Web Application for tracking the laboratory activity

Analysis and Design Document

Student: Izabella Bartalus

**Group: 30231**

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 application is a Web Application for tracking the laboratory activity for the Software Design laboratory. The application has two types of users (student and teacher) which provide a username and a password to use the application. The teacher has control over the student’s information, over laboratory classes and over assignments. The student can view a list of laboratory classes, a list of assignments for a laboratory class and create an assignment submission.

# Functional Requirements

The functional requirements provided by the application are:

* Login and logout for the users;
* CRUD on students;
* CRUD on laboratory classes;
* CRUD on assignments;
* Grade the submitted assignments individually;
* Calculate the final grade;
* Create assignment submission.

# Non-functional Requirements

The non-functional requirements of the application are :

* Availability
  + The application is available 99% of the time.
* Performance
  + The application’s response time shouldn’t be more that 200 ms.
* Security
  + The application should check the top ten OWASP requirements.
* Testability
  + Number of defects of the application should be less than 5.
* Design Constraints
  + For designing this application we consider the following constraints: C# and .NET framework, and the Three Tier Architecture Model.

2. Use-Case Model

Use case: Log In

Level: application level

Primary actor: Teacher, Student

Main success scenario: log in the application

Extensions: Display error message “Invalid username/password!”

Use case: Log Out

Level: application level

Primary actor: Teacher, Student

Main success scenario: log out of the application

Extensions: display error message “Could not log out!”

Use case: CRUD on student’s information

Level: user-goal level

Primary actor: Teacher

Main success scenario: Create, read, update or delete a student

Use case: CRUD on laboratory classes

Level: user-goal level

Primary actor: Teacher

Main success scenario: Create, read, update or delete a laboratory class

Use case: CRUD on assignments

Level: user-goal level

Primary actor: Teacher

Main success scenario: Create, read, update or delete an assignment

Use case: Grade the submitted assignment individually

Level: user-goal level

Primary actor: Teacher

Main success scenario: Update grade model by adding a value to the grade field

Use case: Calculate the final grade as an average of all marks

Level: user-goal level

Primary actor: Teacher

Main success scenario: Compute the final grade by taking the student id and its assignmets

Use case: Create assignment submission

Level: user-goal level

Primary actor: Student

Main success scenario: Students are able to insert a link to a git repository and a comment

Use case: View a list of laboratory classes

Level: user-goal level

Primary actor: Student

Main success scenario: Get all laboratories

Use case: View the assignments for a laboratory class

Level: user-goal level

Primary actor: Student

Main success scenario: Get all assignments for a laboratory class

Diagram

Description automatically generated

3. System Architectural Design

**3.1 Architectural Pattern Description**

Layered Architecture Pattern:

* Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g. presentation logic or business logic).

Model-View-Controller: also known as MVC pattern, divides an interactive application in to 3 parts as:

* model  - contains the core functionality and data;
* view - displays the information to the user (more than one view may be defined);
* controller - handles the input from the user.

We will use a relational database to store out data, and this is how we will be manipulating our data

**3.2 Diagrams**

Diagram

Description automatically generated

4. UML Sequence Diagrams

Sequence diagram for adding a new laboratory class:

Diagram

Description automatically generated

5. Class Design

**5.1 Design Patterns Description**

The Observer pattern is useful and can be used in situations where one object has to observe another object and need to receive notifications when the state of another object changes. The objects that contain some interesting information are represented as the subject. It notifies the other objects to change to its state is known as the publisher. All the other objects that want to track changes to the publisher's state are called subscribers. I used this design pattern to send a message to the student (message is displayed in console) when the teacher updates the assignment by grading it.

**5.2 UML Class Diagram**

Diagram

Description automatically generated

6. Data Model

The Data Models used in this application are: UserModel, StudentModel, LaboratoryModel, AssignmentModel, GradingModel.

Diagram

Description automatically generated

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

8. Bibliography