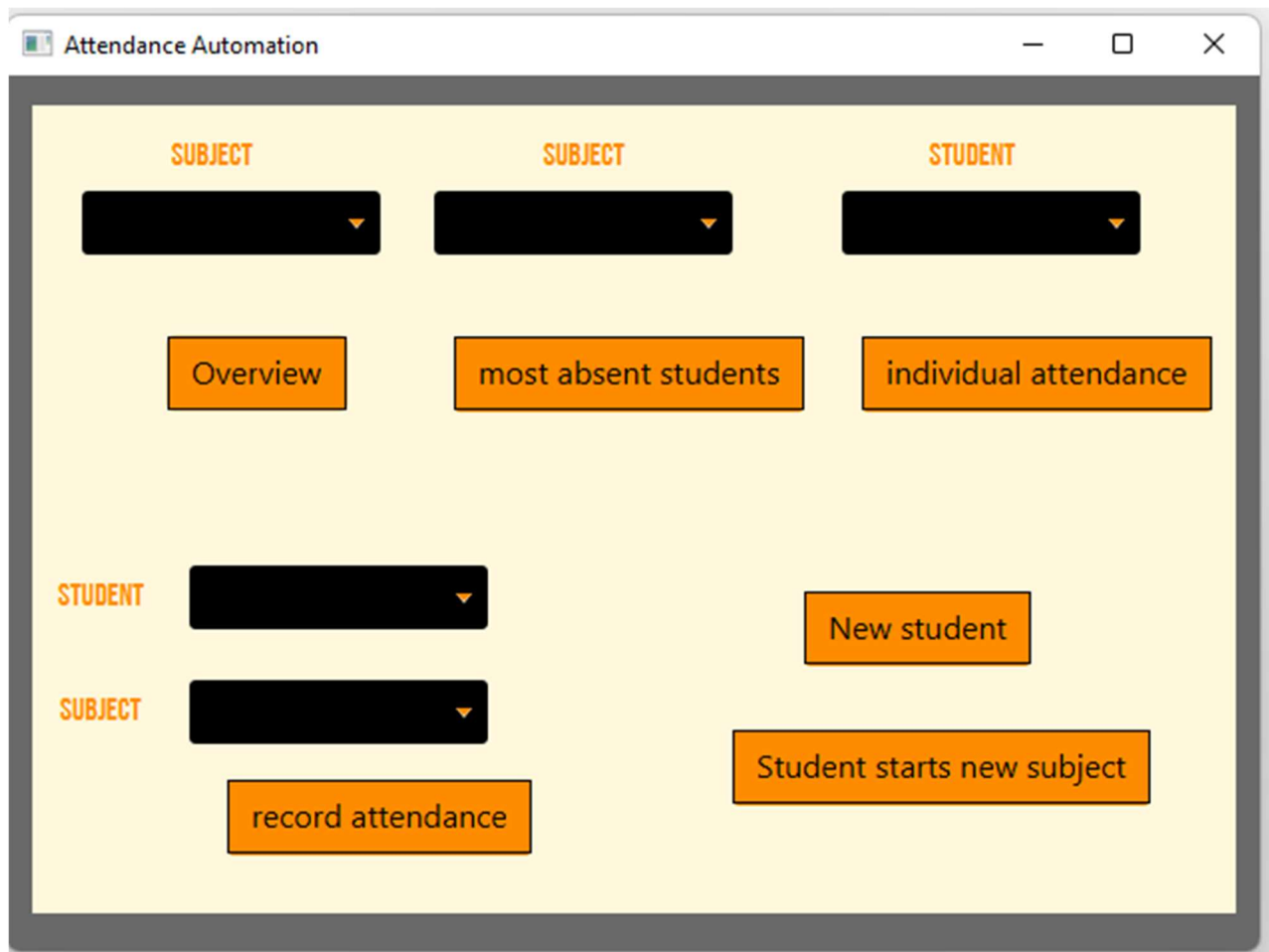


# Attendance Automation



Hand-in by  
Diogo Costa

Nikolay Nikolov

Date: 25/02/2022

GitHub repository: <https://github.com/DiogoDaCheese/Attendance-Automation>

**Attendance Assignment.  
compulsory assignment1.**

**Hand-in: Diogo Costa, Nikolay Nikolov**

## Choices:

We decided to focus our efforts on centralized version as we thought it is going to be easier and more user-friendly. The design was made in way that both student and teachers can use the same manager page.

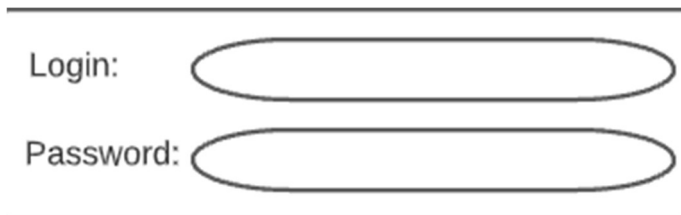
So, our thoughts are to make a Login system using the school email to identify the student and the teachers because it will be the most efficient way to use already existing Database information instead of doing everything from scratch.

Big screen in the hallway of the school will expect every student on the entry of the school simply to log in and give their attendance for the day.

## Prototyping:

Our system has just the essential screens necessary to work. Login form will be the beginning of the User Experience.

*Fig – 1 : (Login page not implemented in the demo)*

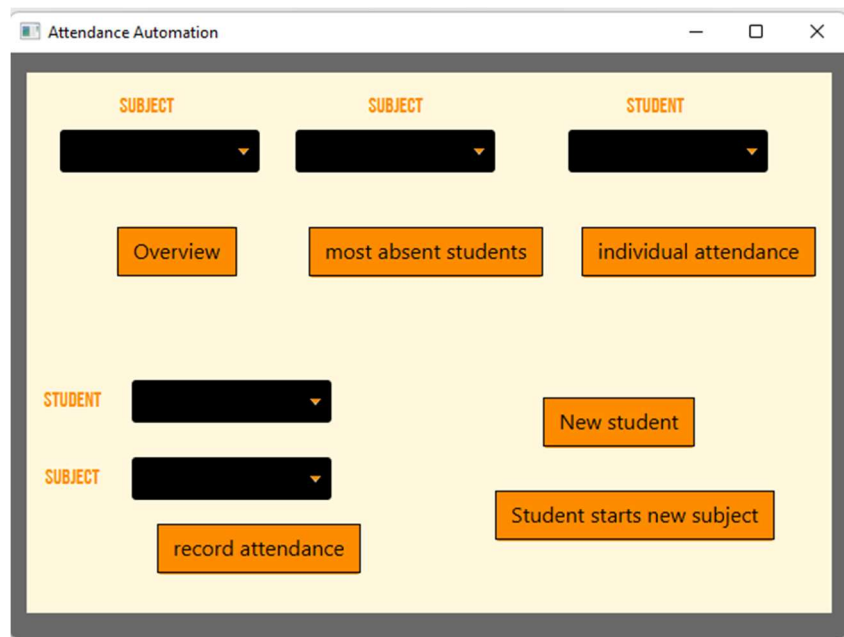


The image shows a simple login form prototype. It consists of a rectangular box containing two labels and two corresponding input fields. The first label is "Login:" followed by a horizontal oval-shaped input field. The second label is "Password:" followed by another horizontal oval-shaped input field. The entire form is enclosed in a thin black border.

The user will be redirected to the main page from which they will have access to all functions:

**Attendance Assignment.**  
**compulsory assignment1.**

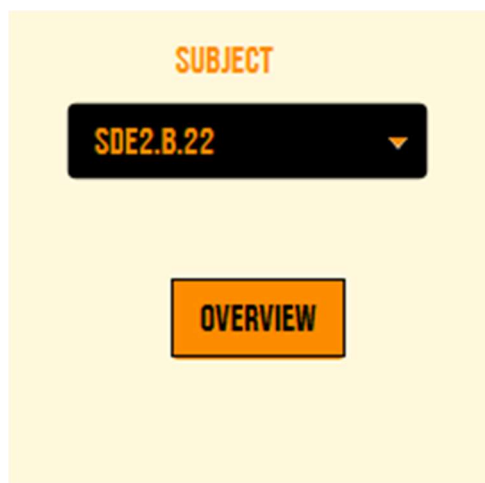
**Hand-in: Diogo Costa, Nikolay Nikolov**



## Application implementation

### 1. Overview of the Attendance

Choose a subject from the dropdown menu and click the button. Pop-up will show with the information.

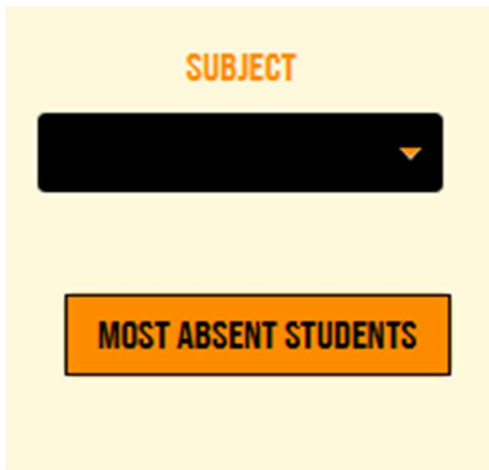


### 2. Most absent students

Choose a subject from the dropdown menu and click the button. Pop-up will show with the information.

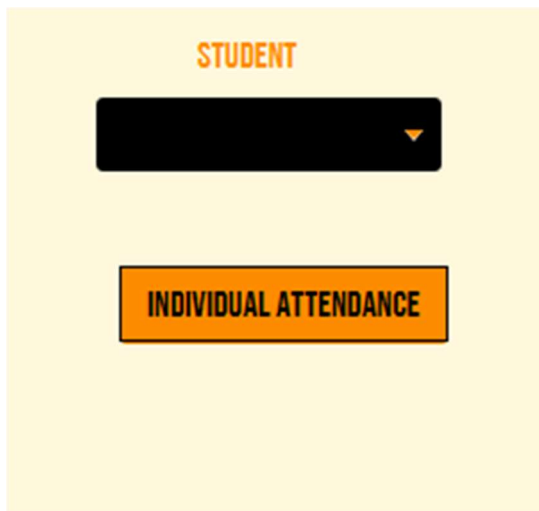
**Attendance Assignment.  
compulsory assignment1.**

**Hand-in: Diogo Costa, Nikolay Nikolov**



**3. Individual attendance**

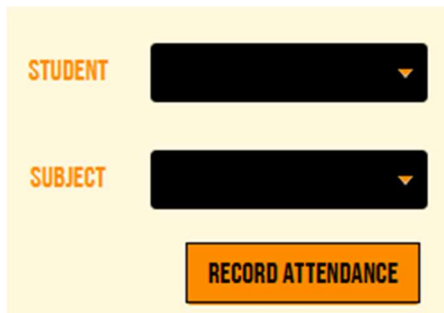
Choose a student name to see his/her attendance in all of the classes.



**4. Record Attendance**

In this sector you will be able to choose you name and subject and then record attendance as you have the option to choose yes or no.

*Fig – 6 : (Function is not implemented in the Demo)*

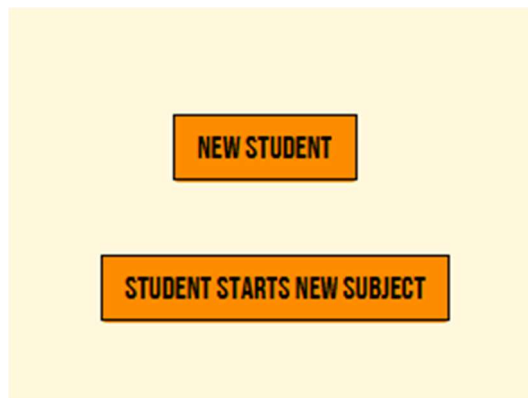


**Attendance Assignment.**  
**compulsory assignment1.**

**Hand-in: Diogo Costa, Nikolay Nikolov**

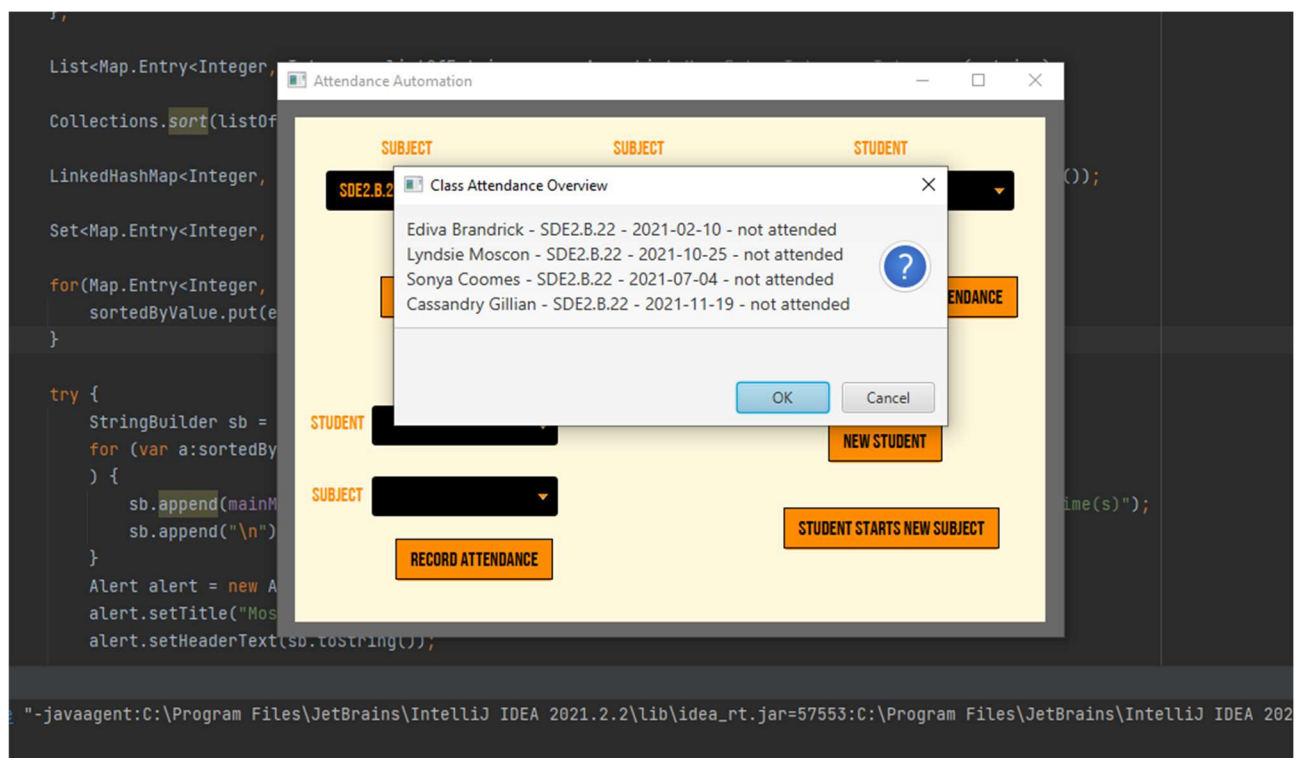
5. Creating sector

Fig – 7:(Function is not implemented in the Demo)



**Usability test**

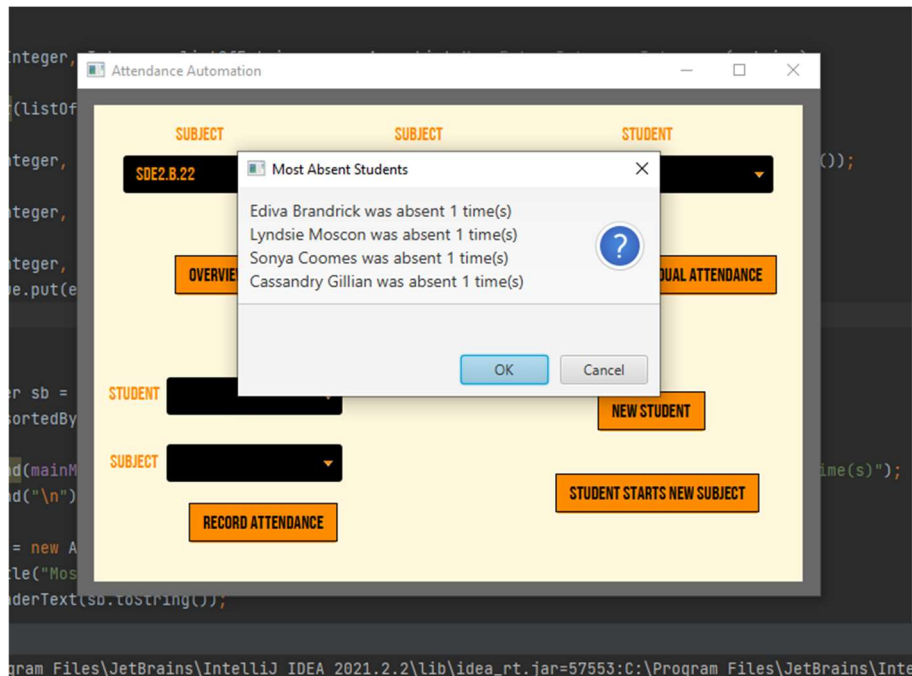
**Class Attendance Overview**



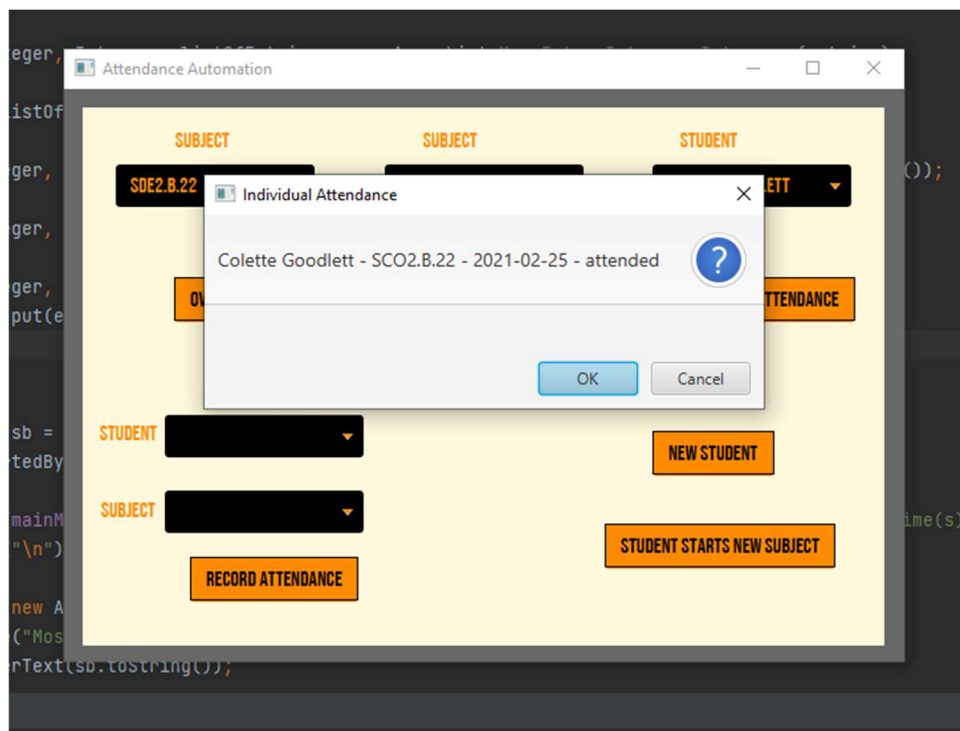
**Attendance Assignment.**  
**compulsory assignment1.**

**Hand-in: Diogo Costa, Nikolay Nikolov**

**Most Absent Students**



**Individual Attendance**



**Attendance Assignment.  
compulsory assignment1.**

**Hand-in: Diogo Costa, Nikolay Nikolov**

Argumentation for choices regarding both visual and functional fidelity

The choices we made were influenced mainly by the idea of making it as easy as possible for the user to use the program.

The functionality of the project was chosen according to the needs of the client (school), teachers and students. After discussing with them, we came to the conclusion that these will be the most important and necessary functions that will be needed. Due to our goal to make the program as simple and user-friendly as possible, we have not added anything extra.

The visualization of the project was made from 1 main window through which each user has the opportunity to easily access each functionality provided by the program. The orange and black accents were inspired by the school logo.

Discussion of the architecture in regard to the SOLID principles

SOLID stands for:

**S – Single-responsibility Principle**

**O – Open-closed Principle**

**L – Liskov Substitution Principle**

**I – Interface Segregation Principle**

**D – Dependency Inversion Principle**

For every action made in the project: buttons, choiceboxes, etc, we made a method for each one of them

Same with the access to the databases, every info that we have to bring from the database, we made a method for each, so the single-responsibility principle is met.

For the open-closed principle, we made choice boxes where you can only choose items that we added there, and whatever item you choose, the methods will deal with them individually but with the same method for every item. So, as soon as you're done with one of the items, the program will "close" that item, and when you choose a new one, it will use the same method as it did for the previous one but as a "new" item.

So, it's open for all the items, but when you got the information you need from that item, it "closes" it.

In order to satisfy the Liskov substitution principle we had the Controllers deal with the outputs, instead of doing it in the model because then we would have to change to String or other thing on the controller, so when it's done on the controller, that will be the one responsible to print the output all the time.

**Attendance Assignment.  
compulsory assignment1.**

**Hand-in: Diogo Costa, Nikolay Nikolov**

So, the values will go smoothly from the models to the controllers, and then they will be dealt with on the controller in order to be seen (output) by the user

In order to satisfy the interface segregation principle, we choose to use choiceboxes, then in this way the user can choose the items available for each "button" attached to it. So, there is no "waste", like the user be able to choose an item from a certain choicebox but then have no use for it or get no information from it

This code establishes that both the high-level and low-level modules depend on abstraction, because we used a mock-data database, the project is forced to use only information from the database, but we also implemented on the project buttons to create new students, and give students new subjects (although this doesn't work for now), also we made a database connector which is connected to a database program called H2, but it can be also used with other databases, so we have both high-level and low-level modules depending on abstractions.