ASSIGNMENT A2
Laboratory Activity Tracking
Analysis and Design Document

Student: Curac Mihai - Ionut
Group: 30431

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The objective of this project is to design and implement a Java Spring Boot application for tracking lab activities of the Software Design laboratory. The application will be used by two types of users: students and teachers. Teachers will manage students' information, laboratory classes, the attendance for each lab, and assignments for specific labs. Students will be able to register using a token generated by the teacher, view a list of laboratory classes, and view the assignments for a laboratory class.

1.2 Functional Requirements

The following are functional requirements of the lab tracking system:

- The system should have two types of users: teachers and students.
- Both types of users must provide a username and a password to log in to the system.
- Teachers must be able to perform CRUD operations on students. When a student is created, a 128-character token should be generated, which the student will use to register.
- The system must allow teachers to add, edit, and delete laboratory classes. For each class, the system must track the laboratory number, date, title, curricula, and a long description with the laboratory text.
- Teachers must be able to perform CRUD operations on the attendance for each lab.
- Teachers must be able to perform CRUD operations on assignments. For each assignment, the system must track the name, deadline, and a long description with the assignment text.
- Students must be able to register using the token generated by the teacher and provide an email and a password to create their accounts.
- Students must be able to view a list of laboratory classes.
- Students must be able to view the assignments for a laboratory class.

1.3 Non-functional Requirements

The following are the non-functional requirements of the lab tracking system:

- The system must be secure and provide a safe environment for data storage and management.
- The data will be stored in a relational database, and the database model must respect 1st, 2nd, and 3rd normal forms, and proper relations between tables (1:1, 1:n, m:n).
- The system must use the MVC architectural pattern to organize the application.
- The API design should be RESTful.
- The system should use Hibernate to access the database.
- The system must use dependency injection to inject services in controllers and repositories in services.
- The system should provide a Postman collection to call APIs.

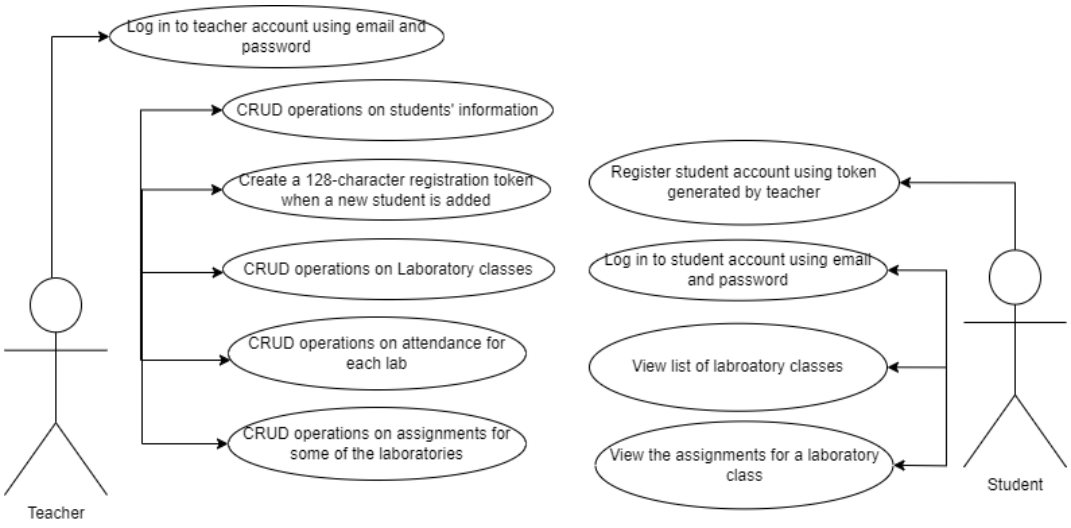
2. Use-Case Model

Use case: Register as a student
Level: User-goal level
Primary actor: Student
Main success scenario:

1. Student makes a HTTP request to “register” a student account.
2. Student provides an email address and a password.
3. Student writes “student” as their role and enters the registration token provided by the teacher.
4. Student sends the request and the system verifies the credentials.
5. System saves the student's information and login credentials in the database.
6. System sends a message confirming the successful registration.

Extensions:

- If the token is not valid or associated to a student in the database, the system sends an error message and returns to step 3.
- If the token has already been used to create a student account, the system sends an error message and asks the student to log into their account.
- If the registration fails, the system sends an error message with the reason why the registration failed.



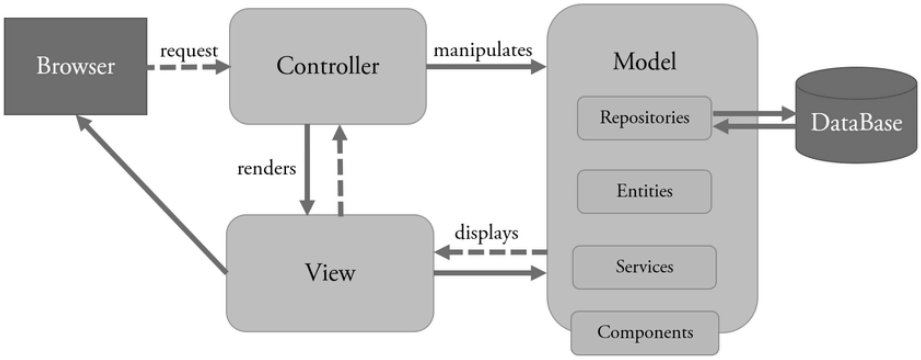
3. System Architectural Design

3.1 Architectural Pattern Description

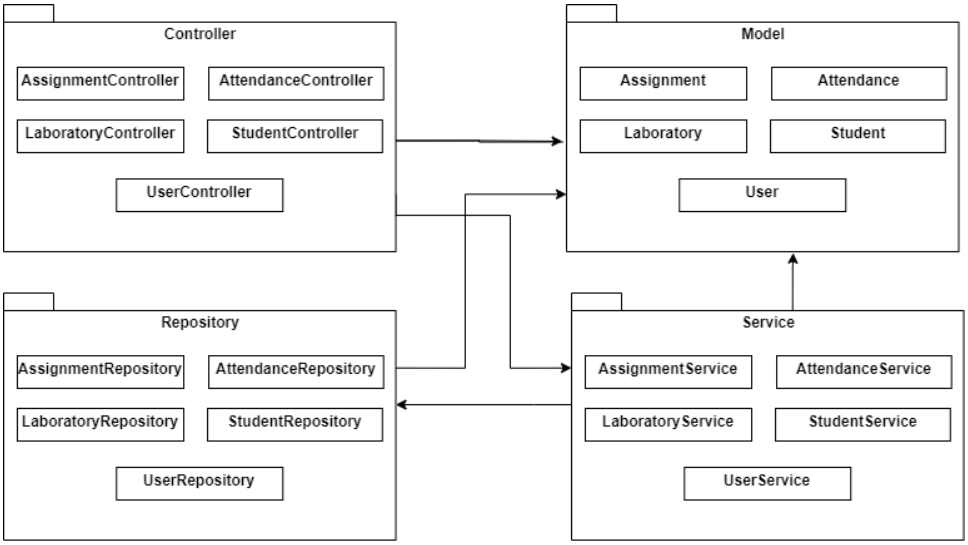
The system will follow the Spring Model-View-Controller (MVC) architectural pattern to organize the application. Spring MVC is a module of the Spring framework dealing with the MVC pattern. In this pattern, the model contains the data of the application, which can be a single object or a collection of objects. The controller contains the business logic of the application. The view represents the provided information in a particular format.

3.2 Diagrams

The use of the Spring MVC pattern allows for separation of concerns and easier maintenance and scalability. For instance, the Controller handles the interactions between the Model and the View. The system will also utilize the Repository pattern for data access. Repositories will be responsible for querying the database and returning entities to the Service layer, which will handle business logic and pass data to the Controllers for display in the Views.
This pattern will allow for better testability, maintainability, and decoupling of the data access layer from the rest of the application. It is illustrated in the following conceptual architecture:

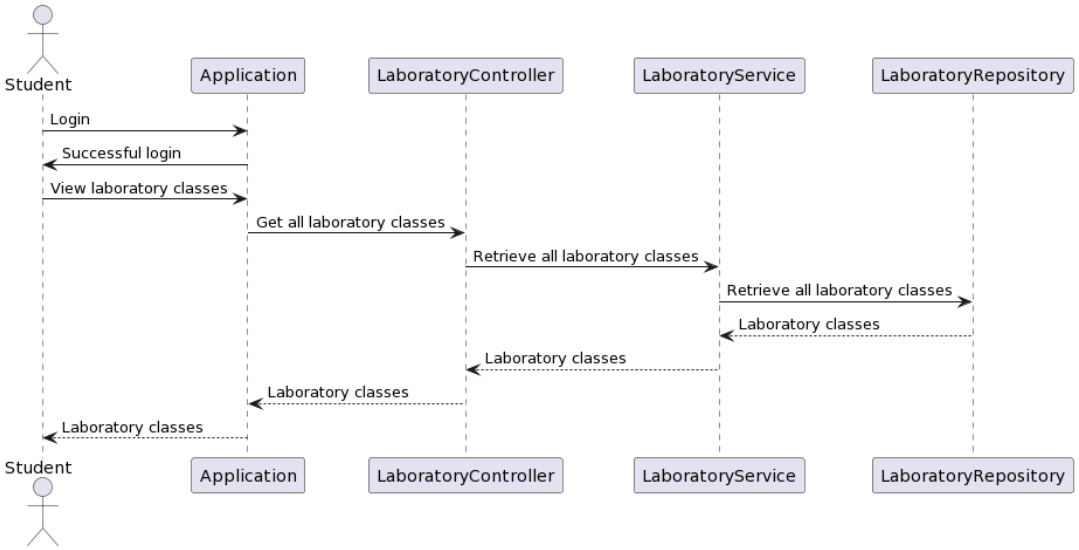


The packages diagram is the following:



4. UML Sequence Diagrams

For the scenario of a registered student requesting to view the laboratory classes, the sequence diagram of the system is the following:



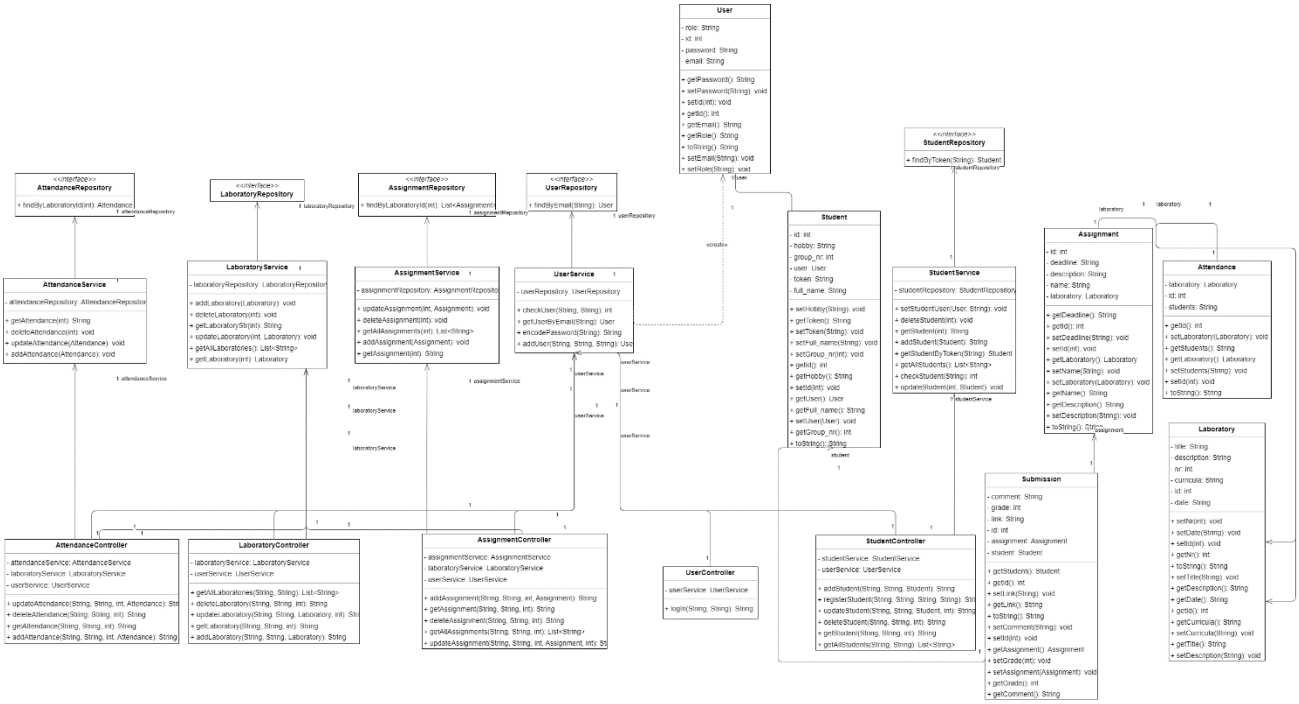
5. Class Design

5.1 Design Patterns Description

For this project, JPA and Hibernate were used to communicate with the database. Since Spring Boot annotations and JPA were used, a design pattern that is being applied is the Template method pattern. The template method pattern is a behavioral design pattern that defines the skeleton of an algorithm in a superclass but lets subclasses override specific steps of the algorithm without changing its structure. In the project, the superclass is the JpaRepository interface, which provides the basic CRUD operations for entities. The subclasses are the repositories, which extend the JpaRepository interface and override the methods as necessary to perform more specific database operations. This pattern helps to reduce code duplication and promote consistency across the application.

5.2 UML Class Diagram

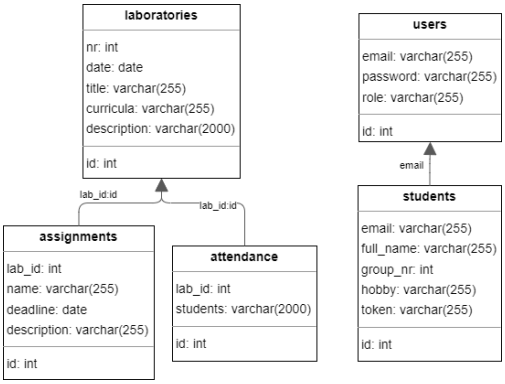
The UML Class Diagram is illustrated below.



6. Data Model

The data model for the system's implementation is represented by five entities: Assignment, Attendance, Laboratory, Student, and User. These entities are annotated with @Entity and @Table to specify the corresponding database table. The relationships between the entities are represented with JPA annotations such as @ManyToOne, @OneToOne, and @JoinColumn.

- The Assignment entity has a many-to-one relationship with the Laboratory entity, indicating that one laboratory can have multiple assignments. It also has attributes such as name, deadline, and description.
- The Attendance entity has a direct relationship with the Laboratory entity. It has an attribute called "students", which is a string that contains the names of the students who attended the laboratory.
- The Laboratory entity has attributes such as nr, date, title, curricula, and description. It has a one-to-many relationship with the Assignment entity.
- The Student entity has a one-to-one relationship with the User entity, indicating that each student has one user account. It also has attributes such as full_name, group_nr, hobby, and token.
- The User entity has attributes such as email, password, and role. It is referenced by the Student entity using a one-to-one relationship.



7. System Testing

The testing strategies that were used in the lab activity tracking system are:

- Integration Testing: This strategy was used to test how the components of the system interact with each other. Integration tests were written to verify that the system components are integrated correctly and working together as intended.
- Validation Testing: This strategy was used to validate the correctness of user input and ensure that the system can handle unexpected or invalid input. Tests were given through Postman HTTP Requests.

8. Bibliography

- https://www.baeldung.com/spring-boot-hibernate
- https://www.baeldung.com/hibernate-one-to-many
- https://javabrains.thinkific.com/courses/springboot-quickstart
- https://www.jetbrains.com/idea/guide/tutorials/marco-codes-hibernate-jpa/introduction/
- https://www.javadevjournal.com/spring-boot/spring-boot-with-hibernate/
- https://www.baeldung.com/jpa-one-to-one
- The information provided in the laboratories.