Student: Ratoi Razvan Valeriu

**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 4

4. UML Sequence Diagrams 5

5. Class Design 6

6. Data Model 6

7. System Testing 7

8. Bibliography 7

1. Requirements Analysis

# Assignment Specification

This assignment was to develop a web application that would serve as a management system for the laboratories for the Software Design course.

# Functional Requirements

The functional requirements were the following:

* Teacher:
  + CRUD on laboratories, students, assignments, attendance
  + Register a student and create a token
  + Grade submitted assignments
  + Log in (available for both users)
* Student:
  + Can access laboratories, assignments
  + Can create an account with the token generated from the teacher
  + Can submit assignments

# Non-functional Requirements

The application had to check the following constraints:

* The data would be stored in a relational database, respecting the 1st, 2nd and 3rd normal forms and also using the appropriate relationships (1:1, 1:n, n:m)
* Use the MVC architectural pattern
* API design should be restful
* Connection string and magic strings should be stored in a configuration file
* Use an Entity framework to access the database

*Diagram

Description automatically generated*2. Use-Case Model

Use case: Register a student and generate a token

Level: User-goal level

Primary actor: Teacher

Main success scenario:

* + The Teacher initiates the Register use case.
  + The system presents the registration form.
  + The Teacher enters the student's information, including name, email, and password.
  + The system validates the information and generates a unique token for the student.
  + The system displays the token to the Teacher.
  + The Teacher records the token and provides it to the student.

Extensions:

* If the system detects that the email is already registered, it displays an error message and prompts the Teacher to enter a new email.
* If the system detects that the password does not meet the security requirements, it displays an error message and prompts the Teacher to enter a new password.
* If the system cannot generate a unique token, it displays an error message and prompts the Teacher to try again later.

3. System Architectural Design

**3.1 Architectural Pattern Description**

Without having a front-end side for this application, the best way to reproduce the MVC architectural pattern with a layered architecture. Hence, I will be using 3 layers for the back-end of the application: Repository (the one accessing the database), Service (the one processing the data) and Controller (that connects the logic with the requests).

**3.2 Diagrams***Diagram

Description automatically generated*

Architecture Diagram

A picture containing graphical user interface

Description automatically generated

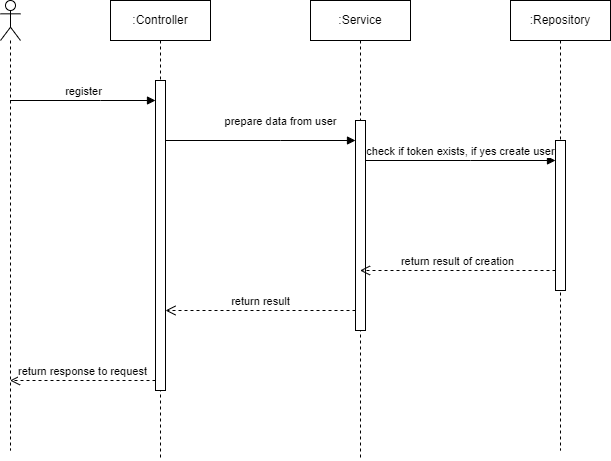
Package diagram

Text

Description automatically generated

Deployment diagram

4. UML Sequence Diagrams

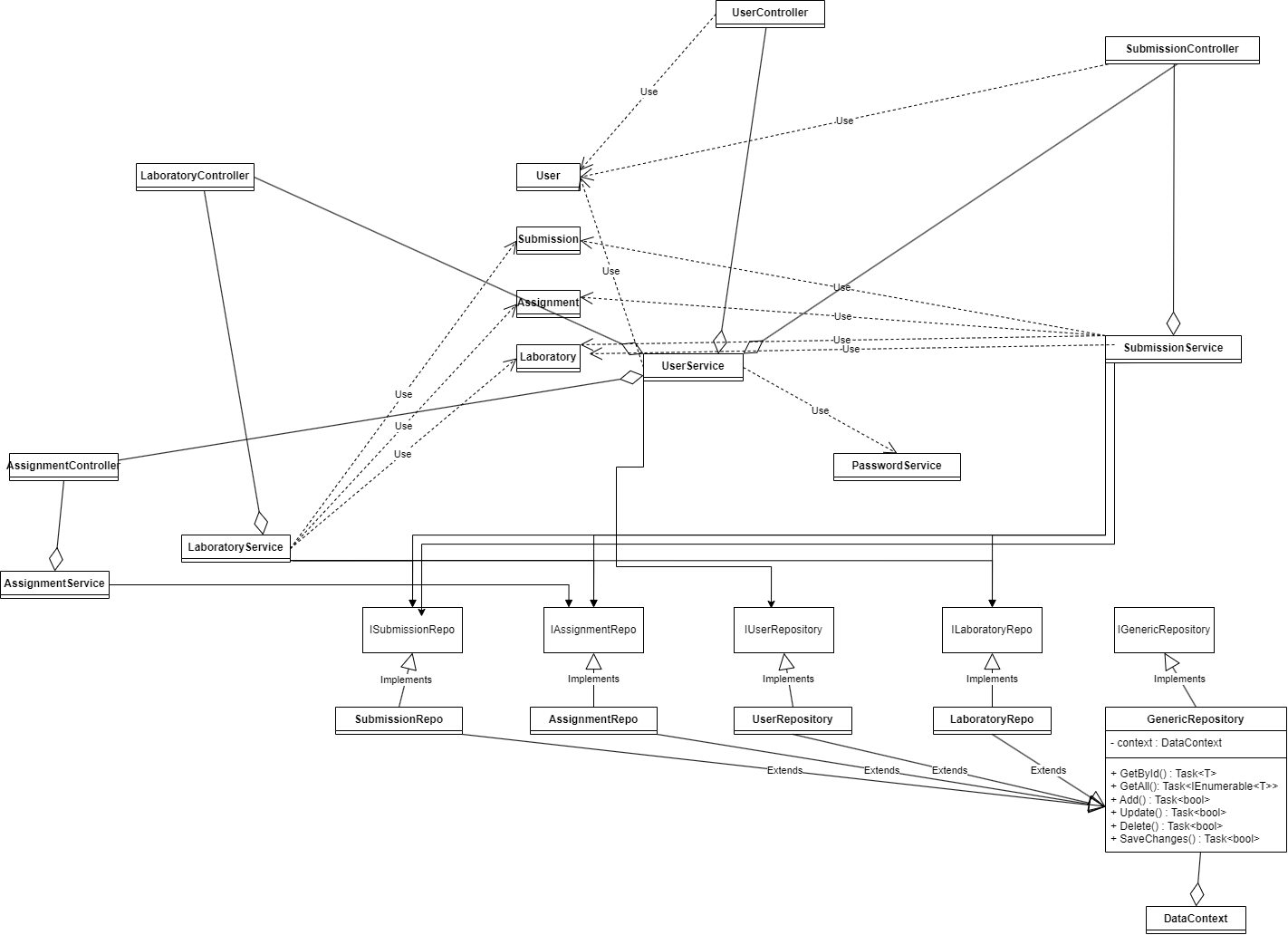


5. Class Design

**5.1 Design Patterns Description**

For this application, I used a variation of the Factory Design Pattern, where I used a Token Creator class to generate the token needed for registering the student.

**5.2 UML Class Diagram**



6. Data Model

For implementing this application, I used the following classes:

* Token, which would have only the id and the value.
* User, which would have the personal details, alongside a role.
* Credentials, that would act as an user, along with a token, to be used when a student would register.
* LoginCredentials, having just the username and password, used for authentication.
* Laboratory, containing a number, details, curricula and date.
* Assignment, having a deadline, a description and a laboratory which is assigned to.
* Submission, having the id of the user who submitted it, the id of the assignment and the link to the solution; acts as a many-to-many relationship between users and assignments.
* Attendance, built in the same idea as the previous class: having just the id of the user and the id of the laboratory the user was present to, acts as a many-to-many relationship between the two.

7. System Testing

For testing the implementation, I used Insomnia to call each request with appropriate data (for both best cases and cases where an error would be expected) and would check the response from the request.

8. Bibliography

* <https://app.pluralsight.com/library/courses/object-oriented-programming-fundamentals-csharp/table-of-contents>
* <https://app.pluralsight.com/player?course=csharp-collections&author=simon-robinson&name=csharp-collections-fundamentals-m1-welcome&clip=0>
* <https://www.amazon.com/NET-Head-First-Microsoft-Programming/s?ie=UTF8&page=1&rh=n%3A764452%2Cp_lbr_books_series_browse-bin%3AHead%20First>
* <https://app.pluralsight.com/library/courses/implementing-restful-aspdotnet-web-api/table-of-contents>