PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Object Oriented Analysis and Design using Java (UE20CS352)

PROJECT TITLE: ASSIGNMENT SUBMISSION PORTAL

OOAD PROJECT

TEAM :	**NIREEKSHA D RAI**	**PES1UG20CS668**
	JAGRUTHI K J	**PES1UG20CS650**
	CHANDANA CH	**PES1UG20CS636**
	NIDHI SHETH	**PES1UG20CS666**

1. SYNOPSIS:

The " Assignment Submission Portal" is a web-based application developed using Java and Spring Boot that allows students to submit their assignments online. The system aims to simplify the process of submitting assignments for both students and teachers.

The application allows students to log in and upload their assignments in .txt format. Once the assignment is uploaded, students can see the submitted file indicating a successful submission. The student can also view his submissions under the ‘My submissions’ section.

Teachers can access the submitted assignments through the portal as well as view all the submissions that were made. The system allows teachers to download the submitted assignments, view them, and grade them accordingly.

The application includes features such as a user-friendly interface, authentication, authorization, and secure file upload and download functionalities. The system also allows students to undo a submission incase of a wrong file upload.

Overall, the Assignment Submission Portal simplifies the submissions,reduces paperwork, and enhances the overall efficiency of the assignment submission system.

2. Architecture Pattern:

MVC Pattern: MVC stands for Model-View-Controller, which is a widely-used design pattern in software engineering. The main idea behind MVC is to separate an application's data, user interface, and control logic into three distinct components.The MVC pattern provides several benefits, including modularity, maintainability, and flexibility. By separating the concerns of an application into distinct components, developers can make changes to one component without affecting the others. This can make the code easier to maintain and test.

Use in our code:

Model: Class **DBFile,** *class ,***Teacher,** *class* **User**
Controller: class **FileController,** *class* **LoginController,** ***class*** **RegisterationController**

3. DESIGN PATTERNS :

1.Factory method: The Factory Design Pattern is a creational pattern that aims to provide an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.It provides a flexible and scalable solution for object creation. It separates the object creation logic from the client code and allows new classes to be added without affecting the existing code.
Use in our code: Class **FactoryUserService**

***T**he class has two instance variables, teacherService and studentService, both of which are autowired using the @Autowired annotation. These variables represent the TeacherService and StudentService classes, which are responsible for saving teacher and student users, respectively.*

The saveUser method takes four parameters: email, password, name, and user. The user parameter represents the type of user that needs to be saved (i.e., "teacher" or "student"). The method then checks the value of the user parameter using an if-else statement to determine whether to create and save a teacher or student user.

If the user parameter is "teacher", the method calls the saveLog method of the teacherService instance variable and passes the email, password, and name parameters to create and save a teacher user. If the user parameter is "student", the method calls the saveLog method of the studentService instance variable and passes the email, password, and name parameters to create and save a student user.

2.Singleton Pattern: The Singleton Design Pattern is a creational pattern that ensures that a class has only one instance, and provides a global point of access to that instance.It provides a flexible and scalable solution for ensuring that a class has only one instance. It provides global access to that instance and uses lazy initialization and thread safety to improve performance and avoid issues with multiple instances.

Use in our code: Class **DBFile**

The DBFile class has instance variables for storing the id, fileName, fileType, and data of a file, as well as the student_id of the file owner. These variables are used to store the metadata and content of a file that is uploaded to the system.

The DBFile class has a private constructor that takes four parameters: fileName, fileType, data, and student_id. These parameters represent the metadata and content of a file that is uploaded to the system. This constructor is used to create a new DBFile object when the getInstance method is called for the first time.

The getInstance method checks if the instance variable is null. If it is not null, it means that an instance of the DBFile class has already been created, and the method simply returns the existing instance. If the instance variable is null, the method creates a new instance of the DBFile class using the private constructor, and sets the instance variable to the new instance before returning it.

3.Memento Pattern: The Memento Design Pattern is a behavioural design pattern that allows an object to save and restore its internal state without violating encapsulation.

Use in our code: Function **deleteFile**

This function lets the user undo a submission in case of a mistake and go back to the previous state from the current state.

It takes a fileId as a parameter, which is the unique identifier for the file that needs to be deleted. The method first creates a SQL query string using the fileId and then executes the query using the jdbcTemplate object. The query uses the SQL "delete" statement to delete the file record from the "files" table in the database where the id matches the fileId parameter.

4. DESIGN PRINCIPLES :

1.Single Responsibility Principle: The Single Responsibility Principle (SRP) is a design principle that states that a class should have only one reason to change. In other words, a class should have only one responsibility or job to do.By following the SRP, a class becomes more cohesive and less coupled with other classes. This means that changes made to one class are less likely to affect other classes, which reduces the risk of introducing bugs into the system.

2.Open Close Principle: The Open-Closed Principle (OCP) is a design principle that states that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification

In other words, the behaviour of a software entity can be extended without modifying its source code. This is typically achieved through the use of abstraction, interfaces, and inheritance. By doing so, we can add new features or change the behaviour of an existing feature without modifying the existing codebase.

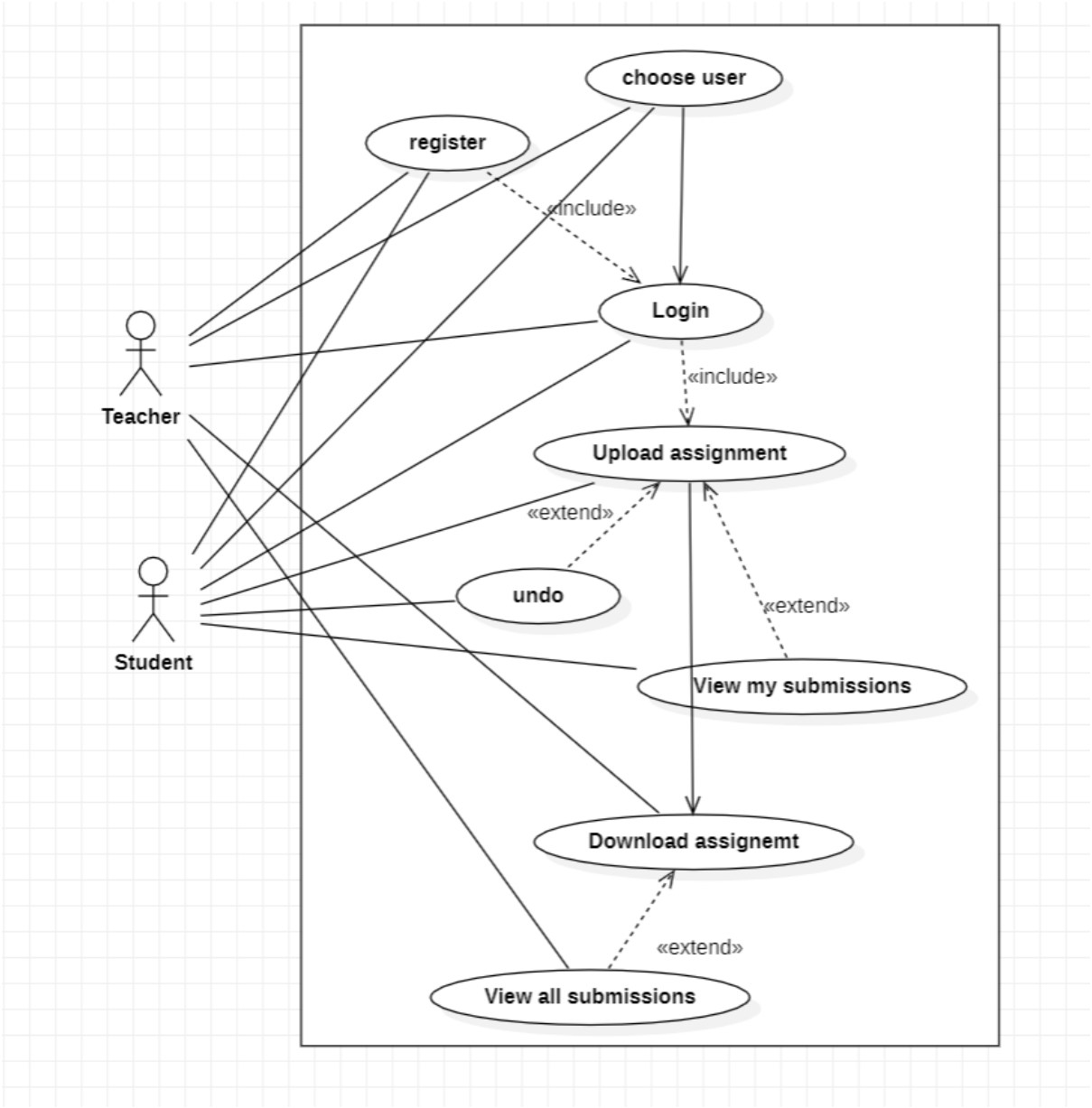
3. Dependency Inversion Principle: The Dependency Inversion Principle (DIP) is a design principle that states that high-level modules should not depend on low-level modules. Instead, both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

This means that classes or modules should depend on abstractions, such as interfaces or abstract classes, rather than concrete implementations. This allows for greater flexibility and modularity in the design of the software, as changes to concrete implementations do not affect the behaviour of higher-level modules.

4.Interface Segregation Principle: The Interface Segregation Principle (ISP) is a design principle that states that a class should not be forced to implement interfaces that it does not use. Instead, interfaces should be segregated into smaller and more focused ones, each addressing a specific behaviour.

In other words, instead of having a large and generic interface, it is better to have multiple smaller interfaces that are more specific to the needs of each class. This way, a class only needs to implement the interfaces that it needs, and not those that it does not.

5. USE CASE DIAGRAM:



Description :

Actors:

- Student: A user who submits an assignment through the portal.
- Teacher: A user who accesses and downloads assignments through the portal.

Use cases:

- Choose user: Login to the portal either as a teacher or as a user. Each role has user specific actions and permissions granted.
  Actors: Student,Teacher

- Register: A new user can register itself in the portal as a student or a teacher using an email ID and password.
  Actors: Student,Teacher

- Login: Both the student and teacher actors can log in to the portal using their credentials.
  Actors: Student,Teacher

- Upload Assignment: The student actor can upload an assignment through the portal.
  Actors: Student
  Pre-condition: Student must be a registered user.

- Undo: Students can delete the submission they just made in case of a mistake in uploading the assignment.
  Actors: Student
  Pre-condition: Assignment can be deleted only before any further submissions are made.

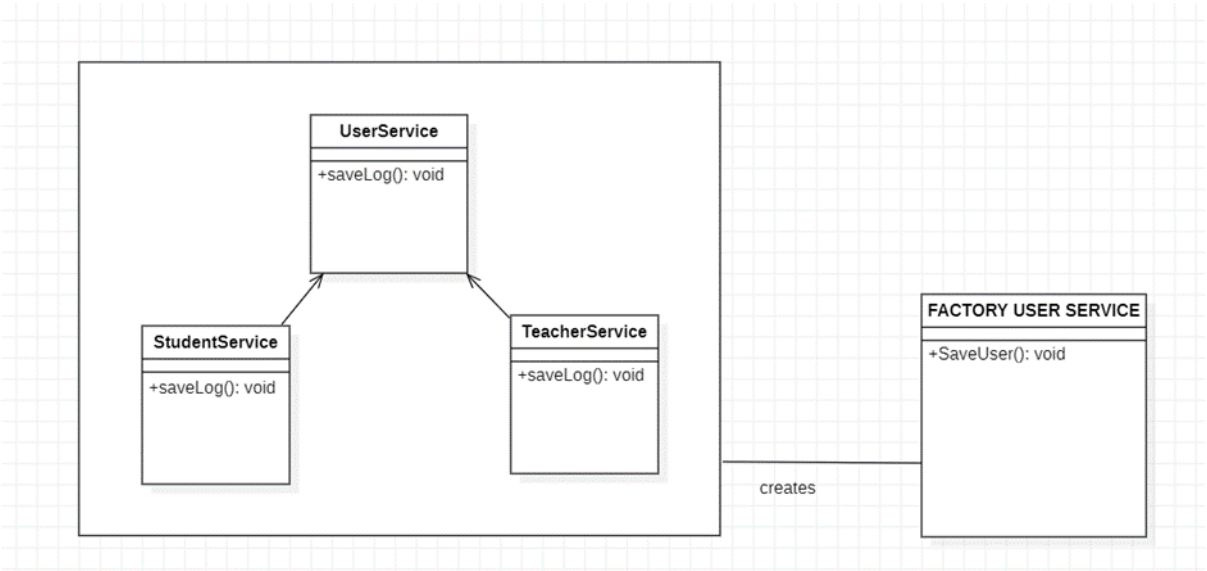
- View my submissions: Student can see all the files he has uploaded for that assignment.
  Actors: Student

- Download Assignments: The teacher actor can download the assignments to view and give grades accordingly.
  Actors: Teacher
  Pre-condition: Teacher must be a registered user.

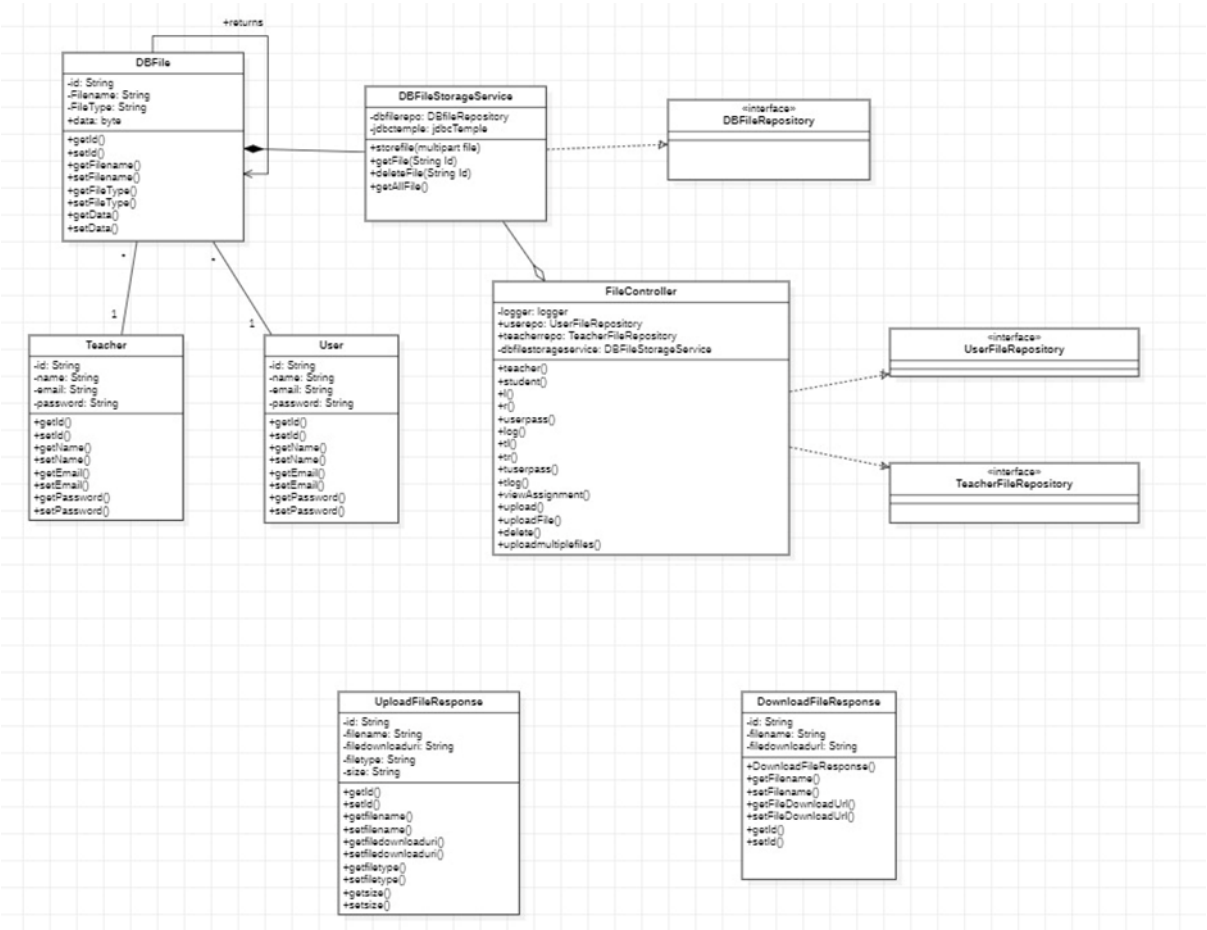
- View Assignments: The teacher actor can view the assignments that have been submitted through the portal.
  Actors: Teacher

6. CLASS DIAGRAMS:

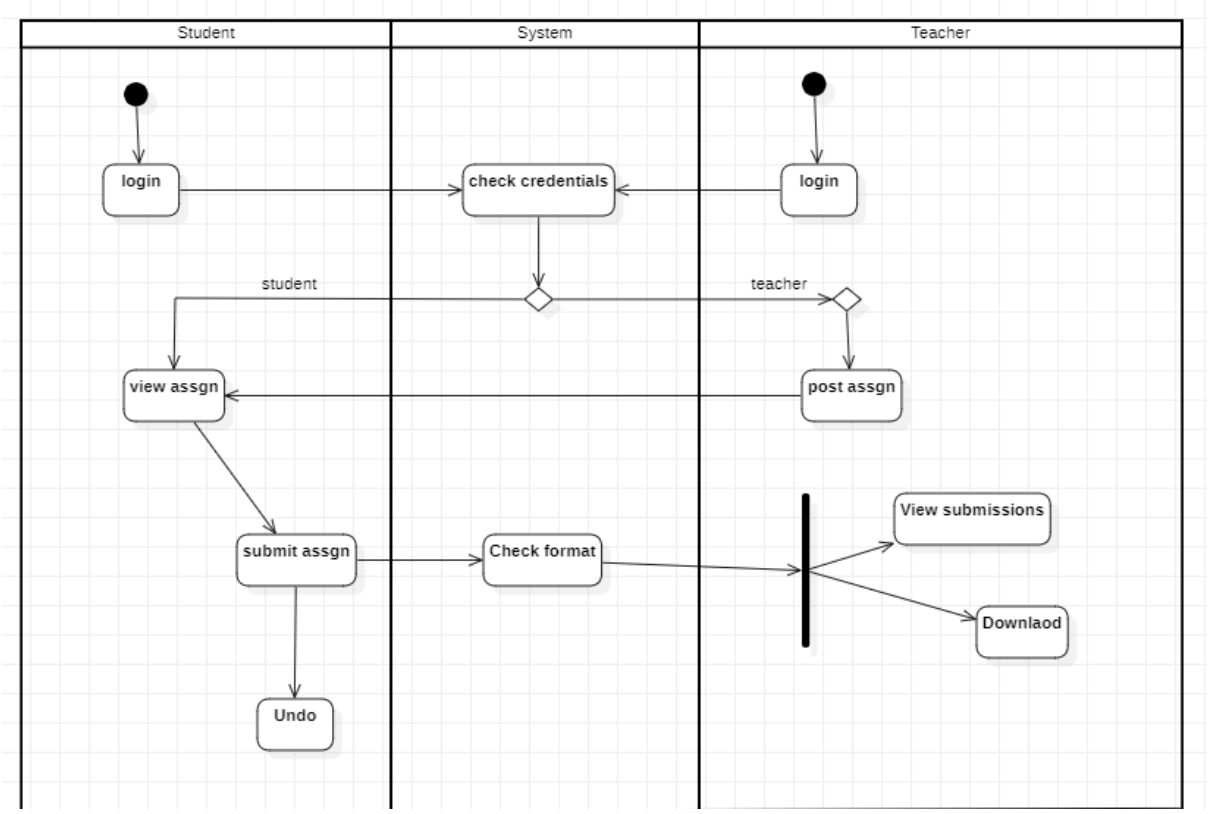
FOR user login:



For file upload and download:



7. ACTIVITY AND STATE DIAGRAM:

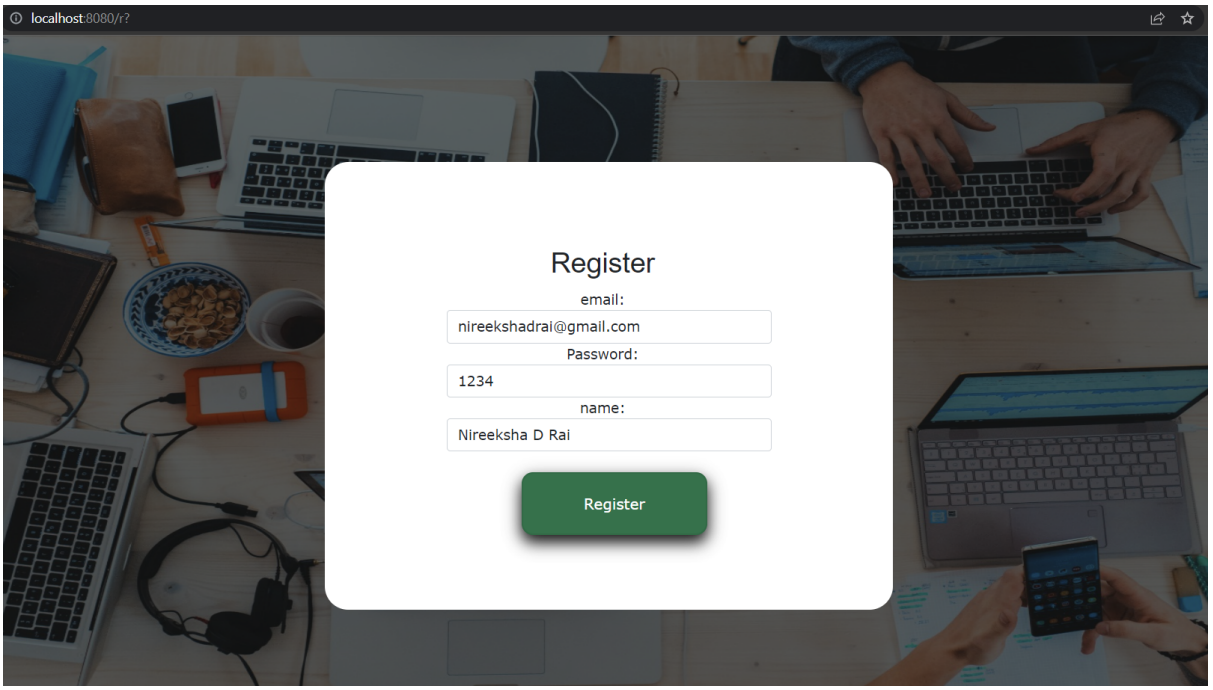


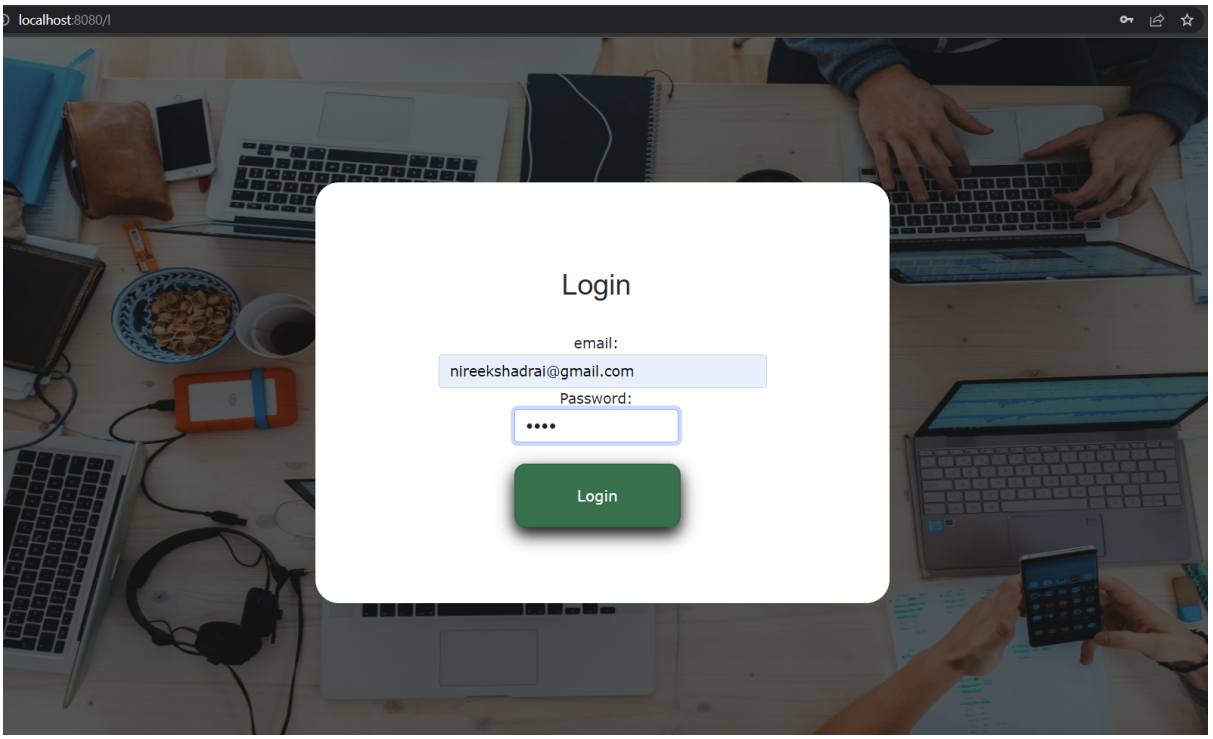
8. Github link to code base:
   https://github.com/Nireekshadrai/Assignment-Submission-Portal-OOAD

9. Individual contribution:
   1. Nireeksha D Rai: SpringBoot setup,Upload assignment, undo my submission, view my submission
   2. Jagruthi K J: Teacher Signup, Teacher login, CSS
   3. Nidhi S Sheth: Download assignment, View all submissions
   4. Chandana CH: Student Signup, Student login

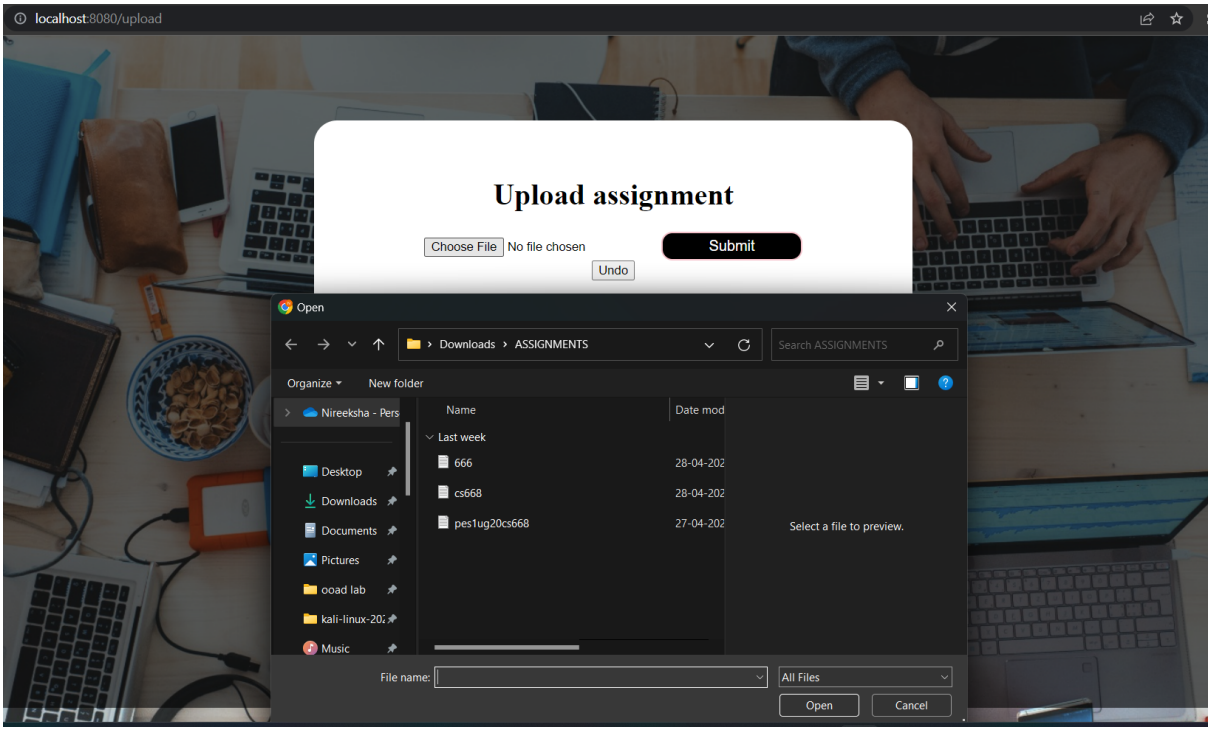
10. Screenshots with input values populated and output shown

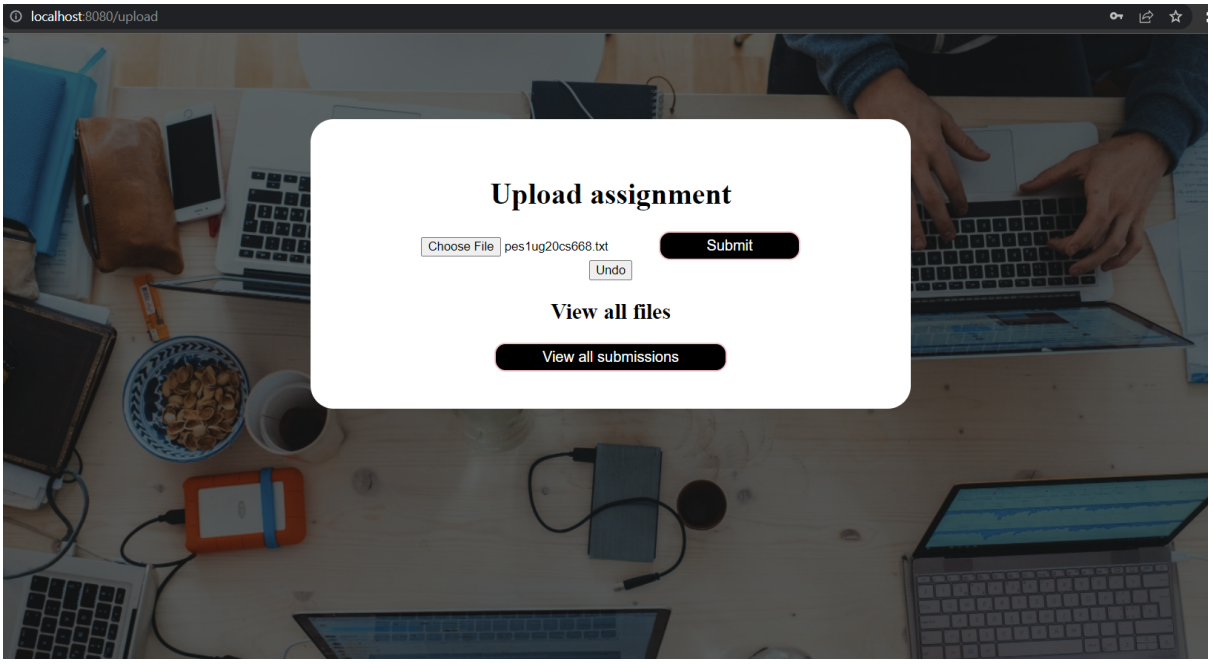
Student signup and login:



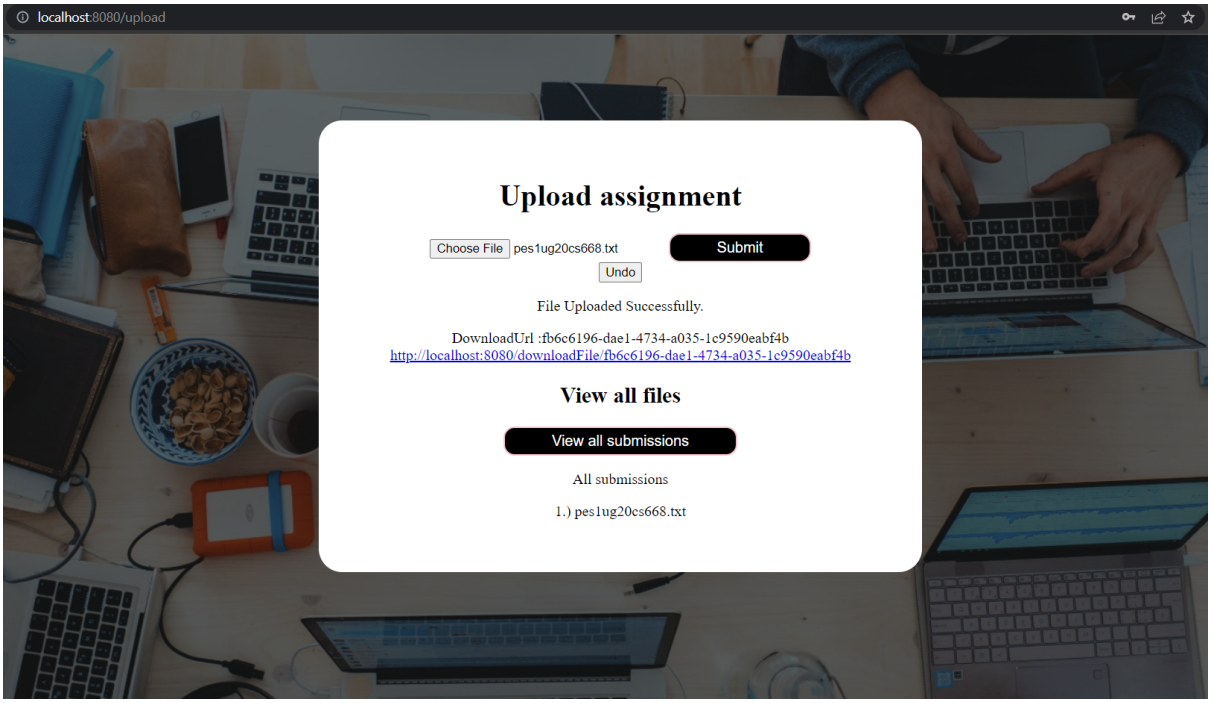


Upload file:
Choose file

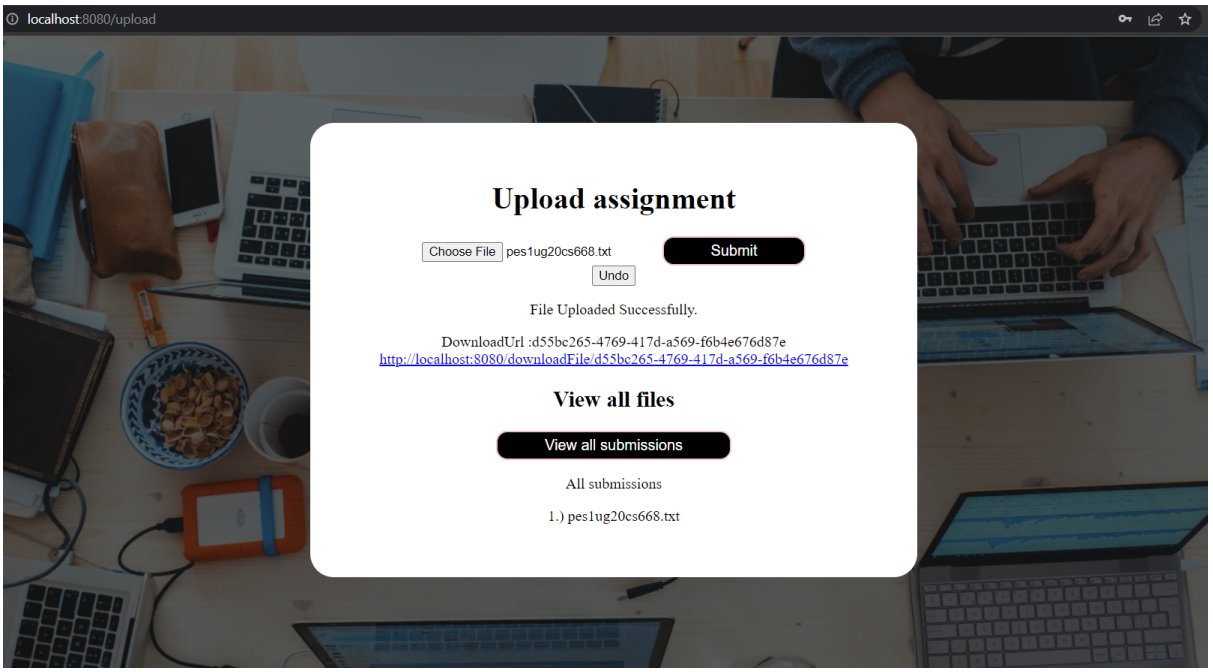




Submit assignment:



View my submission:



Undo submission:

Before

```
3 •  select * from files;
4
5 •  select * from users;
6 •  select * from teachers;
7 •  truncate table files;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	data	file_name	file_type	student_id
fb6c6196-dae1-4734-a035-1c9590eabf4b	BLOB	pes1ug20cs668.txt	text/plain	2e1ee4d1-9910-43e3-9664-187dc4d6bf54
NULL	NULL	NULL	NULL	NULL

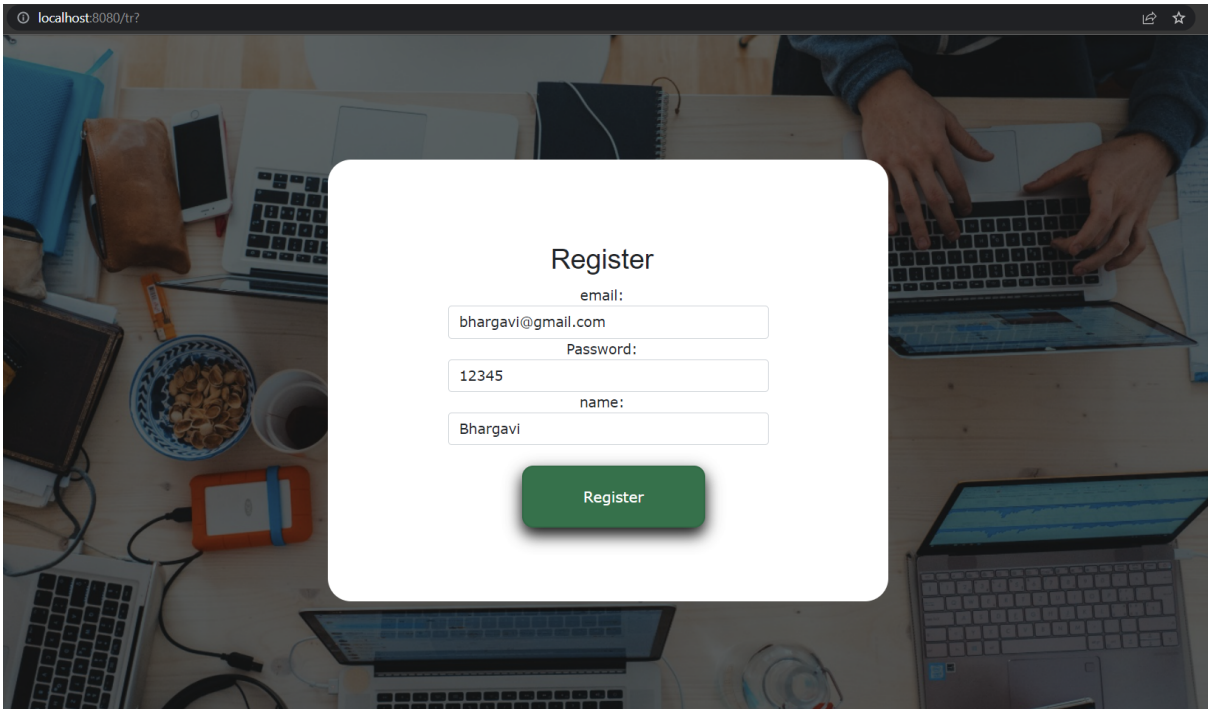
After

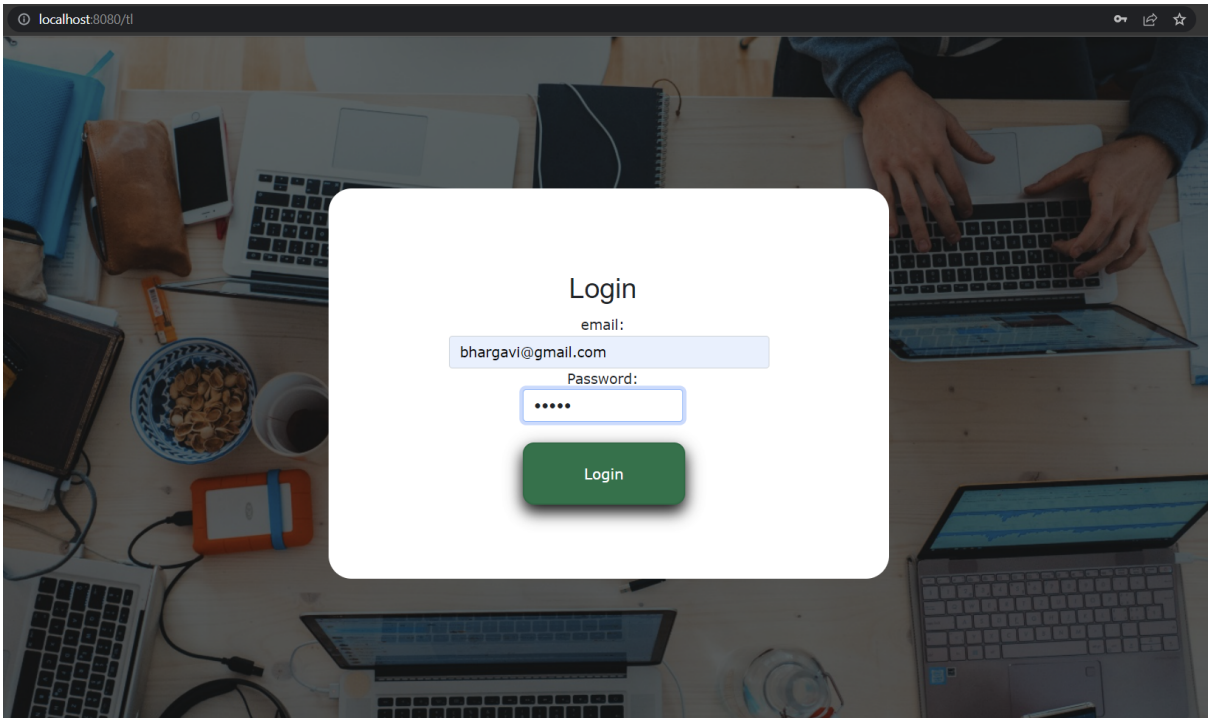
```
3 •  select * from files;
4
5 •  select * from users;
6 •  select * from teachers;
7 •  truncate table files;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

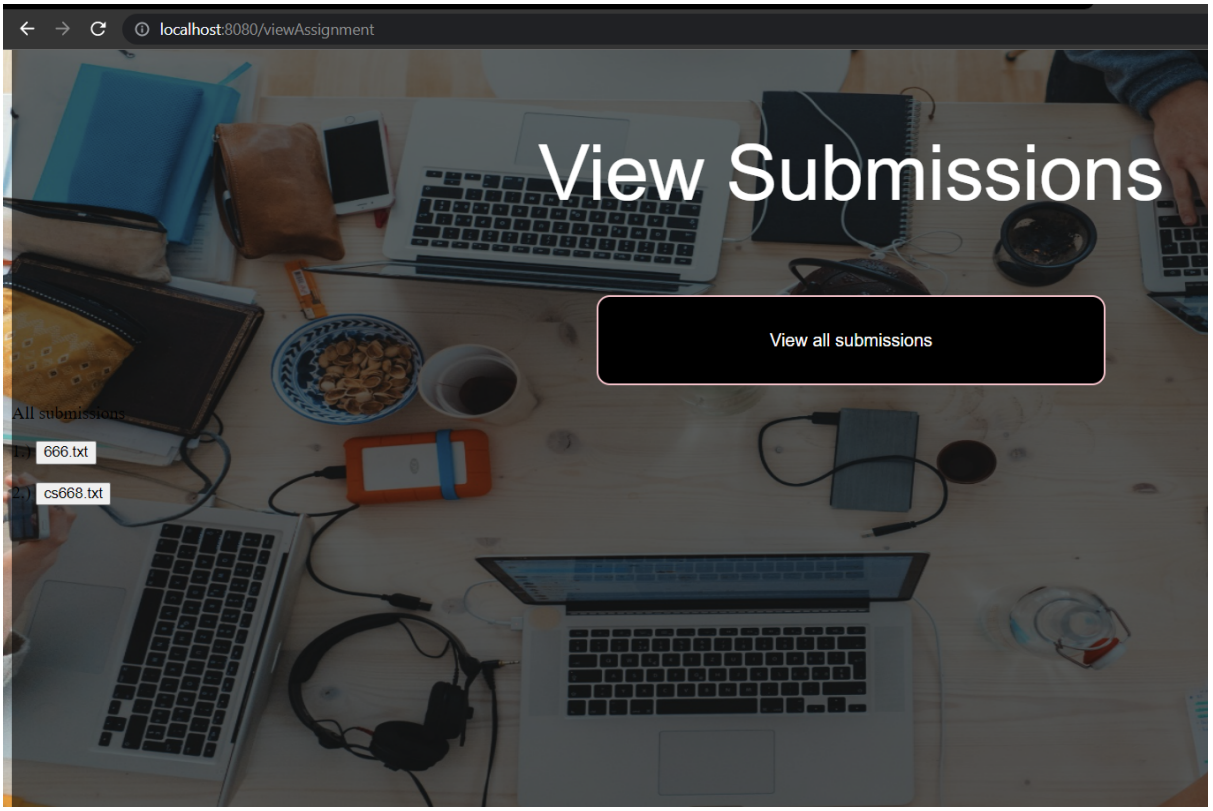
id	data	file_name	file_type	student_id
NULL	NULL	NULL	NULL	NULL

Teacher Signup and login:





View All submissions:



Download submitted assignment: