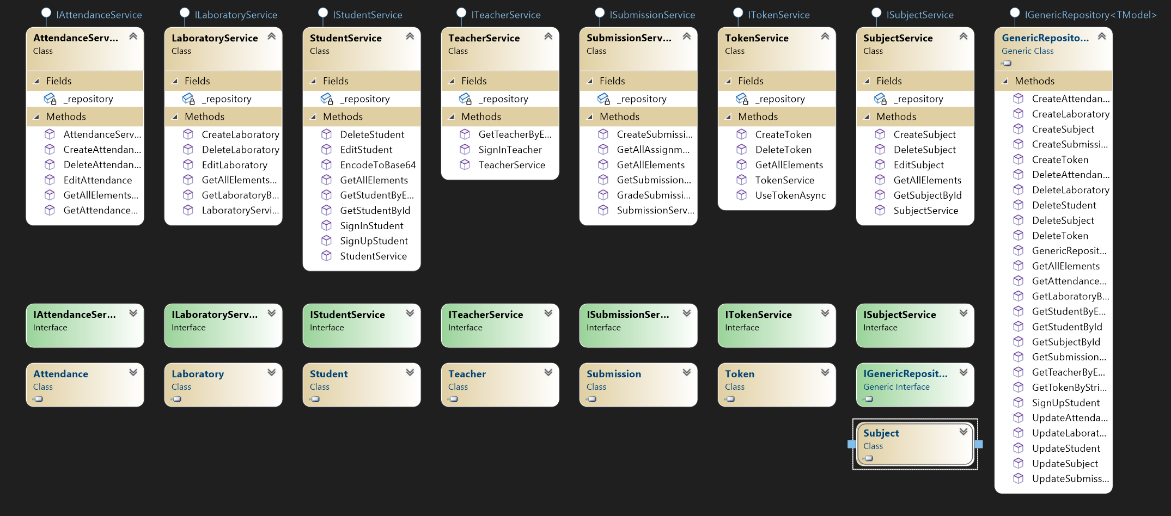
5. Class Design

**5.1 Design Patterns Description**

For the view layer, I decided to use an MVC design pattern. The main controller is the HomeController. I have different views for each user and CRUD operation. However, the Model layer is only used for persistent data such as Booleans that determine whether an admin or cashier is logged in or not.

**5.2 UML Class Diagram**

The only proper classes in the project are the ones above. The controller contains an object for each of the interfaces above.



6. Data Model

*Diagram

Description automatically generated*

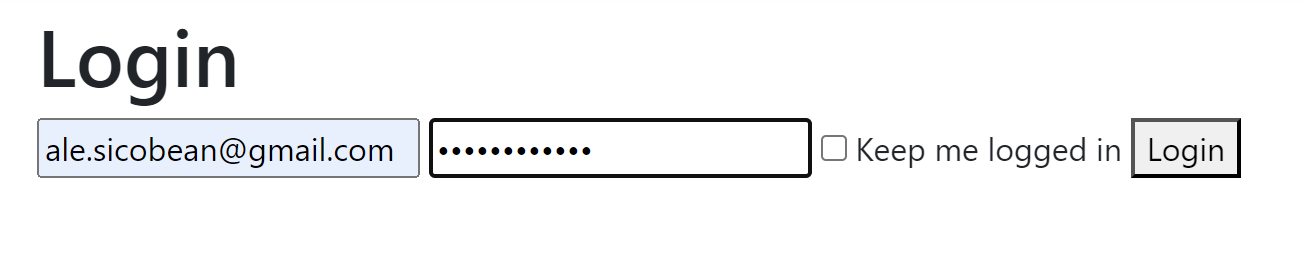
Diagram

Description automatically generated

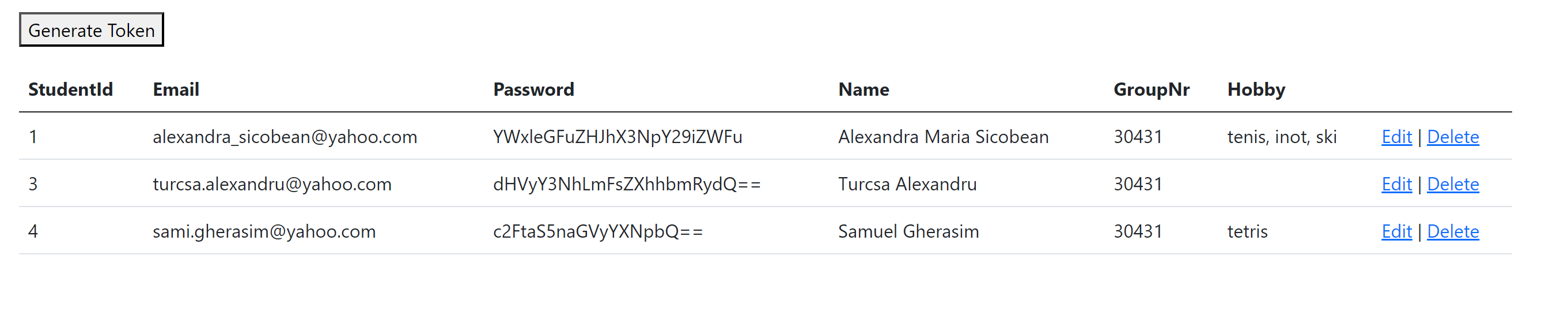
1. System Testing

. **Tested flow for 10 (Example):**

1. Login as Teacher

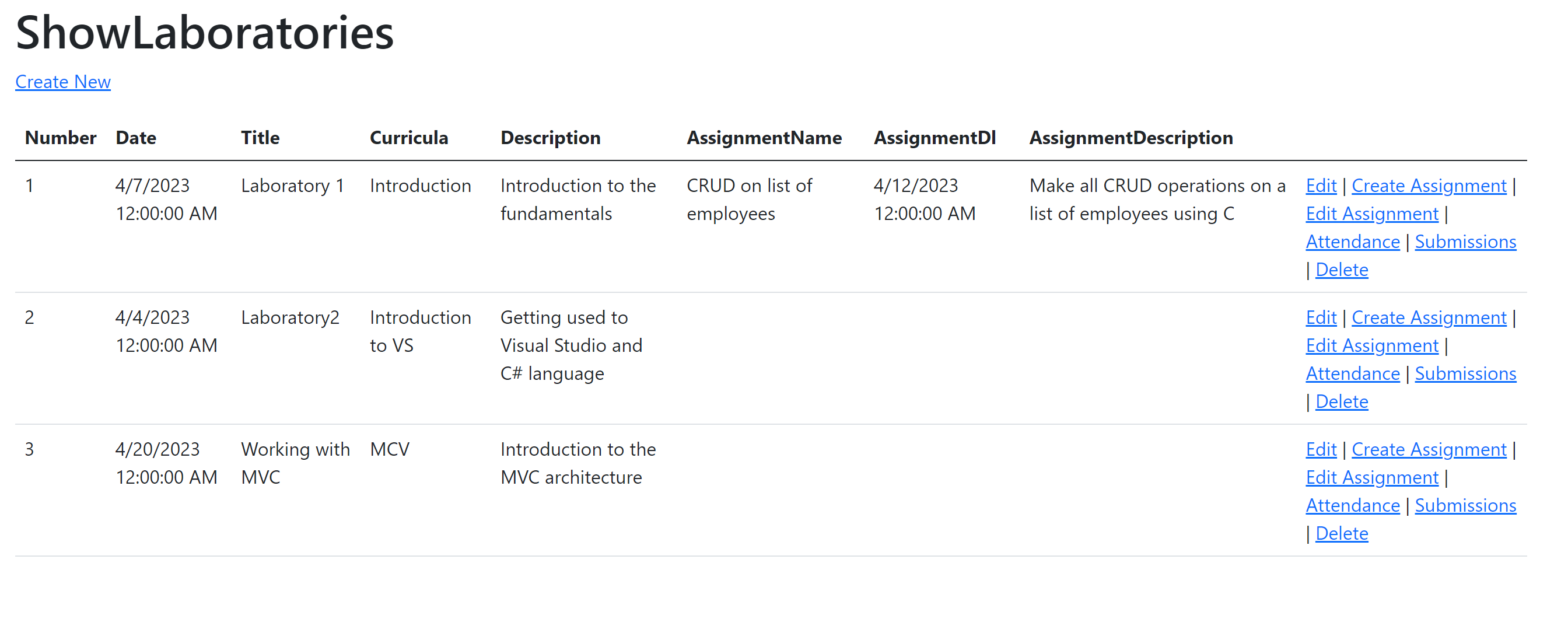


1. Create Student, save token

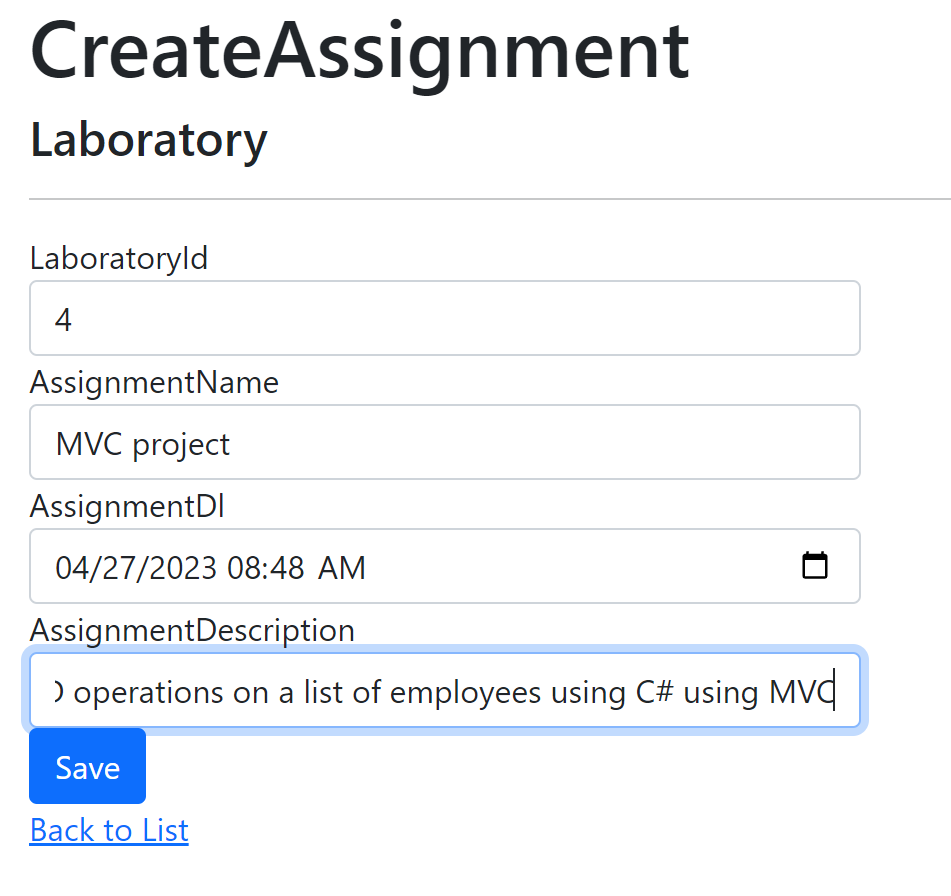




1. Create Laboratory



1. Create Assignment for lab created at 3

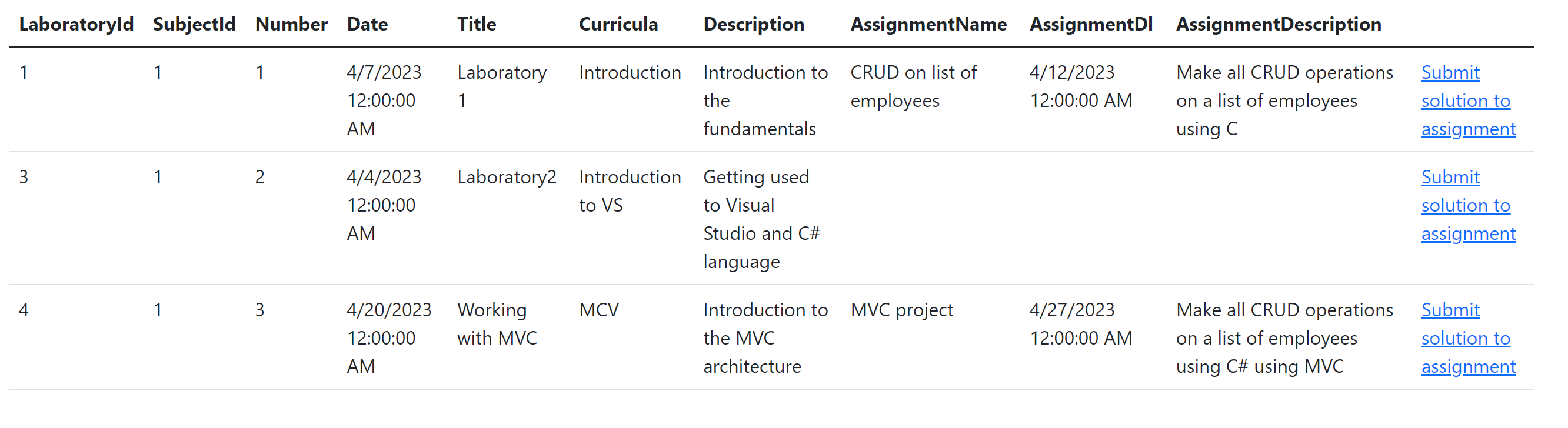


5. Register as student with user, password and token provided

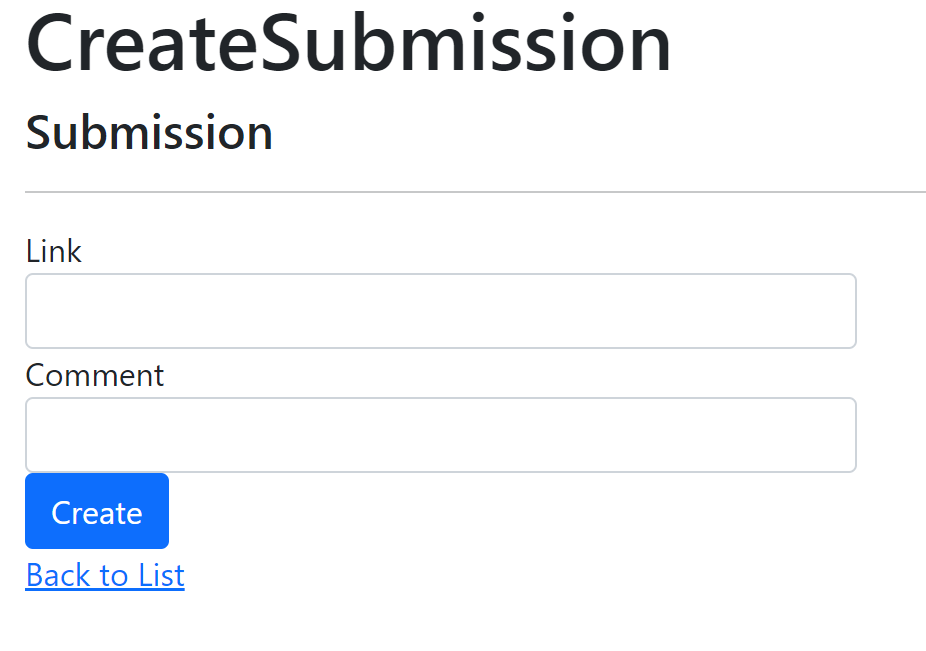
6. View list of lab classes



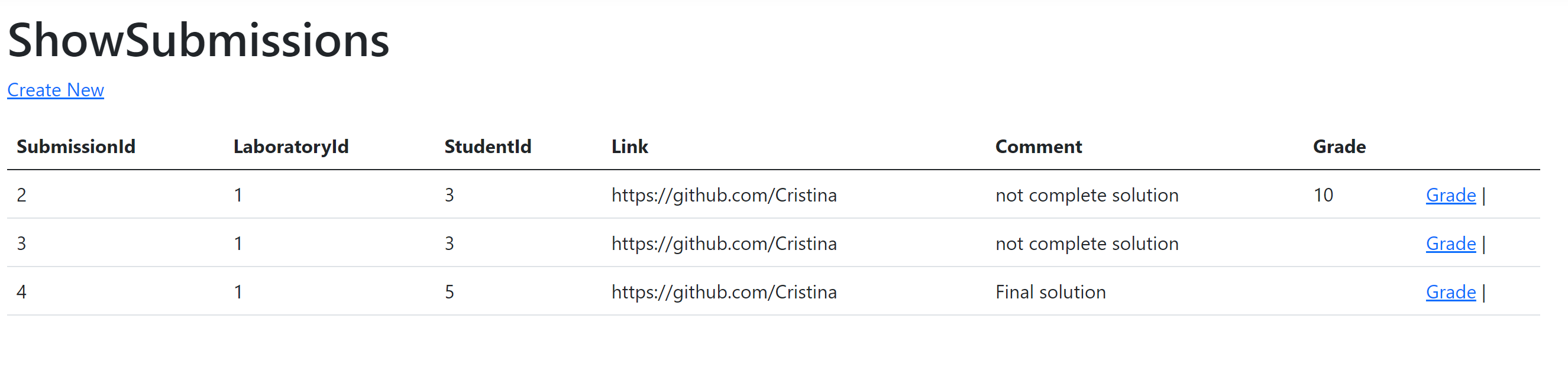
1. View assignments for lab



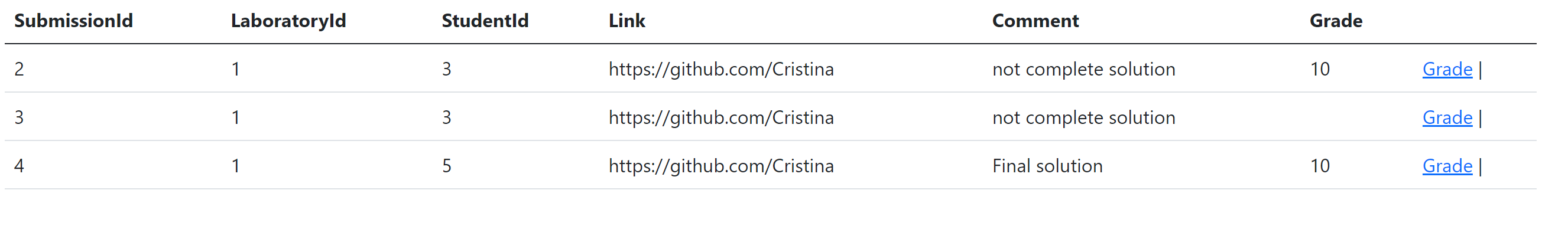
1. Submit assignment for lab



1. Login again as Teacher



10. Grade submitted assignment



8. Bibliography

1. <https://medium.com/aspnetrun/layered-architecture-with-asp-net-core-entity-framework-core-and-razor-pages-53a54c4028e3>

2. <https://dotnet.microsoft.com/en-us/learn>

3. <https://www.javatpoint.com/net-framework>