Laboratory Activity Tracking System

Student: Deac Denisa Bianca

**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 application to be developed is a Java Spring Web API-based system that will allow the tracking of laboratory activity for the Software Design laboratory. The system will have two types of users: students and teachers. Students and teachers will need to provide an email address and password to use the application. The teacher will have the ability to create, read, update, and delete (CRUD) student records, laboratory classes, attendance records, and assignments. The teacher will also be able to grade individual assignments. Students will be able to register using a token generated by the teacher, view laboratory classes, view assignments, and submit assignments with a link to a Git repository.

# Functional Requirements

The following represent functional requirements that have been identified for this project:

* Allow teachers to log-in.
* Allow teachers to perform CRUD operations on student records.
* Generate a 128-character token for each new student record.
* Allow students to register using the token generated by the teacher and set a password.
* Allow students to log in using their email and password.
* Allow students to view a list of laboratory classes.
* Allow teachers to perform CRUD operations on laboratory classes.
* Allow teachers to perform CRUD operations on attendance records for each laboratory class.
* Allow teachers to assign an assignment for each laboratory class.
* Allow students to view the assignments.
* Allow students to make a submission with a link to a Git repository and optionally a comment.
* Allow teachers to grade submitted assignments individually.

# Non-functional Requirements

The non- functional requirements are the following:

* The application is responsive and performs tasks quickly and efficiently.
* The system provides clear and informative error messages for incorrect inputs or other errors.
* The application provides secure and encrypt passwords in the database with a one-way encryption algorithm.
* The application follows the Layers architectural pattern, is well-organized and maintainable.
* The database schema respects 1st, 2nd, and 3rd normal forms.

2. Use-Case Model

Use cases for teacher:

Diagram

Description automatically generated

Use cases for student:

Diagram

Description automatically generated

Use case: Create Student with token

Ticket Level: User-goal level

Primary actor: Teacher

Preconditions:

* Teacher is authenticated.

Main success scenario:

1. The Teacher selects the Register Student option through an http request.
2. The Teacher enters the required student details such as email, full name, group, and hobby.
3. The system generates a 128-character token for the student.
4. The system saves the student details and the generated token in the database.
5. The Teacher manually sends the token to the student via email.
6. The student uses the token to register in the system.
7. The system saves the student's password and marks the token as not used until the student sets a password.
8. The system will display a message telling that the action was executed with success.

Extensions:

* If the teacher credentials are wrong, the system will display an error message.
* If the student token is already used, the system displays an error message.Top of Form

3. System Architectural Design

**3.1 Architectural Pattern Description**

A well-liked software design pattern for creating scalable and maintainable programs is the layered architecture pattern, which I chose for my application. This structure separates the application into various layers, each with its own duties. Through clearly defined interfaces, these layers communicate with one another, resulting in a modular and adaptable design.

**3.2 Diagrams**

**System Architecture Diagram:**

The Presentation, Business, Model and Persistence layers make up the three layers that commonly make up the Layered pattern. The user interface and user interaction are handled by the Presentation layer, identified by the Controller package. Business logic and data processing are handled by the business and uses service classes. The Model layer contains all the entities needed for the application, and the persistence layer, represented by the Repository package, is responsible for interacting with the database through queries.

**Diagram

Description automatically generated**

PERSISTANCE LAYER

**Package diagram**

Diagram

Description automatically generated

* **Model**: This package contains all the entity classes that represent the data model of your application. These entities include **Student**, **Teacher**, **Laboratory**, **Assignment**, and **Submission**, which are all related to the functionality of the application
* **Business**: This package contains the services for all the entities in the model. These services are responsible for implementing the business logic of the application. The services will be called by the controllers to perform operations on the entities, such as creating, updating, deleting, and retrieving records.
* **Controller**: This package contains the controllers for the student and teacher entities. The controllers are responsible for receiving HTTP requests from clients, processing the requests, and returning responses to the clients.
* **Repository**: This package contains one repository for each entity in the model. A repository is responsible for handling data access for a specific entity.

**Component diagram**

Diagram

Description automatically generated

**Deployment Diagram**

A deployment diagram shows the physical architecture of a system and how the software components are deployed on hardware.

A picture containing diagram

Description automatically generated

4. UML Sequence Diagrams

Sequence diagram for the operation representing a teacher creating a new student with a corresponding token.

*Diagram

Description automatically generated*

5. Class Design

The classes I design follow the architectural pattern in terms of separation into packages and their purpose.

The classes that do the logical manipulation of data are represented by the **Controllers**, one for teacher and one for student, where the services, methods specific for each use case, are called. The interaction with the user is done via htpp requests, through postman. Each method has a specific path with the necessary annotation for posting, getting, putting, or deleting.

The service classes are the ones doing the computations and the logical part of the CRUD operations. There is a service class for each entity with specific methods for them. The repositories are instantiated and used here to retrieve the data from the database. They represent the business logic of the application.

The repositories are the persistence part of the project. I created a repository class for each entity, since these are interfaces that extend JBA Repository, making the connection with database through queries. Some basic query functions are already implemented, but for some entities I made a customized method.

**5.1 Design Patterns Description**

The design pattern I used is Repository Pattern. This pattern is used to abstract away the details of how data is persisted in the database. By implementing a Repository for each entity, the client code does not need to be concerned with the details of how data is stored and can simply interact with the entities through the Repository. This is achieved by creating a repository class for each entity that extends JpaRepository.

**5.2 UML Class Diagram**

*Diagram

Description automatically generated*

6. Data Model

The Student entity has attributes such as email address, full name, group, hobby, password, and token. The Teacher entity has attributes such as email address and password. The Laboratory entity has attributes such as laboratory number, date, title curriculum assignment and attendance list . The Assignment entity has attributes such as name, deadline, and description and a list of submissions. The Submission entity has attributes such as a link to the git repository, submission date, a short comment for the teacher and the grade of the submission.

In addition to these entities, there are also relationships between them. For example, a Laboratory can have an Assignment, relationship OneToOne and an also many submissions, from different students, in relationship OneToMany.

The data model is designed to follow the principles of 1st, 2nd, and 3rd normal forms, which means that the data is organized in a way that minimizes redundancy and improves data consistency. Proper relations between tables are also enforced to ensure data integrity. An ORM such as Hibernate or Entity Framework is used to interact with the database, making it easier to perform database operations and reducing the amount of boilerplate code required.

*Graphical user interface

Description automatically generated*

7. System Testing

For testing the system, I used validation through postman application. I created collections by separating the requests in folders depending on their use. I have 2 main folders, one for student requests and one for teacher. In the teacher folder there are subcategories for each main use case: CRUD for students, laboratories, assignments, attendance.

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

The requests look like this and at each testing they return validations messages.

Graphical user interface, text, application, email

Description automatically generated

A picture containing text

Description automatically generated

1. Bibliography

* <https://java-design-patterns.com/patterns/layers/>
* <https://www.youtube.com/watch?v=vOr1ewmCBDw&t=1057s>
* <https://www.youtube.com/watch?v=9SGDpanrc8U>