Laboratory Activity Tracker

Student: Zeibel Antonia

**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 4

3. System Architectural Design 5

4. UML Sequence Diagrams 6

5. Class Design 6

6. Data Model 7

7. System Testing 9

8. Bibliography 10

1. Requirements Analysis

# Assignment Specification

The goal of this assignment is to design and implement an application for the tracking the laboratory activity for the Software Design laboratory. The application should have two types of users (student and teacher) which must provide an email and a password to use the application.

# Functional Requirements

The teacher user can use the application only if the authentication process is successful. The authentication process consists of providing the username and the password to the system. In case of non-existing credentials, the teacher user can register to the system. The student user cannot create a new account and should authenticate using the username, password and token provided by the teacher.

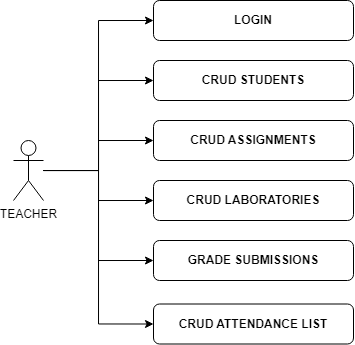
After the authentication step, the user can request different operations on the system based on their specific role. For example, for a teacher user can perform CRUD operations on students, add/edit/delete laboratory classes, perform CRUD operations on attendances and assignments. The student user, can perform read operations on both laboratory classes and assignments and also create assignment submissions which contain a link to a git repository and an optional short comment for the teacher.

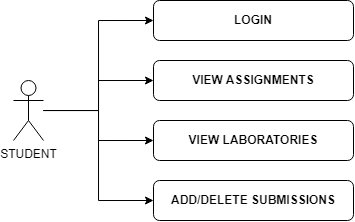
# Non-functional Requirements

As I have briefly mentioned, the application provides non-functional security features. When the users access their accounts, register or login, the provided details’ are stored into the database, while the password is encrypted for privacy purposes. Moreover, there are specific operations for each type of user according to their role.

There are also validations for these operations. One can observe these validations when creating an object or updating one.

2. Use-Case Model





**Use-case:** Create a student

**Primary Actor:** Teacher

**Success Scenario:** The user logs into the application as a teacher using a valid username and password which were previously created in the registration step. After the teacher is logged in, the student can be created by sending a post request providing the necessary details: username, password, full name, hobby and group number. The new student is created and added to the connected database.

**Extensions:** If the teacher does not provide valid details for creating a student, the response of the request is not a successful one and a new student will not be created.

**Use-case:** View list of laboratories.

**Primary Actor:** Student

**Success Scenario:** The user logs into the application as a student using a valid username, password and token which were previously created in the registration step. After the student is logged in, the list of laboratories can be read by sending a get request. All the laboratories will be fetched and the response of the request will be a list of laboratories with all the fields of a laboratory: date, title, curricula for what are the topics presented in that lab and a long description with the laboratory text.

**Extensions:** The user could not log into the application as a student and the list of laboratories will not be presented.

3. System Architectural Design

**3.1 Architectural Pattern Description**

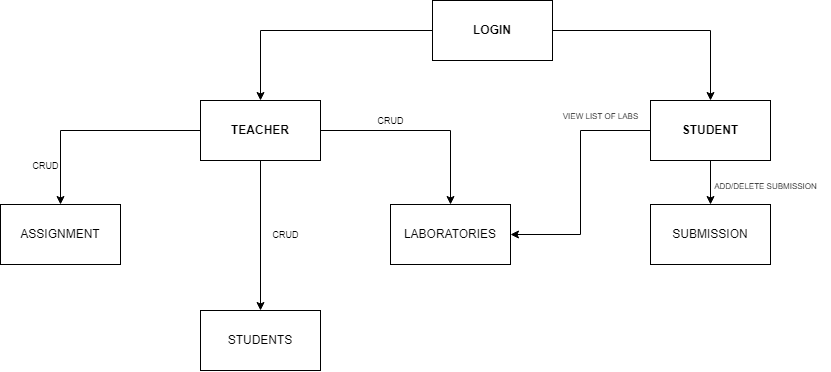
The architectural pattern used for this project is a layered one. This type of architecture respects the concept of layers of isolation which means that changes made in one layer of the architecture do not impact components in other layers and moreover, that each layer is independent of the other layers.

**3.2 Diagrams**

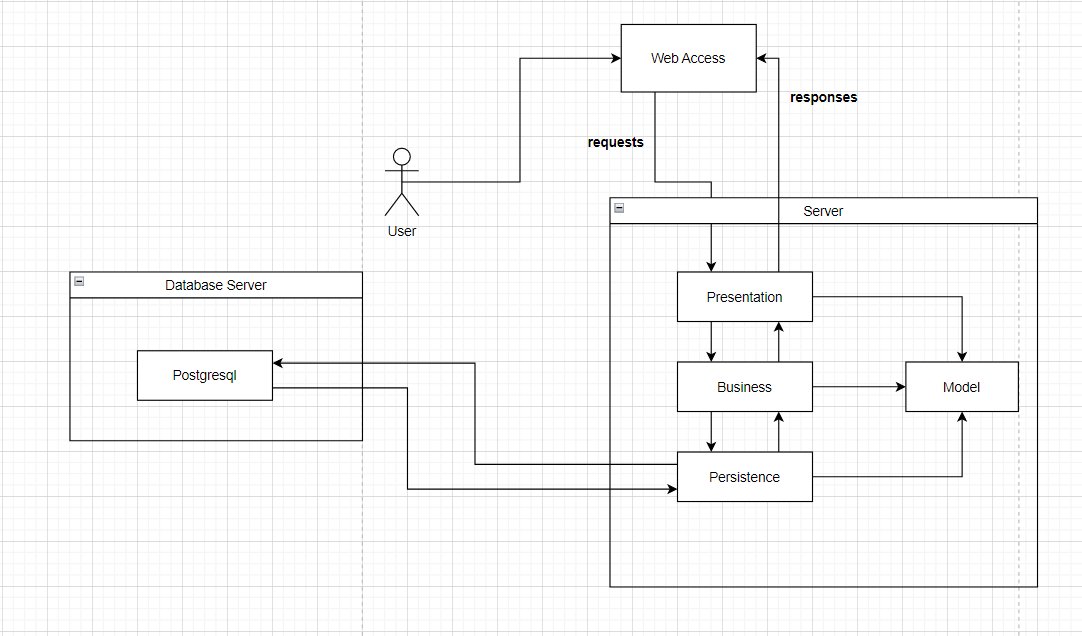
**Package diagram:**

****

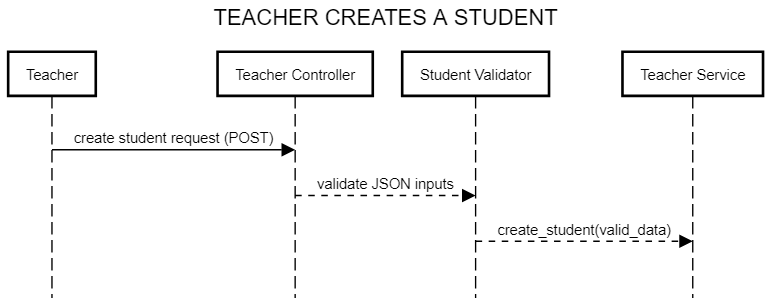
**Component diagram:**

****

**Deployment diagram:**

****

4. UML Sequence Diagrams

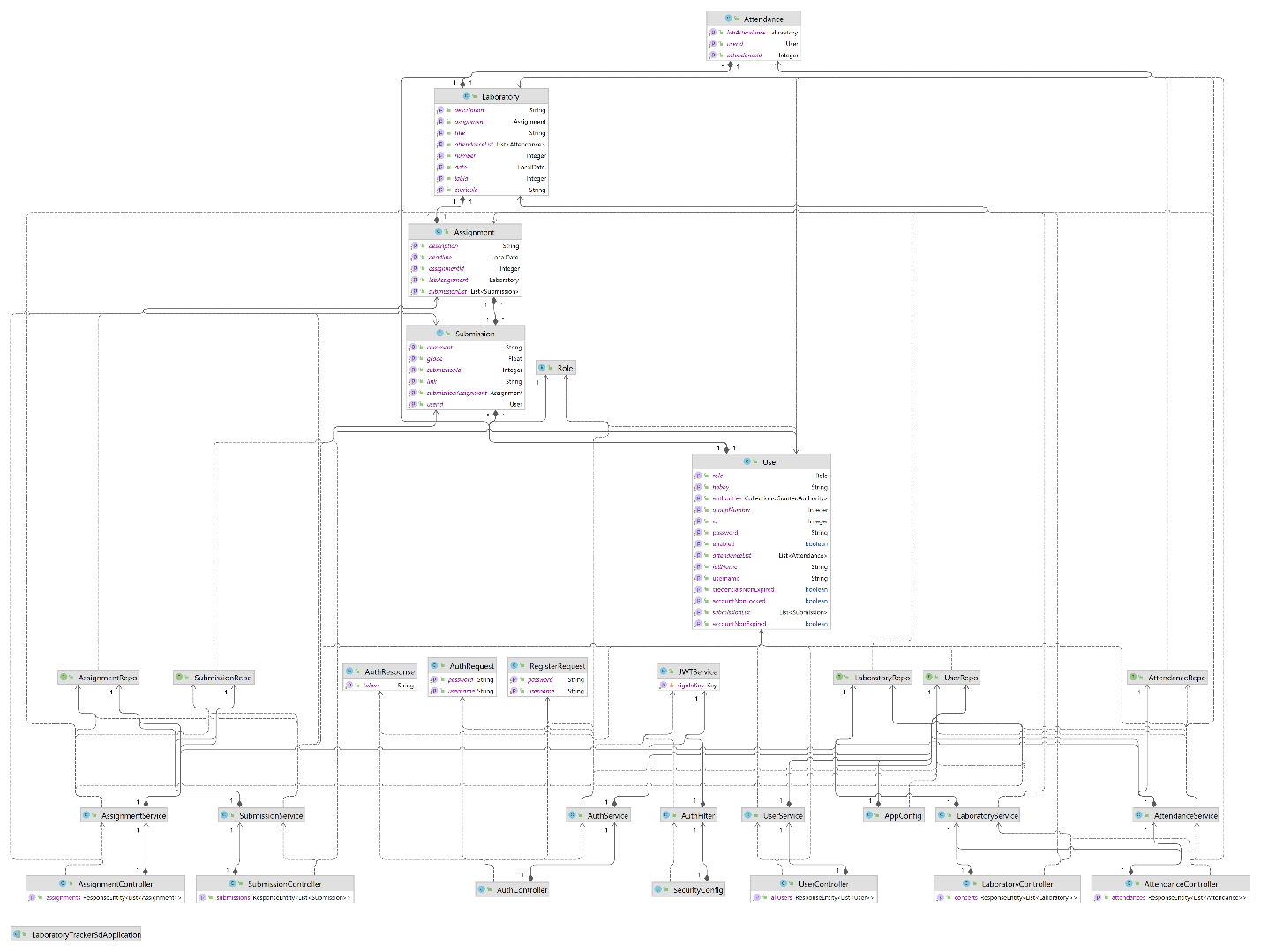


5. Class Design

**5.1 Design Patterns Description**

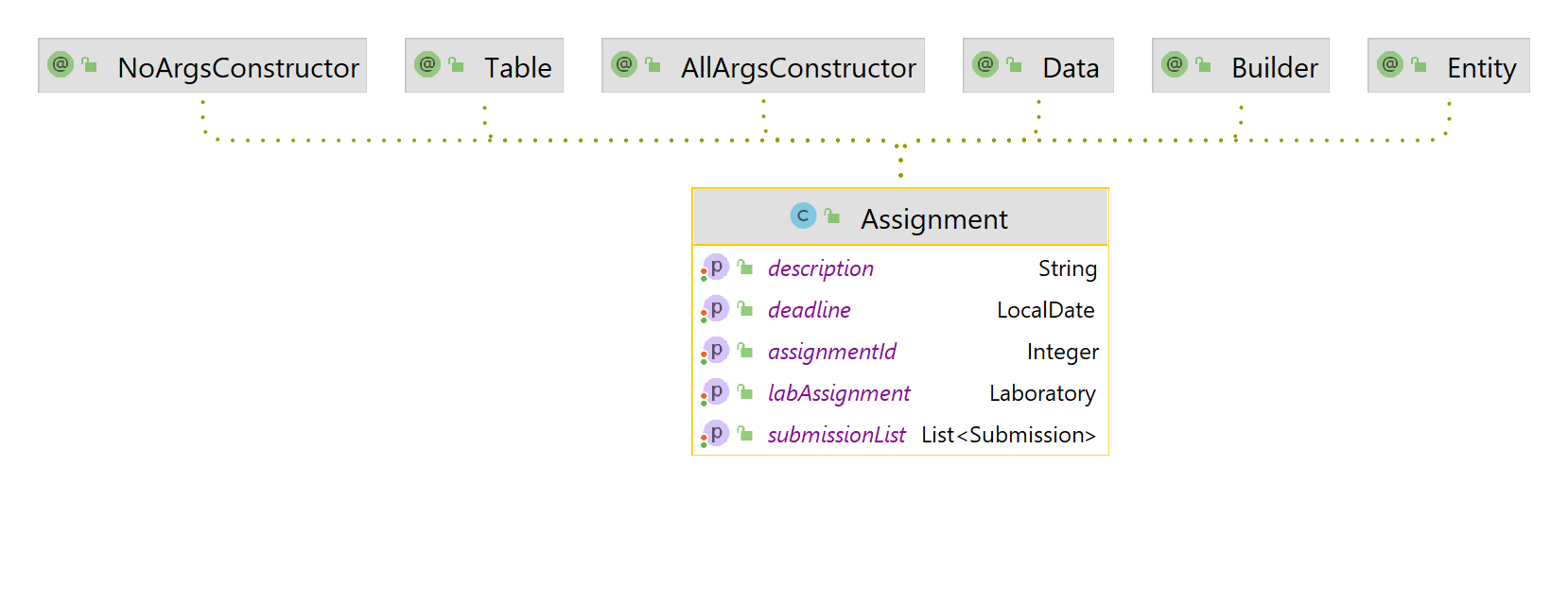
The factory method design pattern is used in this project and it defines an interface or abstract class for creating an object, but gives the decision to the subclasses to decide which class to instantiate. This design pattern allows the sub-classes to choose which type of objects to create. It also promotes the loose-coupling.

**5.2 UML Class Diagram**

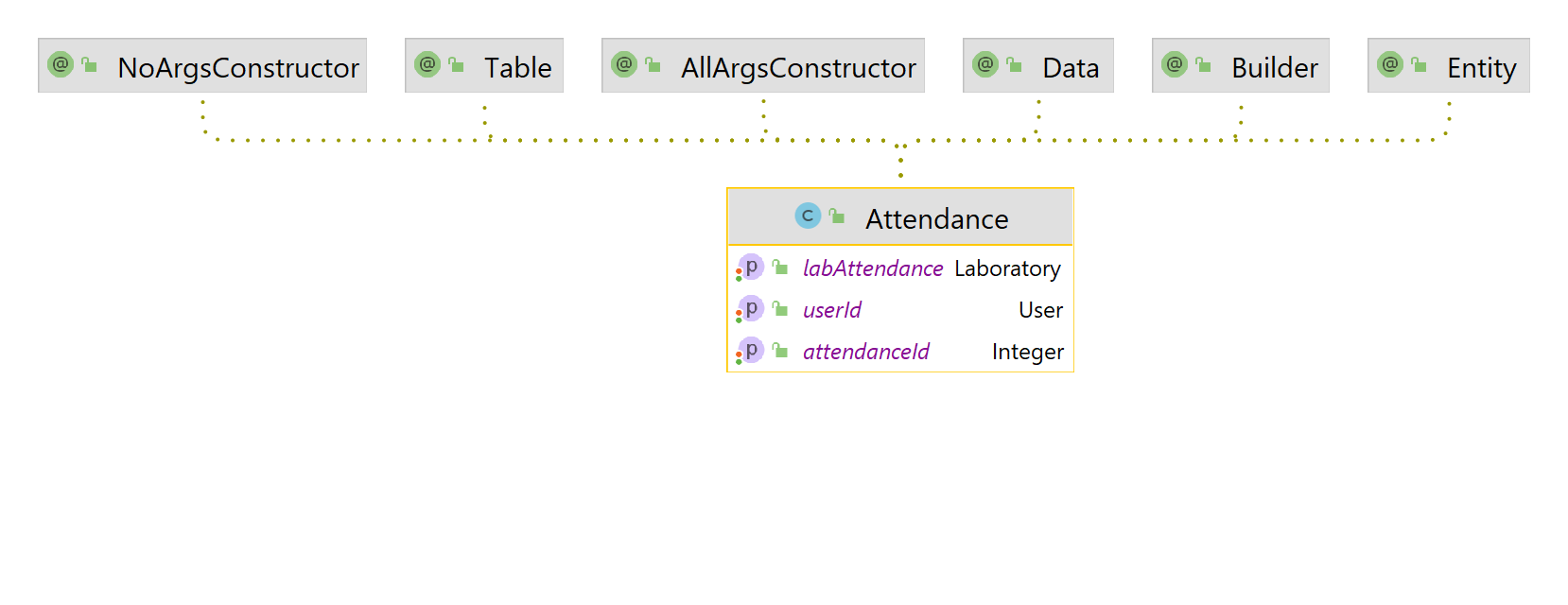


6. Data Model

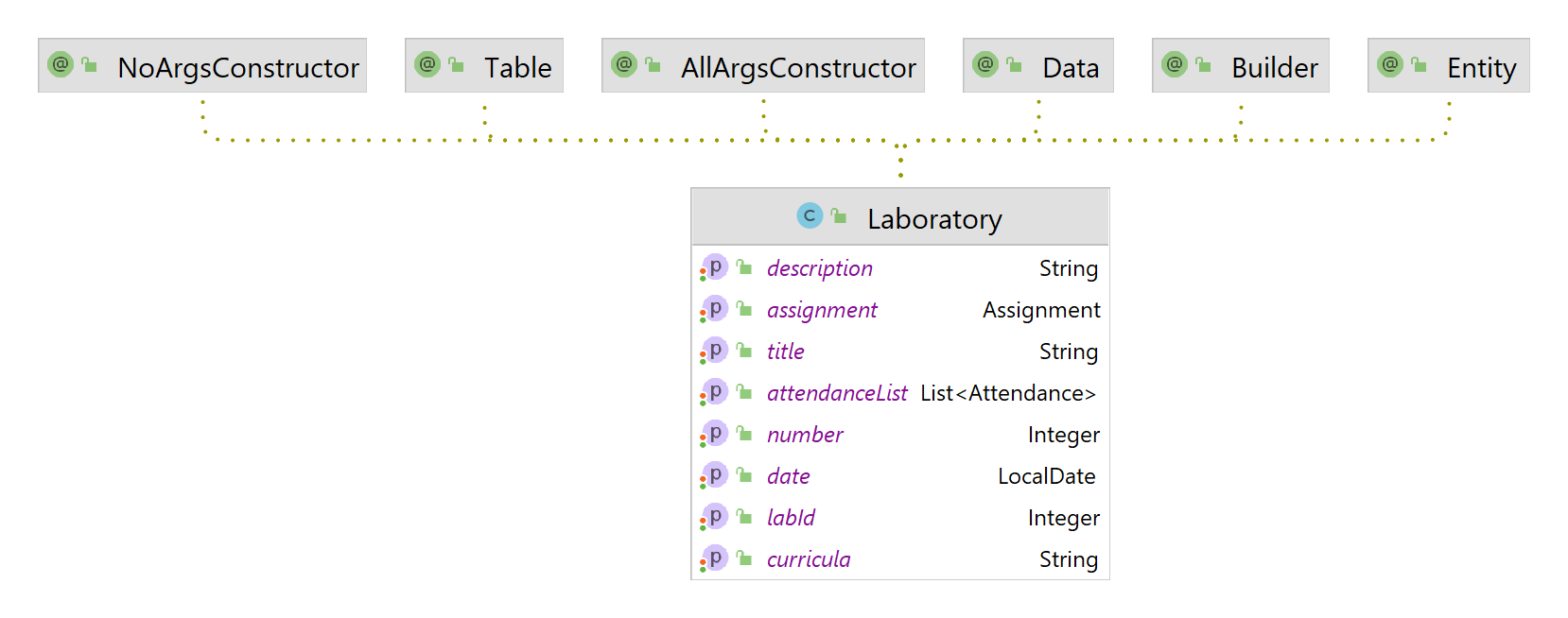
**Assignment model:**



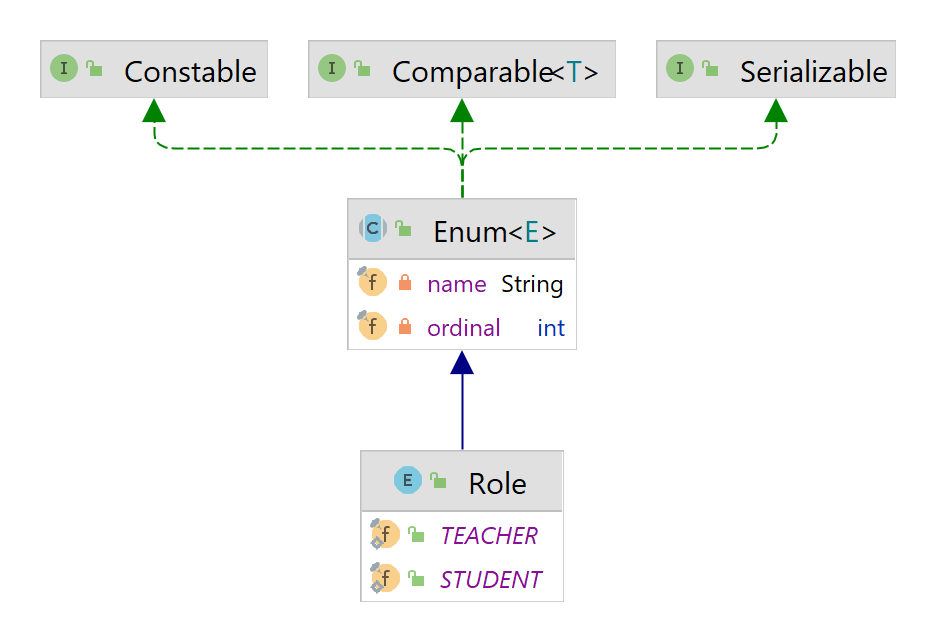
**Attendance model:**

****

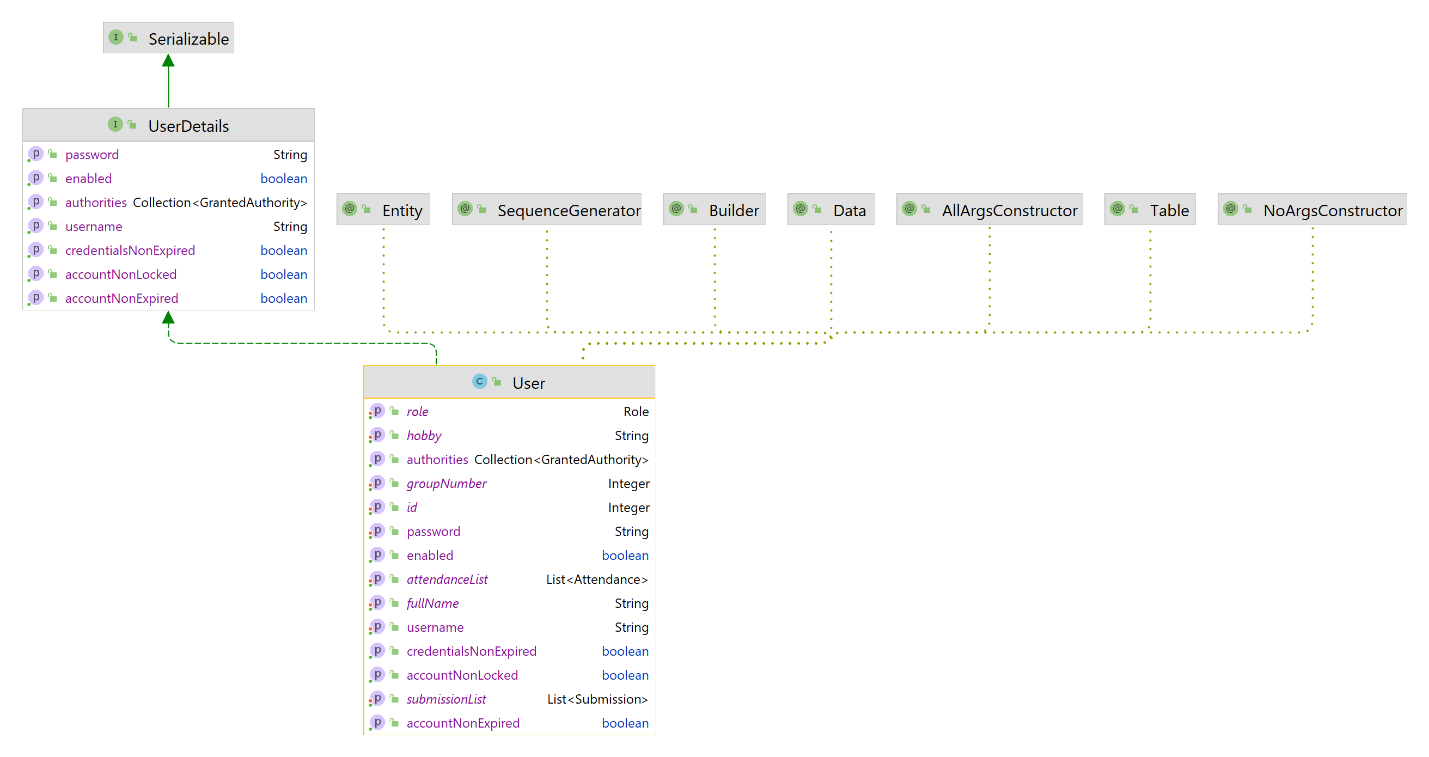
**Laboratory model:**

****

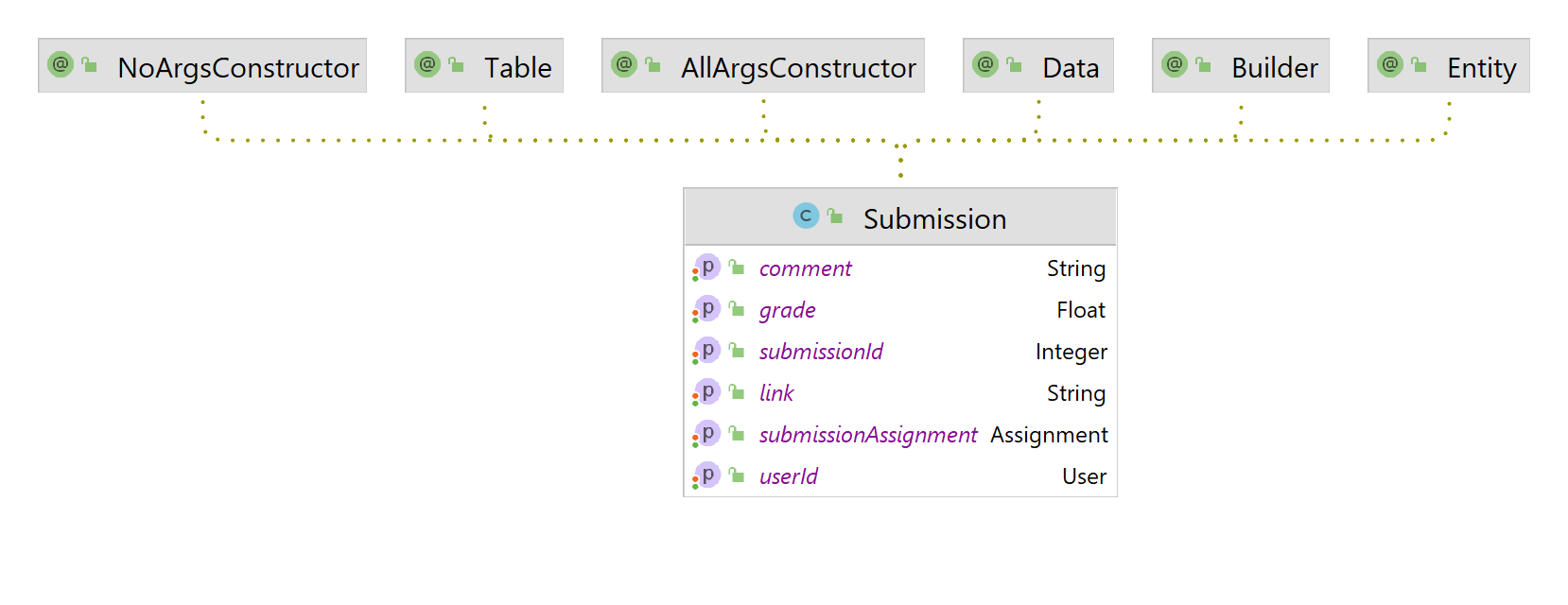
**Role model:**

****

**User model:**

****

**Submission model:**

****

7. System Testing

The testing strategy used for this application is validation testing which ensures that each operation performed is a valid one using valid inputs. In case of invalid inputs, the operations cannot be performed.

8. Bibliography

* Software Design lectures
* <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>
* <https://www.javatpoint.com/factory-method-design-pattern>
* <https://youtu.be/KxqlJblhzfI>
* <https://www.youtube.com/watch?v=9SGDpanrc8U&t=2187s>