**JAVA**

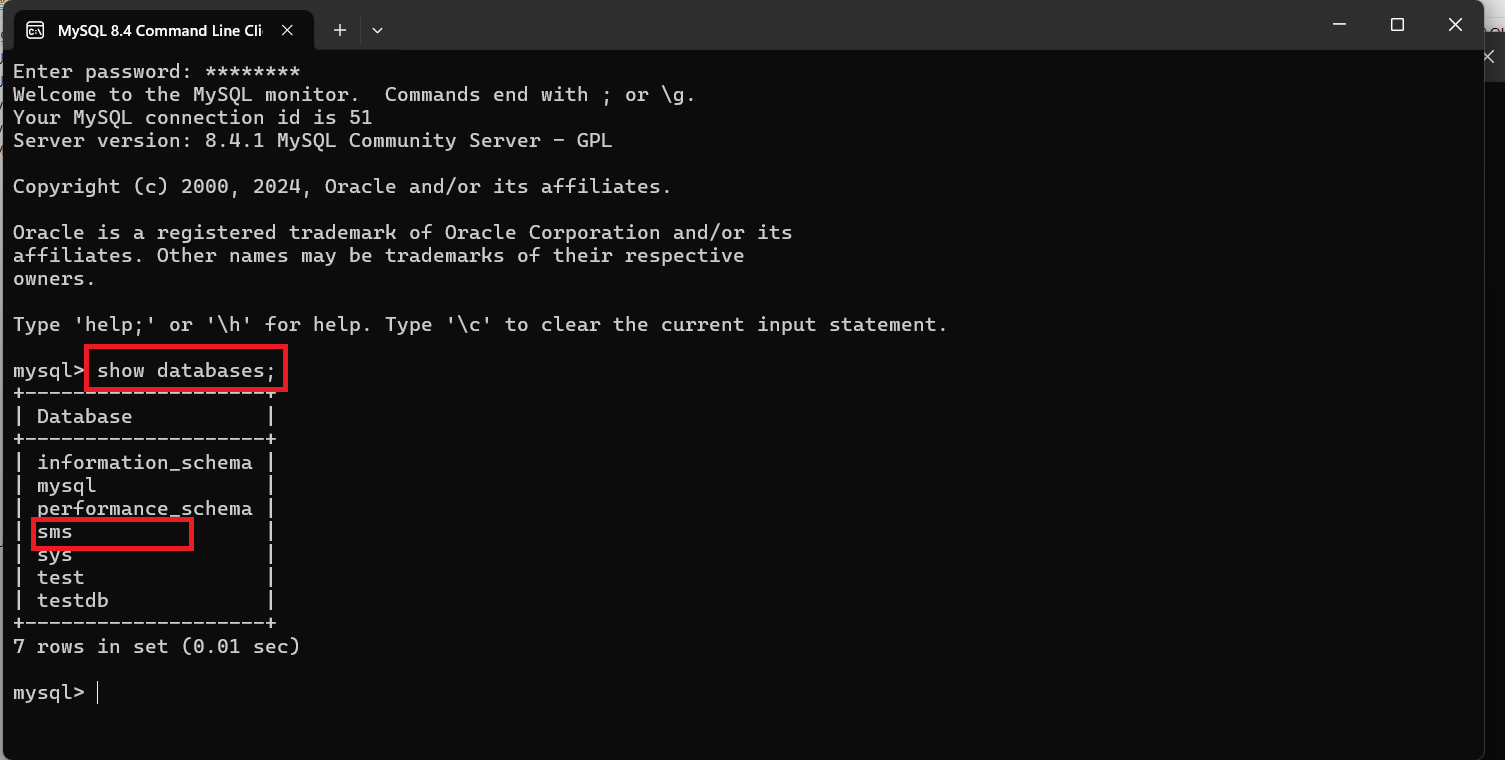
**Student ID : C0930321**

**Student Name : Shreejana Shrestha**

# # Student Management System

## # Database and its connectivity to the SMS project

### # showing database

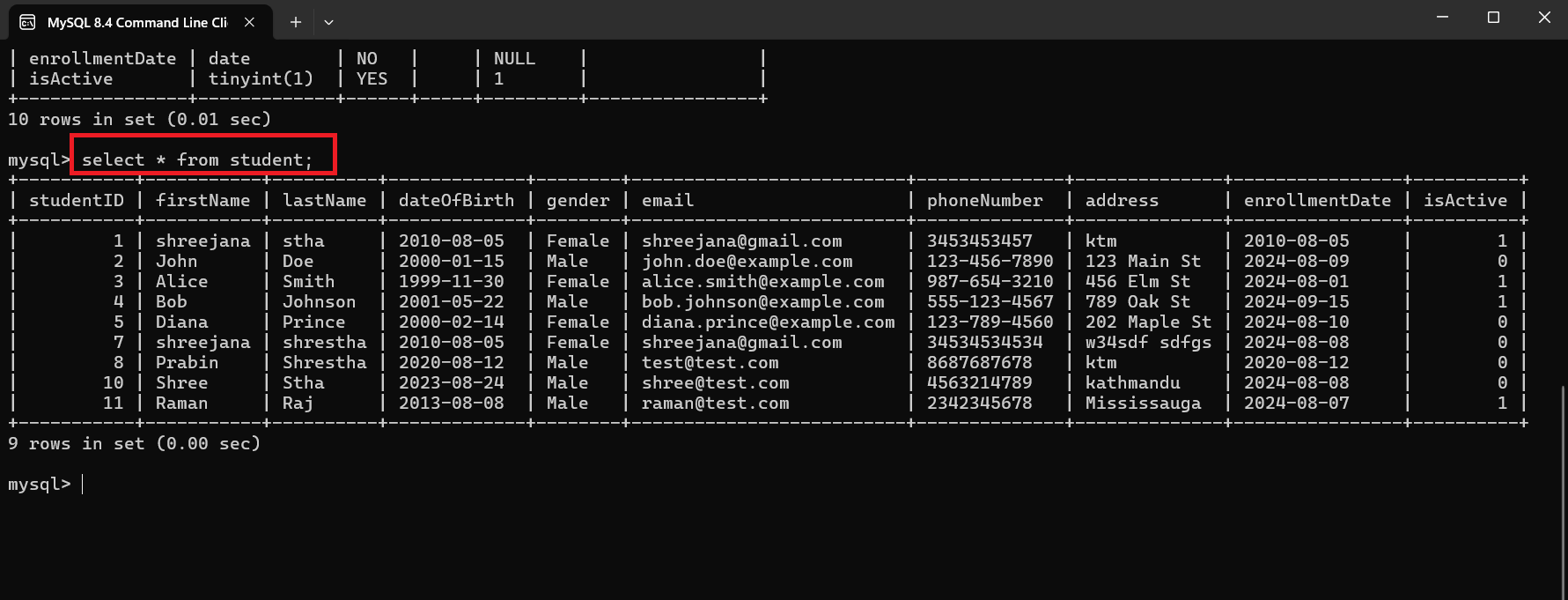


### # Structure of student table

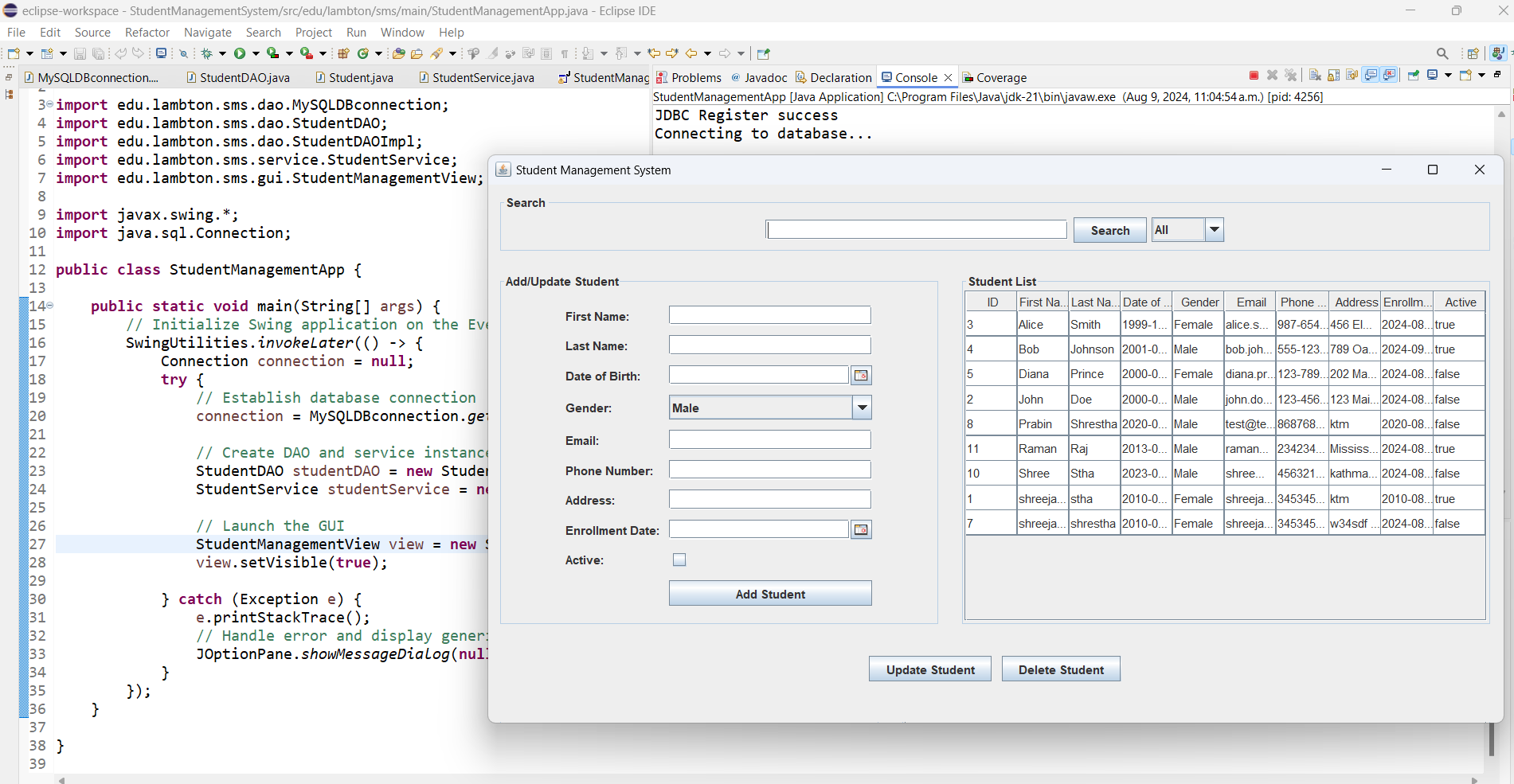
A screenshot of a computer

Description automatically generated

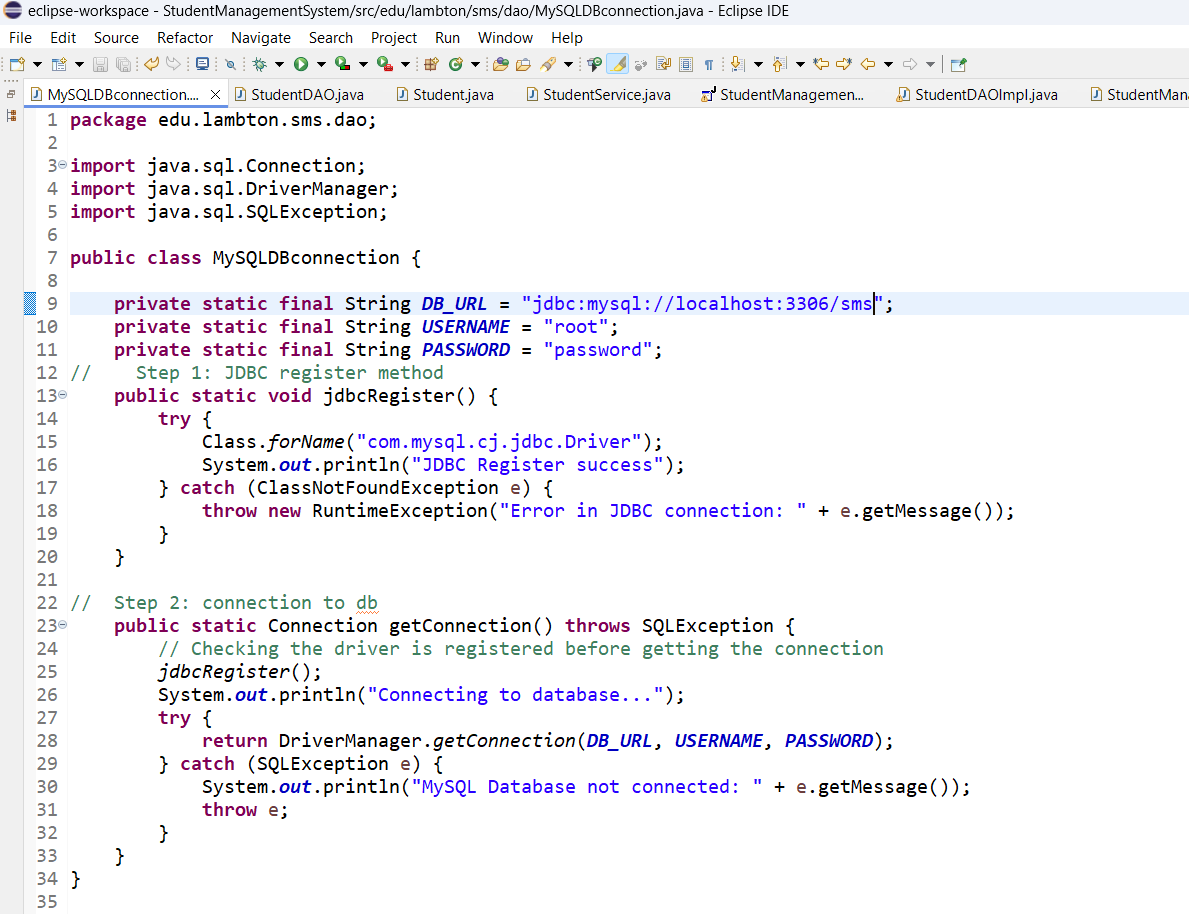
### # Existing Records on the student table



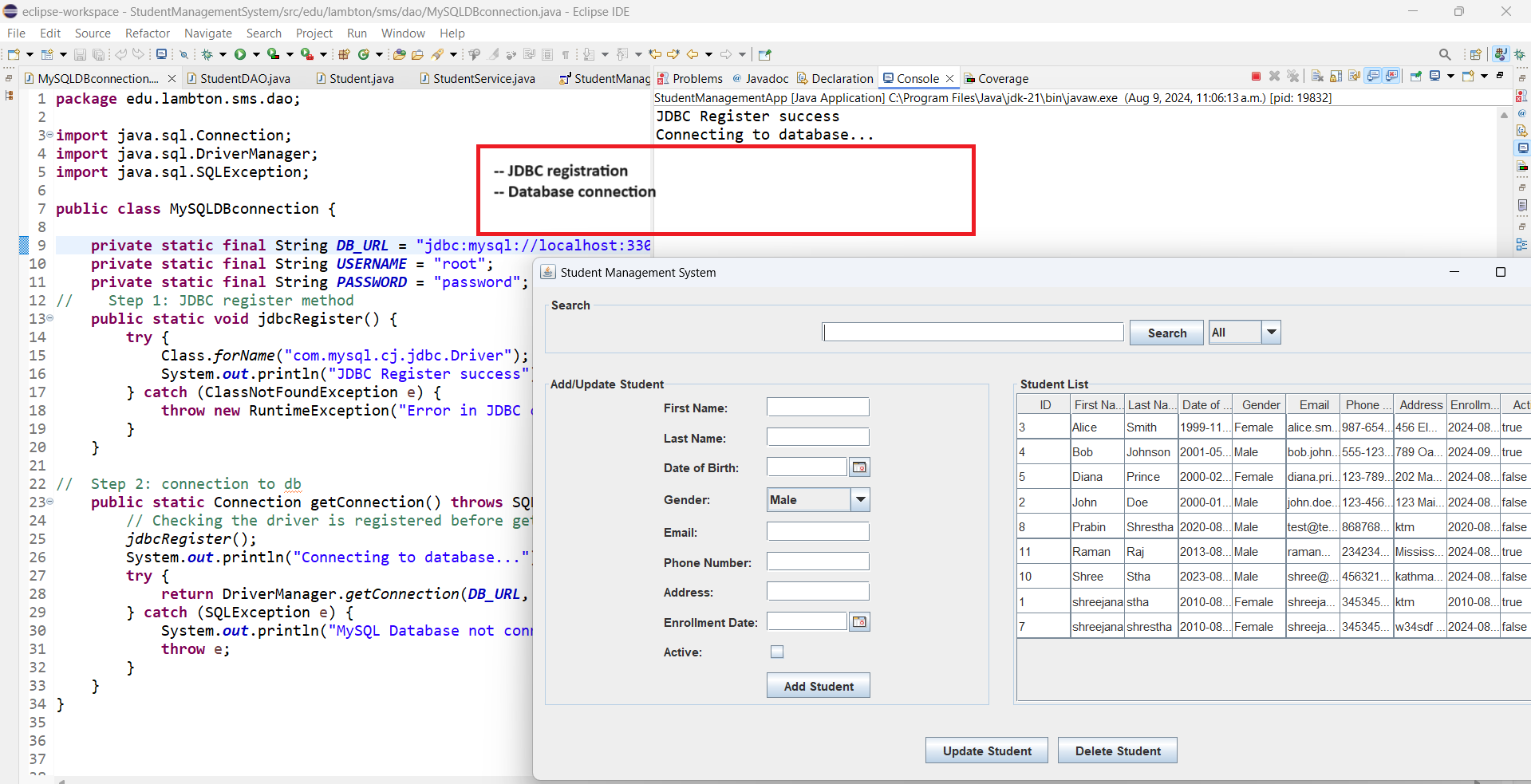
### # Making connection to the database



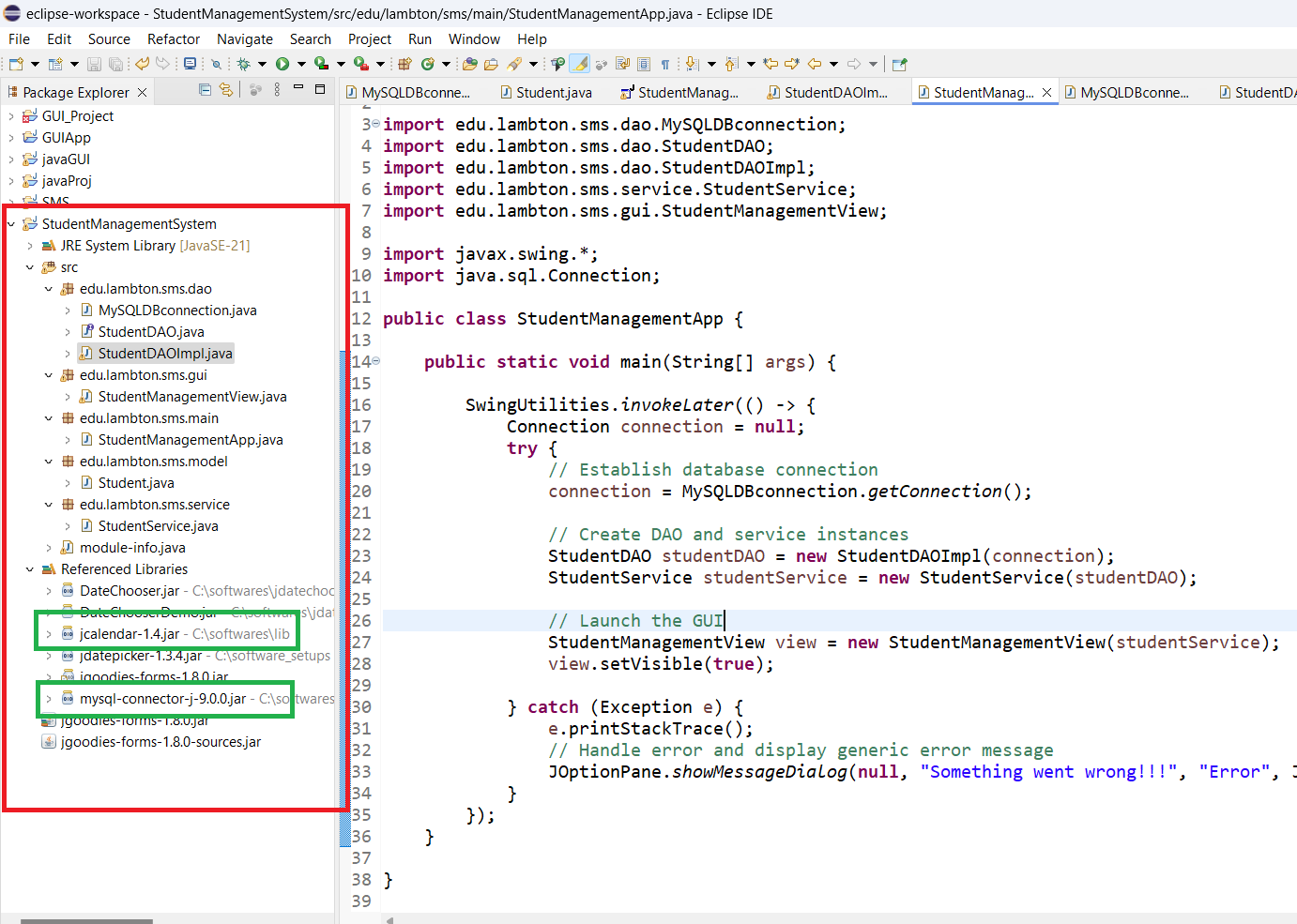
### # Database connectivity code snippet



* **MySQLDBconnection** class handles the database connection setup which is initialized from the **StudentManagementApp** class. Here, we have registered the jdbc and make database connection using DriverManager.getConnection(DB\_URL, USERNAME, PASSWORD).



## # Folder structure in the project



Here, we have used edu.lambton.sms as the root package for the project SMS. All the other packages gui, dao, service, main, model are organized under this. We have created a separate package and respective class for each functionality.

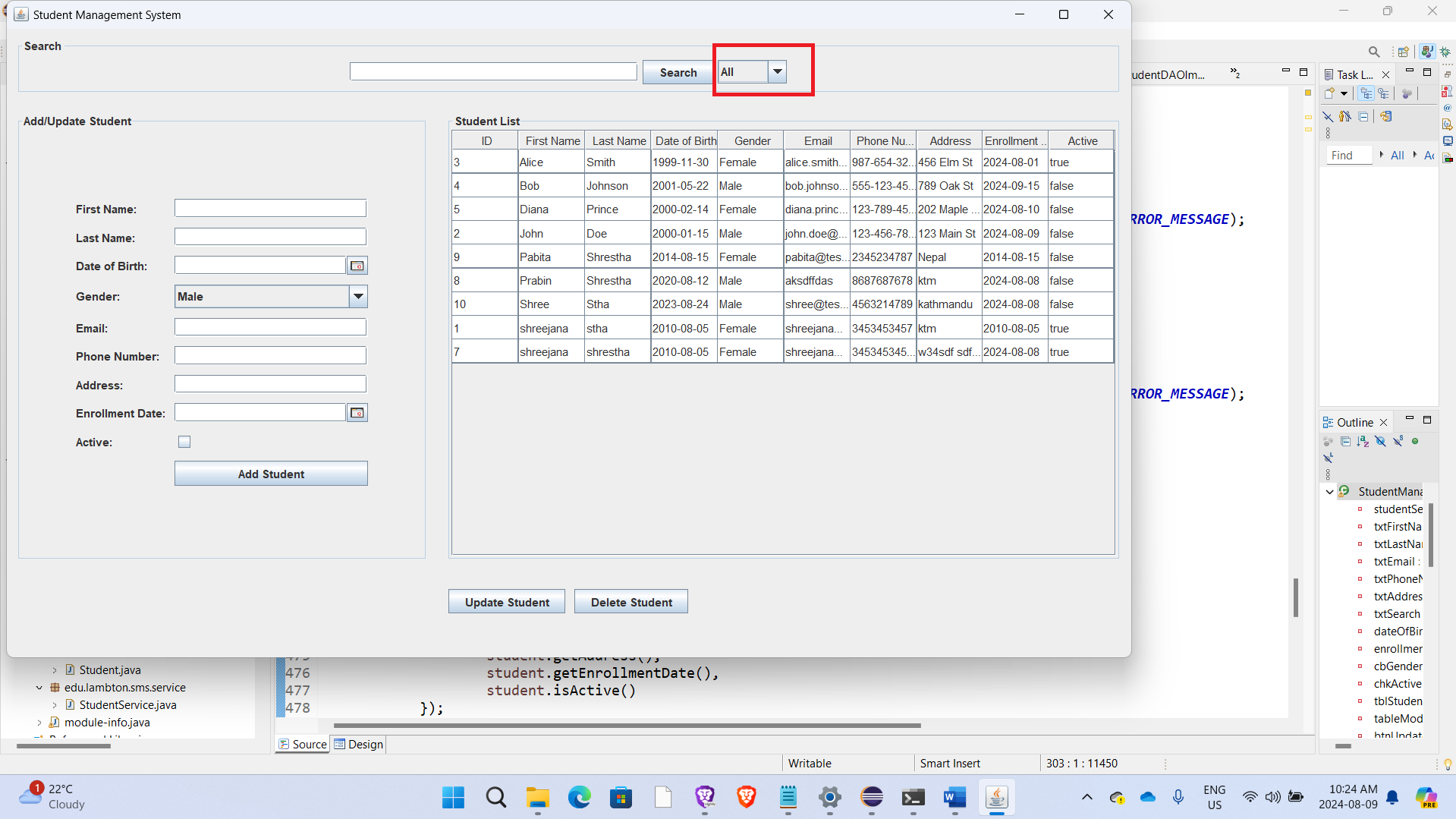
* Gui contains the components viewed by the admin/user. It manages the user interface, including forms and table views in our application.
* Dao contains data access object interface and implementation. It defines the data access methods for the student entity and contains all the SQL queries and database interaction logic.
* Model contains domain classes that represent the data model i.e Student in our application
* Service contains the service layer where the main business logic resides such as adding, updating, retrieving and deleting. It interacts with DAO layer to perform these operations.
* Main contains the main entry point of our application

## # First GUI of Student Management System

A screenshot of a computer

Description automatically generated

## # GUI with all from the student table listed below

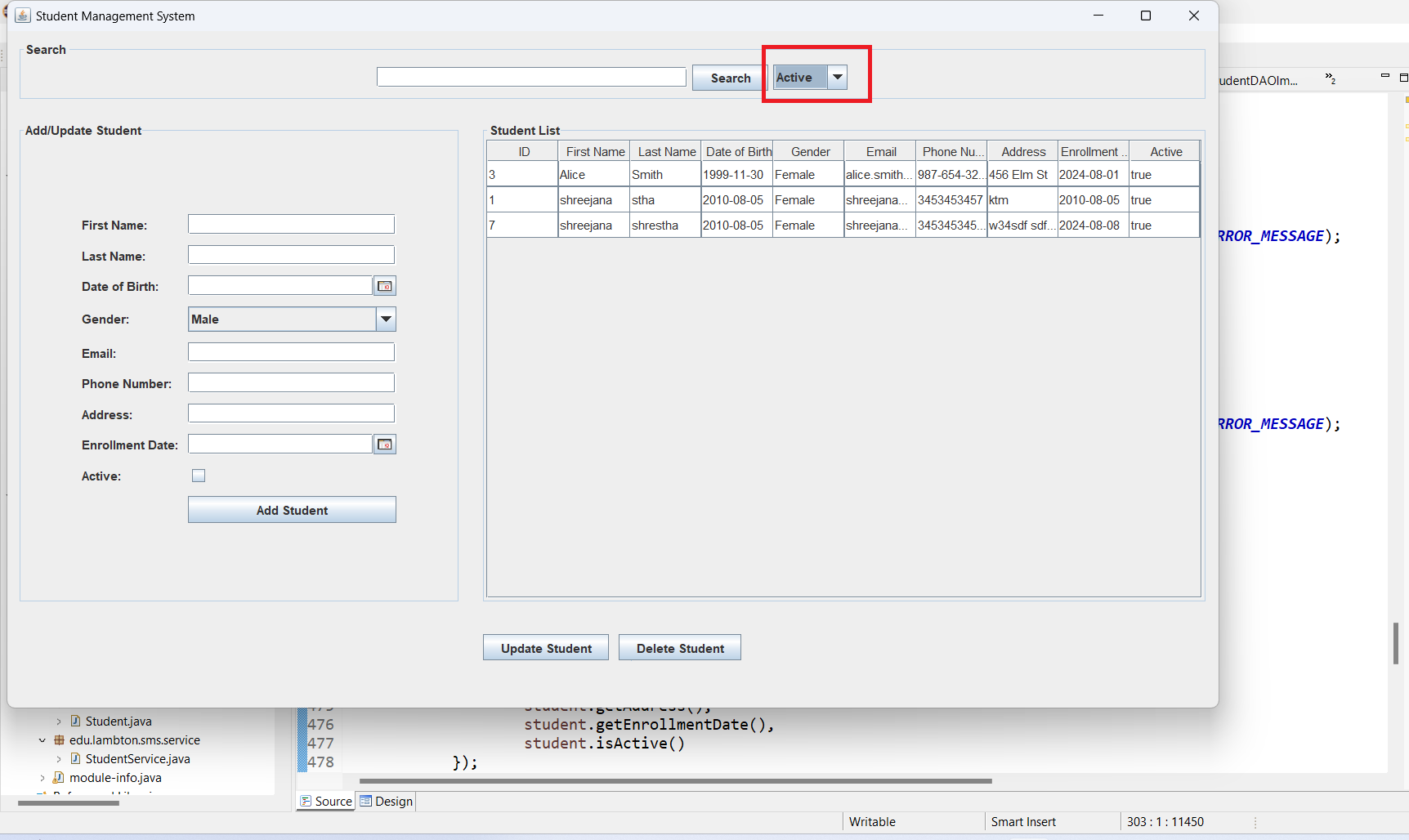


A screenshot of a computer program

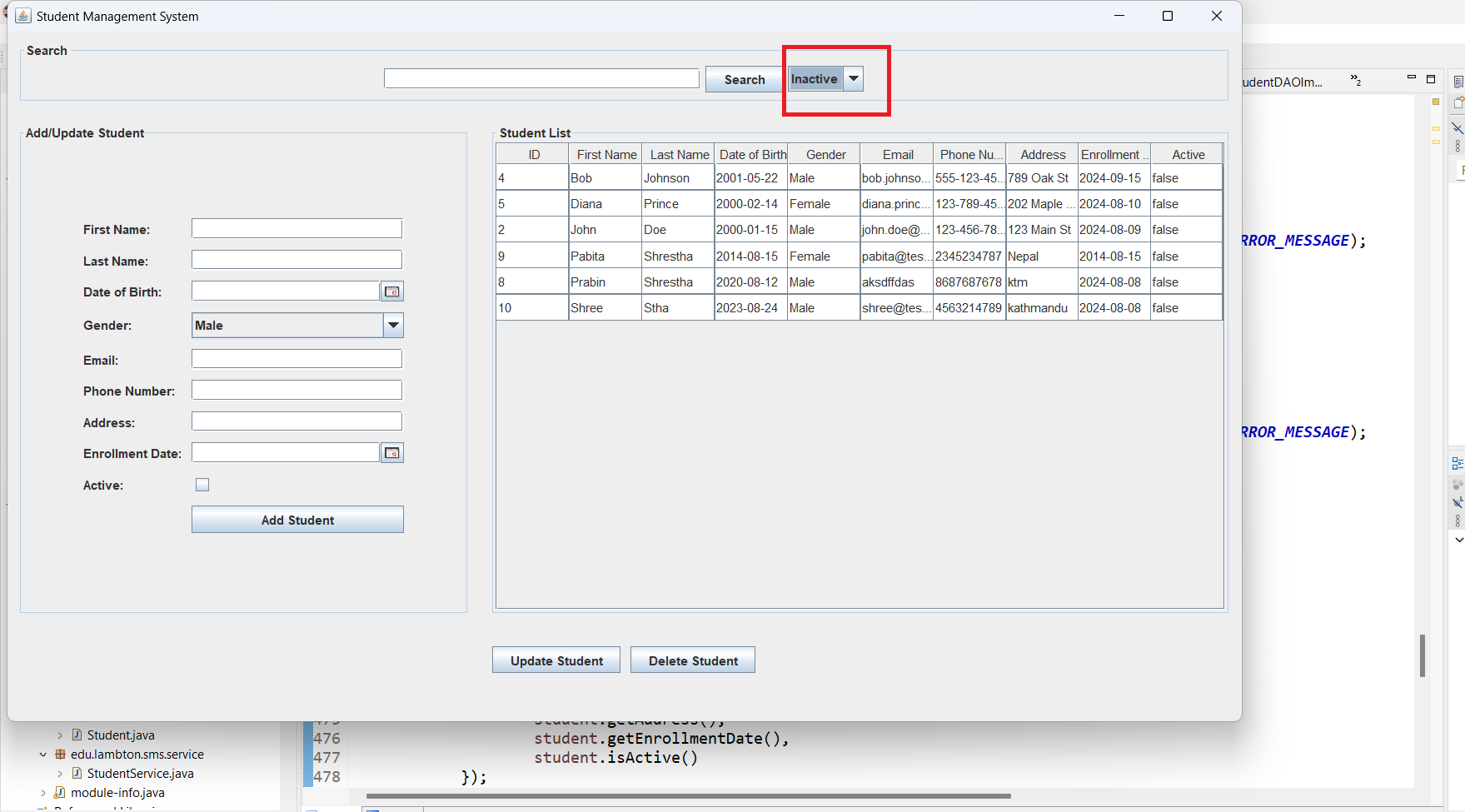
Description automatically generated

* This is the logic implemented to display all the records from the student table. The table is listed with data if exists when the application is first run.

## # Table listing only the active students record



## # Table listing only the inactive students record



## # Search filter

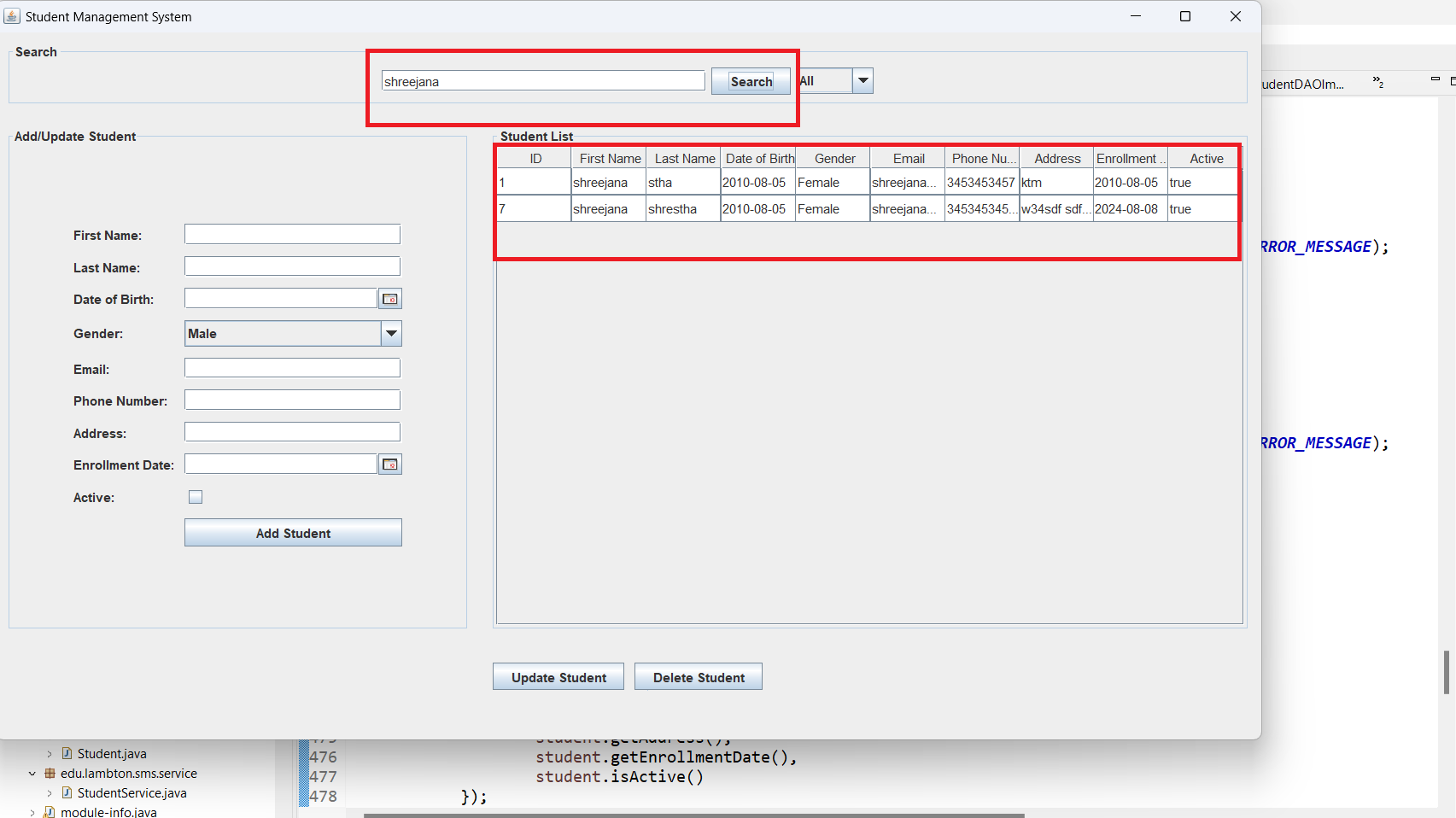
A computer screen shot of a code

Description automatically generated

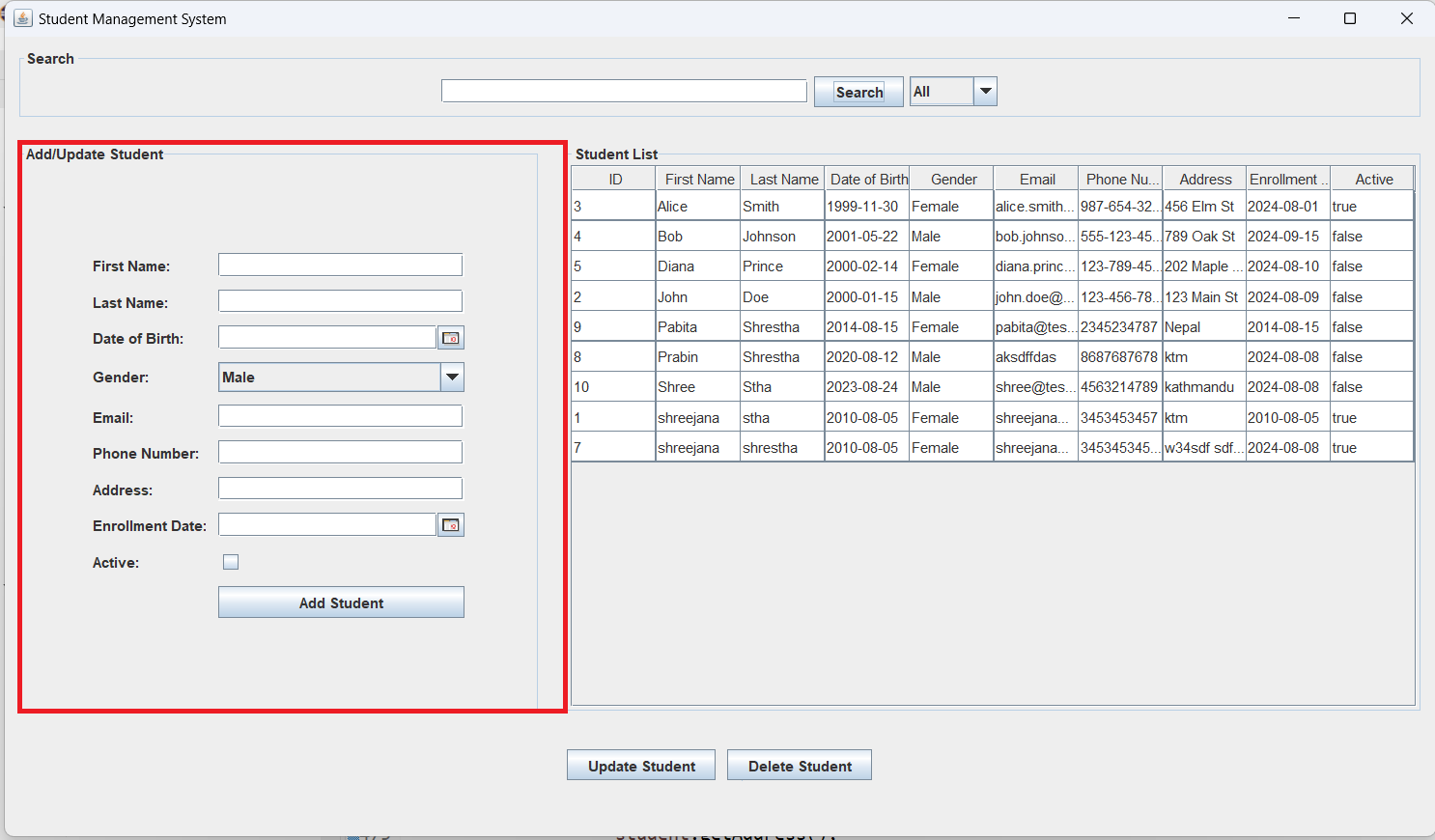
A computer code with text

Description automatically generated with medium confidence

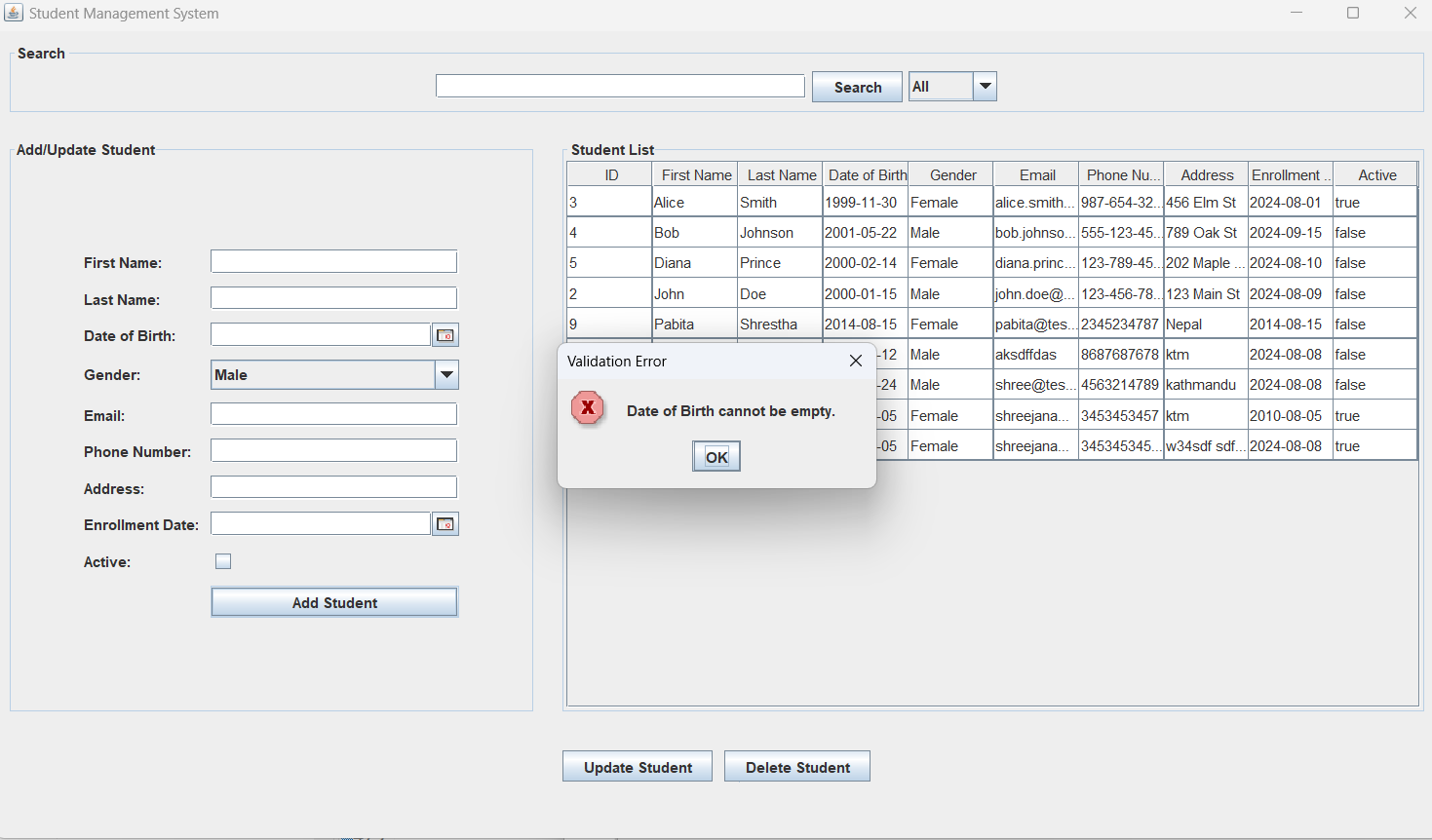
* The search filter button enables the functionality to filter and retrieve student records based on specific criteria such as students’ firstName, lastName and email. When a search query is entered, the system queries the database and returns a list of students that match the criteria and only the relevant student information is displayed.

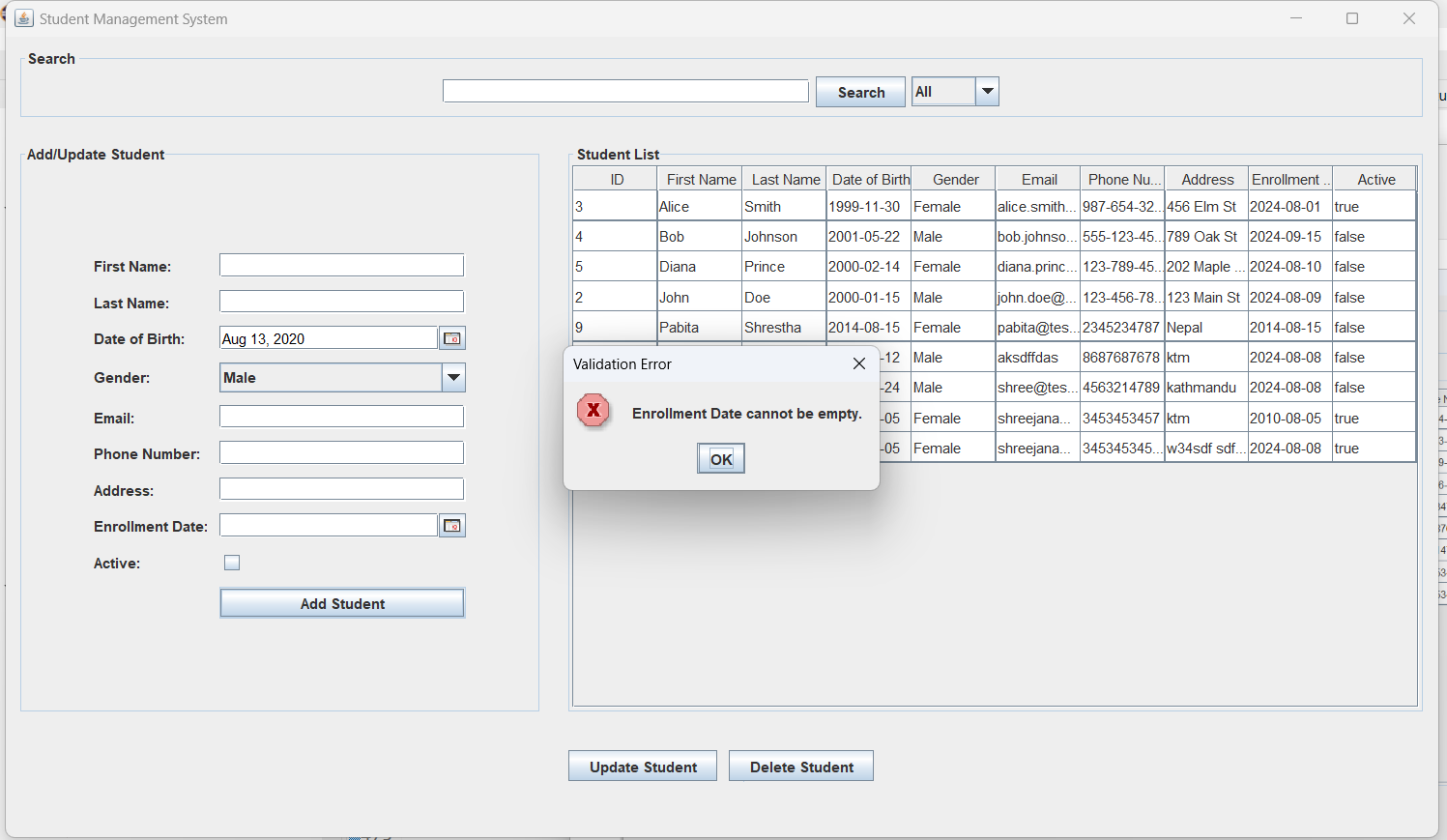


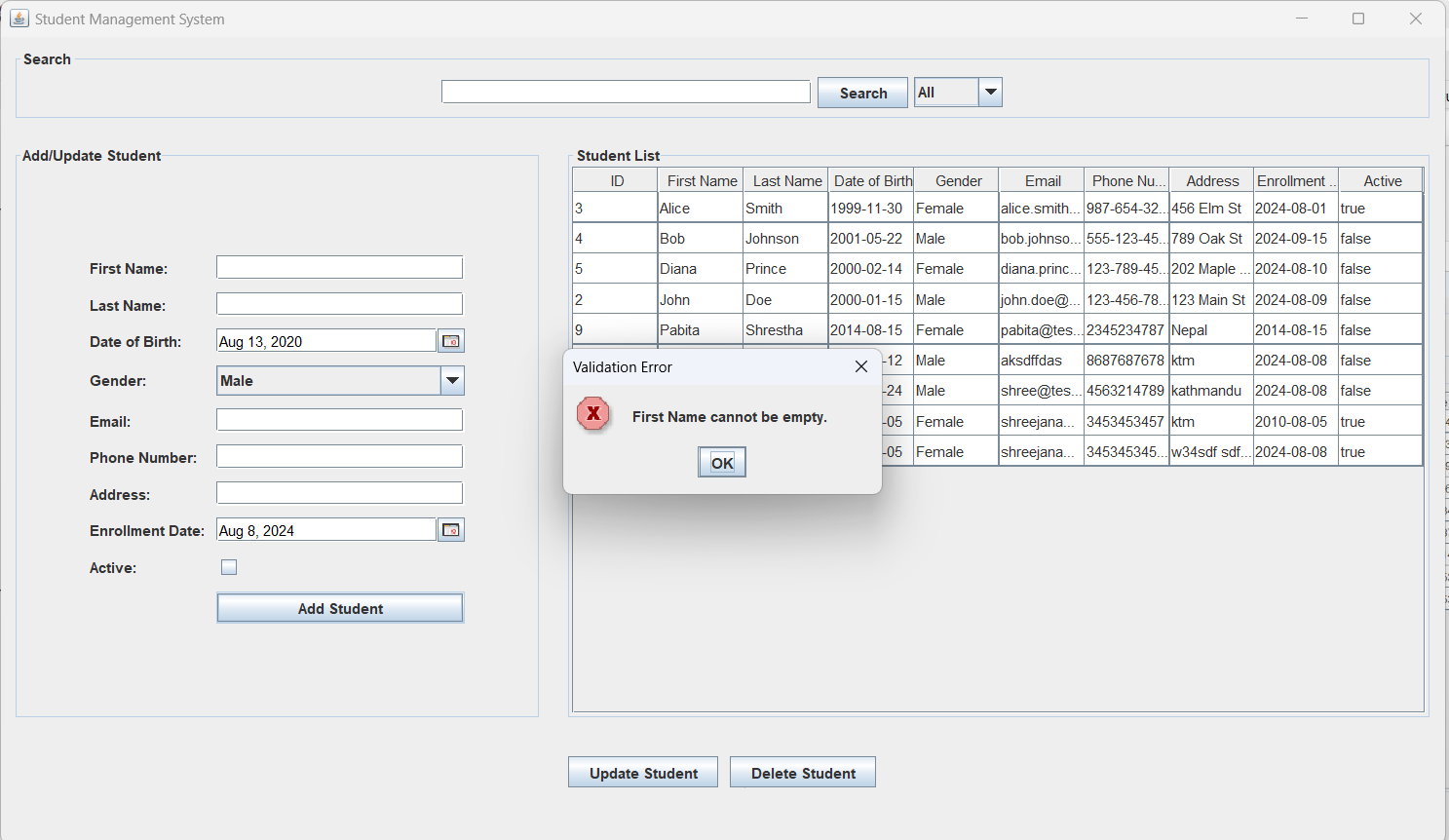
## # GUI to get the student data



## # Input validation for the form fields







## # Adding functionality

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

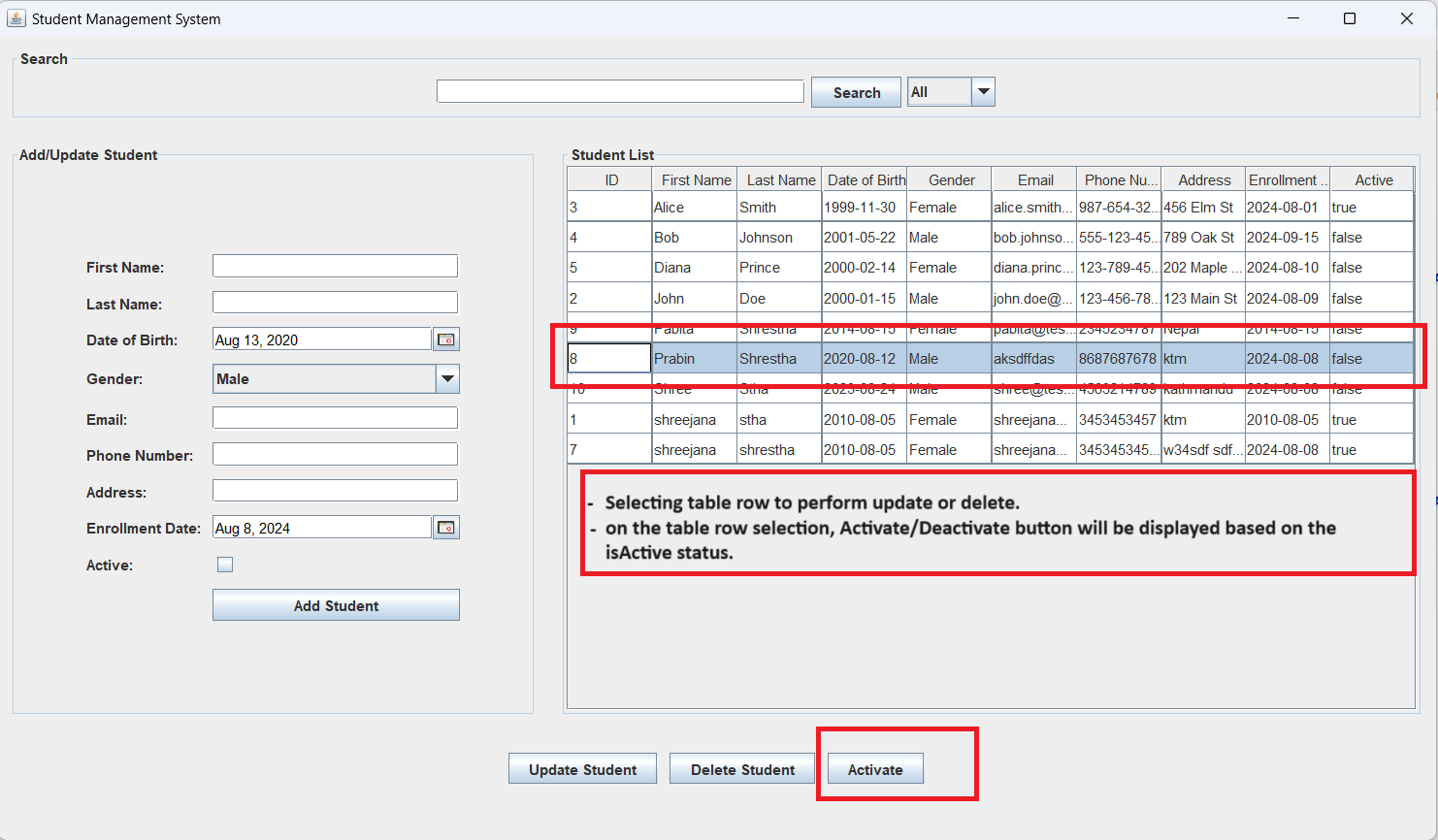
Description automatically generated

* The "Add Student" functionality in our project allows to input student details through a form in the GUI and save them to the database. When the admin user submits the form, the input data is validated, ensuring that all required fields are filled out correctly. Once validated, a Student object is created using the form data, and the StudentService interacts with the StudentDAO to insert the new student record into the database. If the insertion is successful, the form is reset, and the student table in the GUI is refreshed to display the newly added student.

## # Table after successful addition of record



## # Selecting the record to perform update/delete or activate/deactivate operation



## # Error message if no record is not selected

A screenshot of a computer

Description automatically generated

## # Update Operation

A computer screen shot of a code

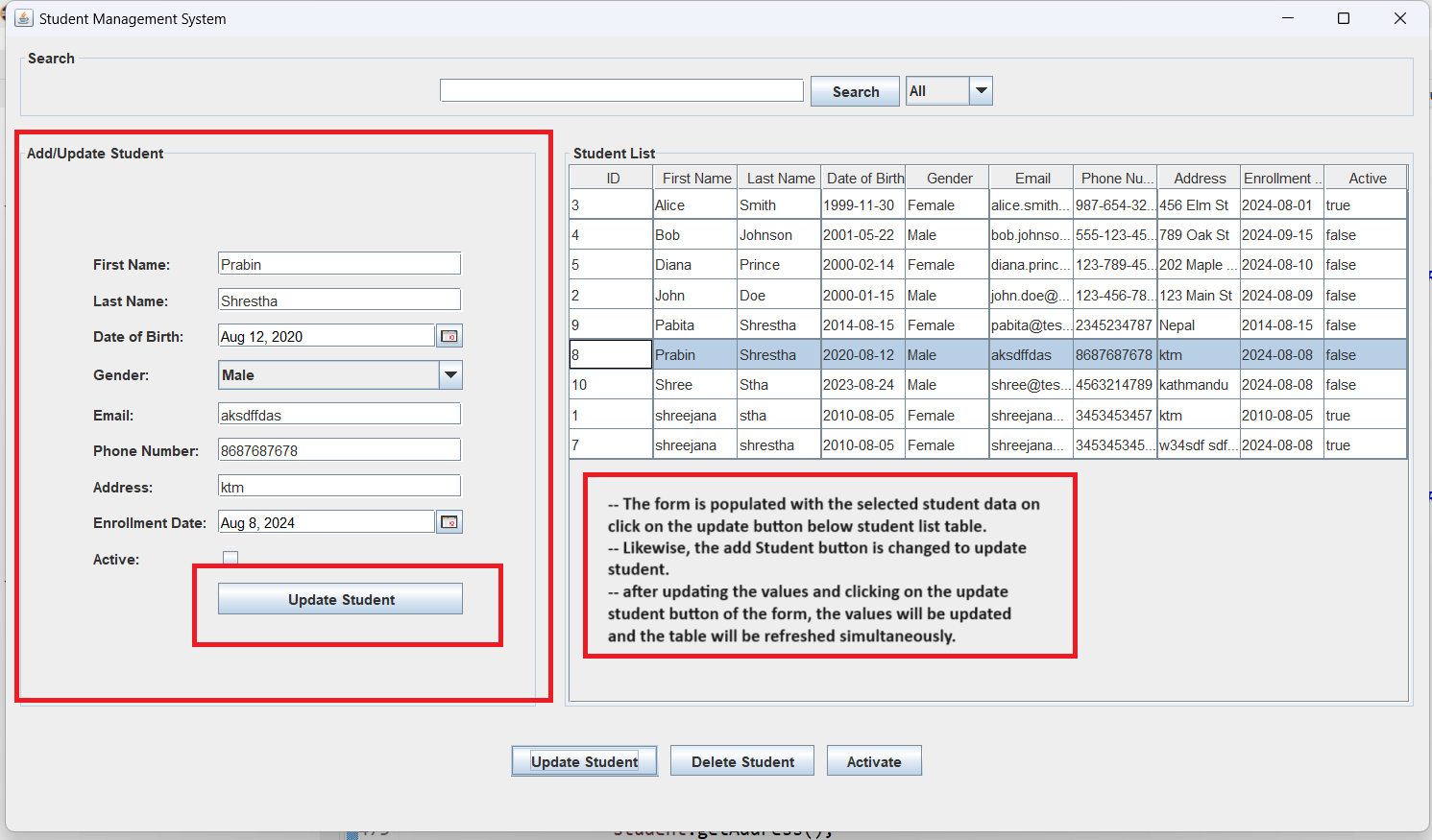
Description automatically generated

A screenshot of a computer program

Description automatically generated

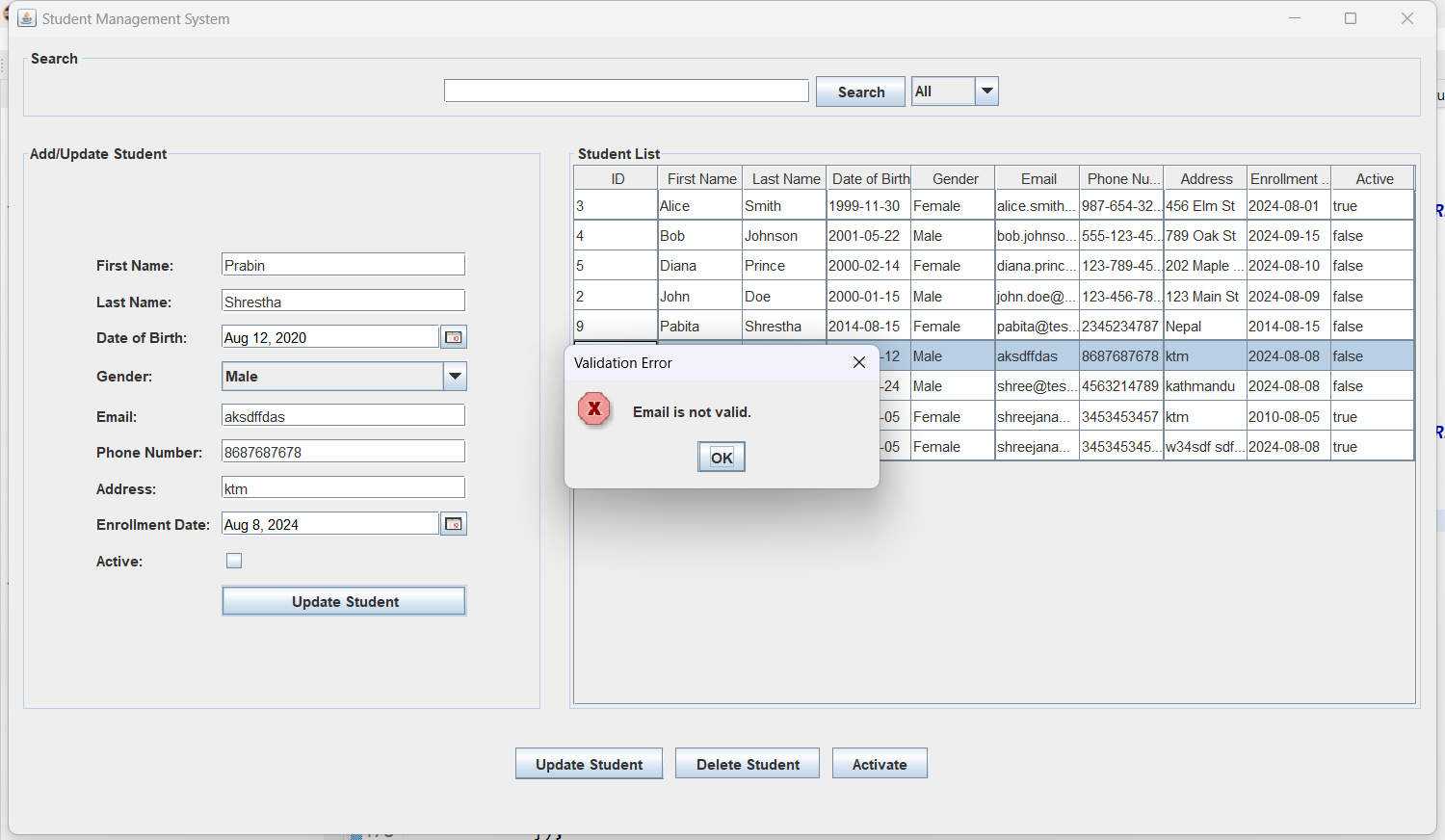
A screenshot of a computer program

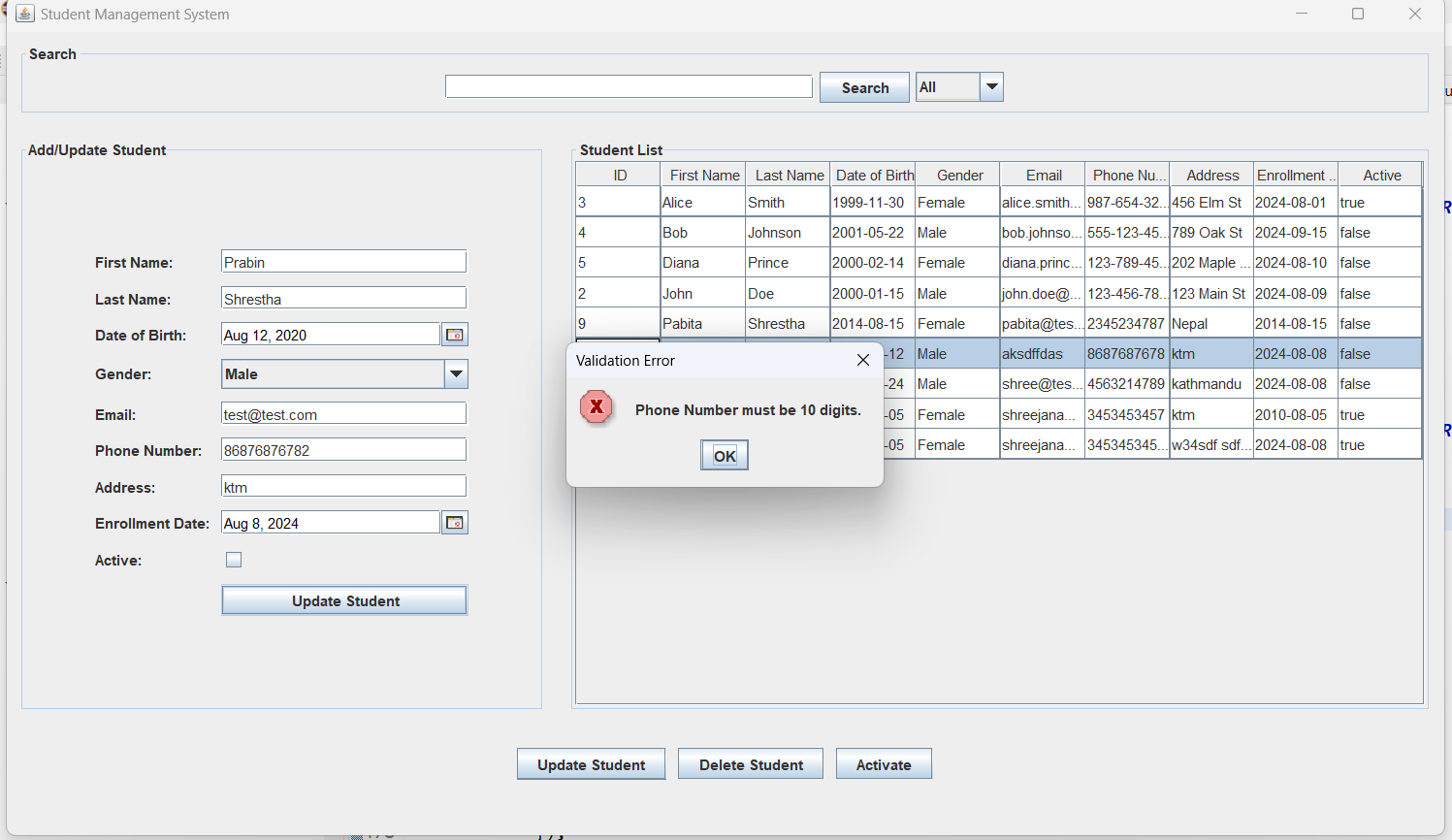
Description automatically generated



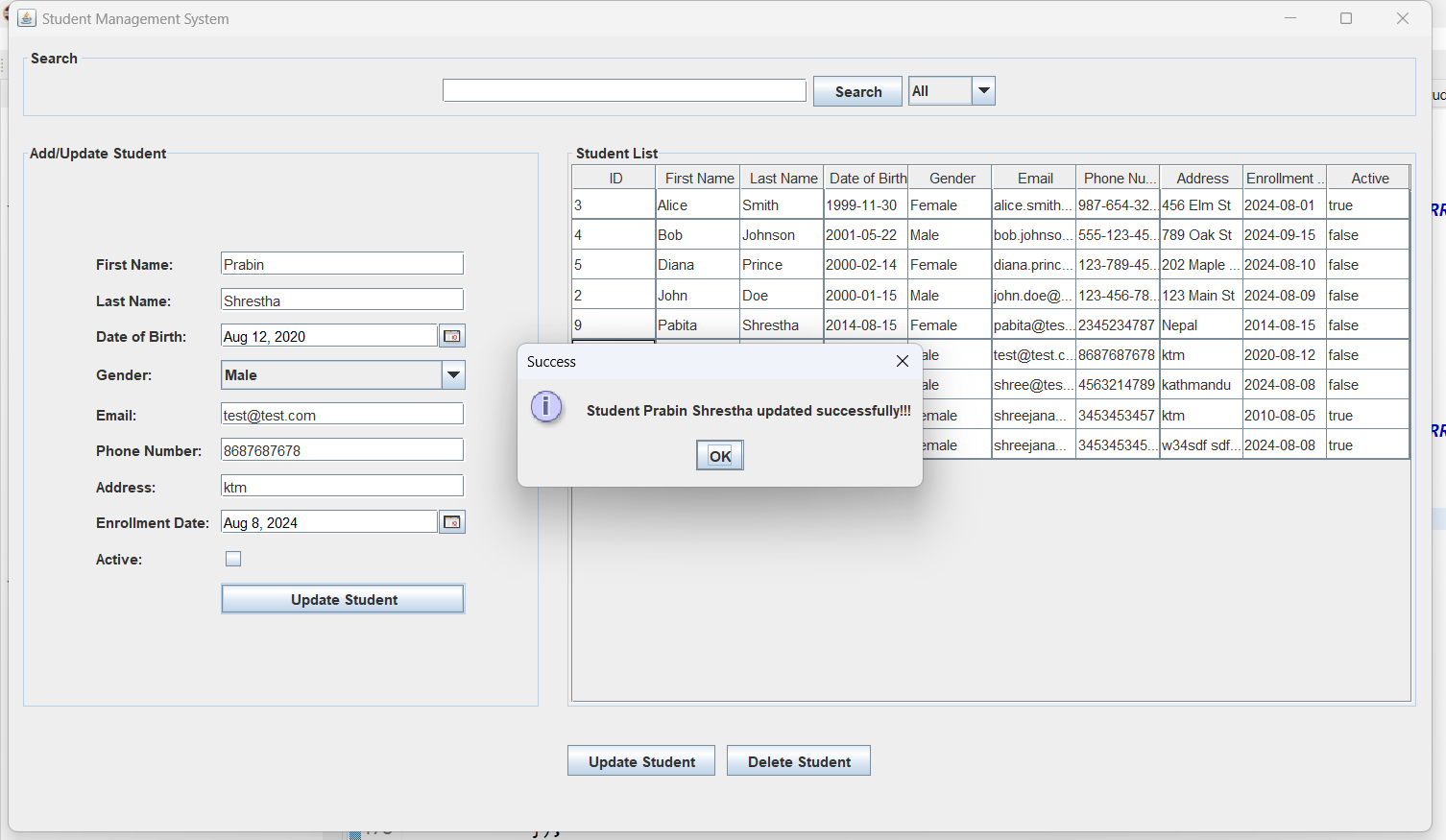
* With he "Update Student" functionality, admin user can modify an existing student's details. When a row in the student table is selected, the student's current data is populated into the form fields. The user can then edit these fields and submit the form. Upon submission, the data is validated to ensure correctness. The StudentService then interacts with the StudentDAO to update the student's record in the database using the modified data. After a successful update, the form is reset, the table is refreshed to show the updated information, and the button text reverts to "Add Student."

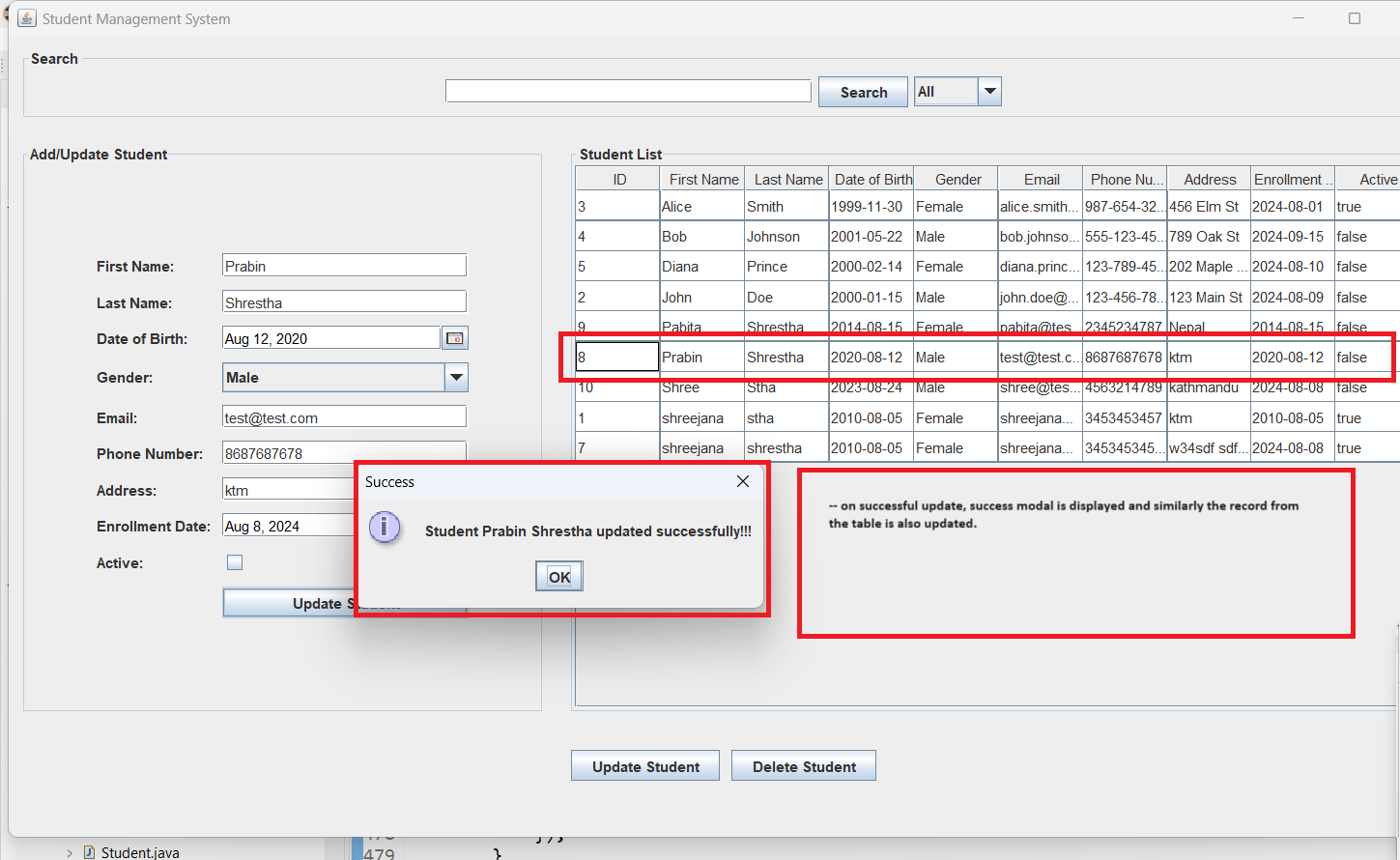
## # Input validation





## # Successful update of data





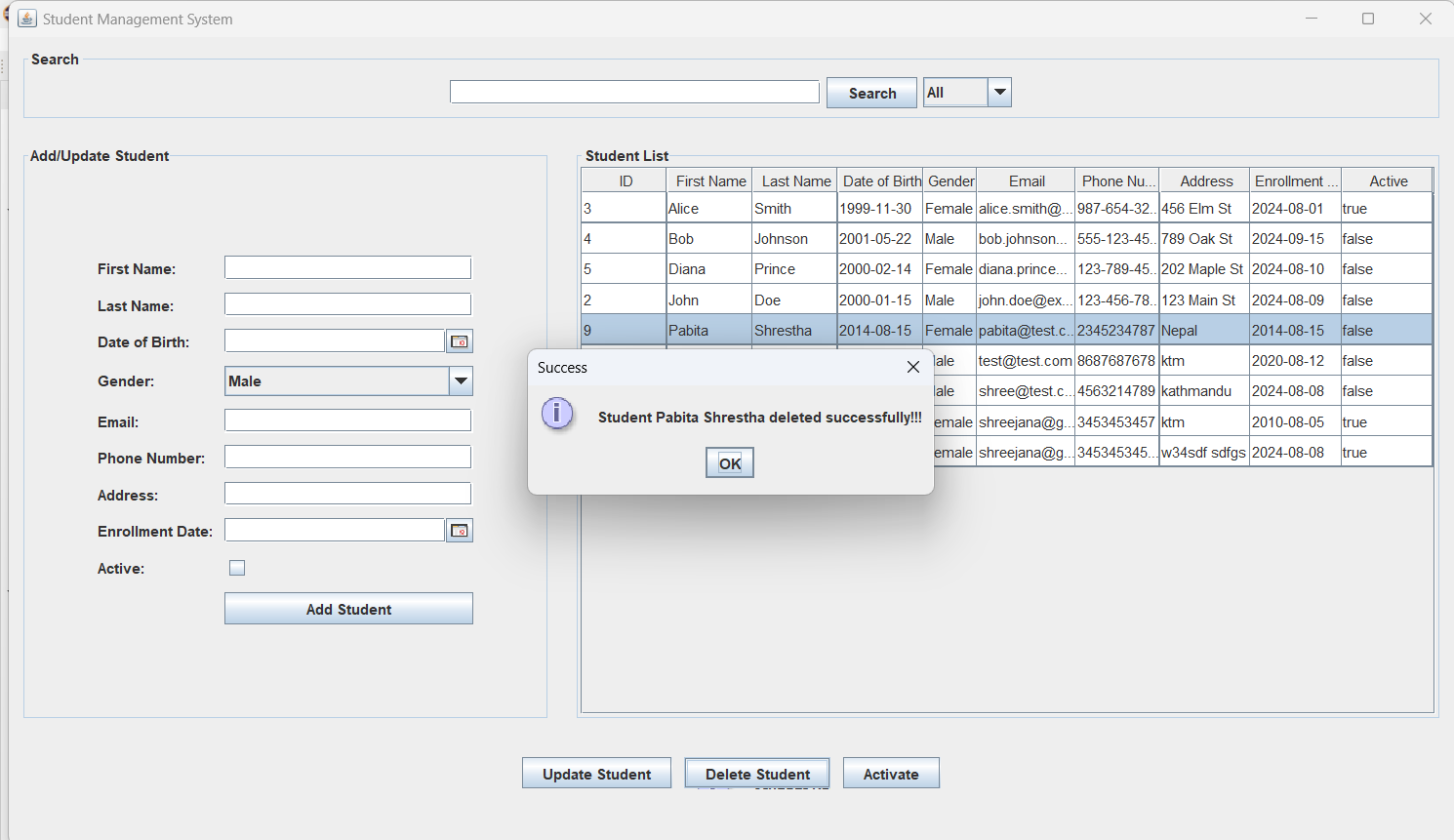
## # Delete operation

A computer code with many text

Description automatically generated with medium confidence

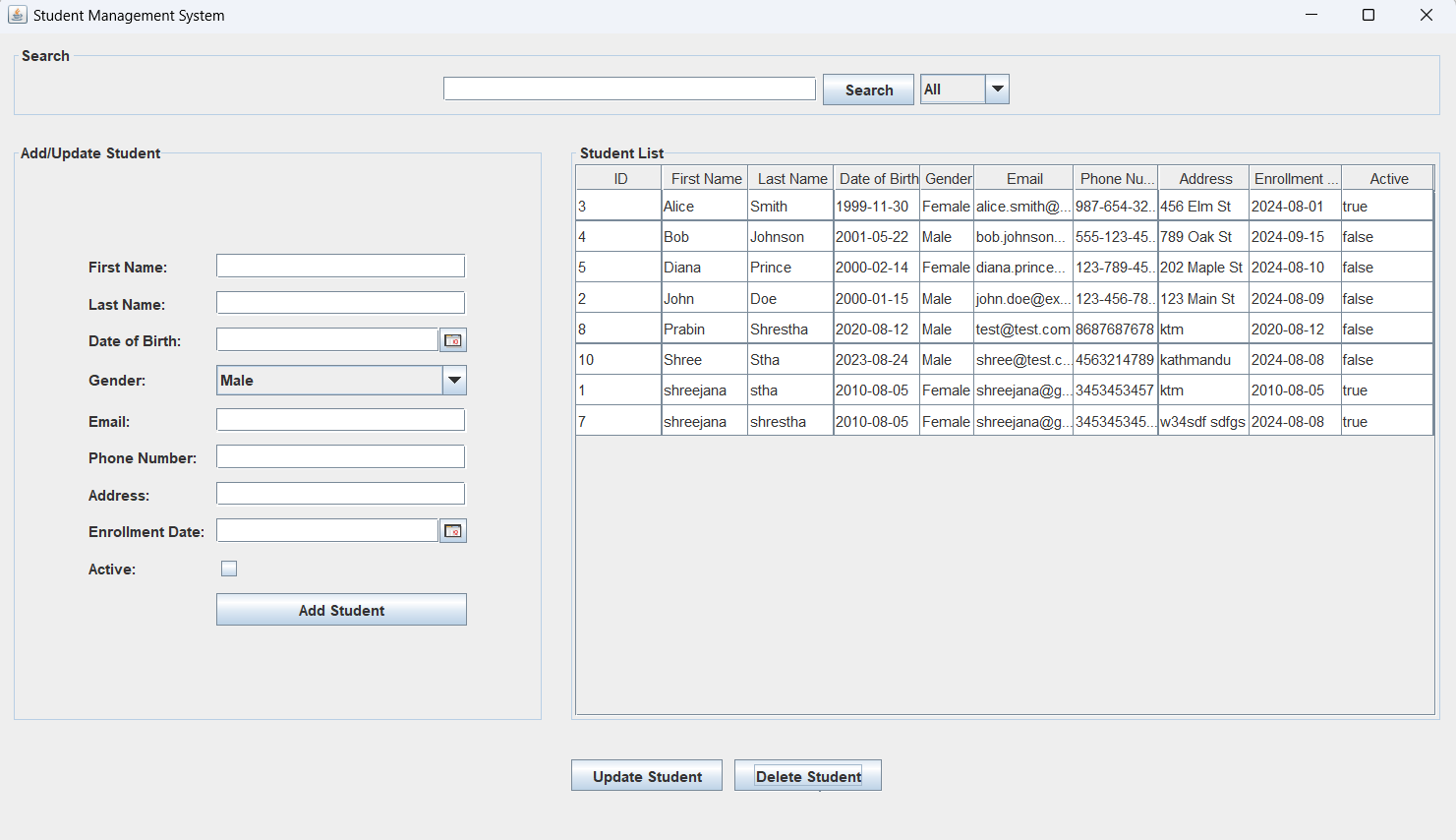
A computer code with text

Description automatically generated



* The ‘Delete Student’ functionality remove a student's record from the database permanently. When a user selects a row in the student table and chooses the delete option, the system prompts the user for confirmation to prevent accidental deletions. If confirmed, the StudentService interacts with the StudentDAO to execute a delete operation on the selected student's record in the database. Once the record is deleted, the table is refreshed to reflect the removal, and the form is cleared to reset the interface for the next operation.

## # Table after successful deletion



## # Activate / Deactivate functionality

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

* We have activate/deactivate functionality and the corresponding logic to show the data in the table based on the status selected from the drop down filter.
* It allows admin to activate or deactivate the students which can be achieved from the toggle of the button on the bottom section. If the student is currently active, clicking the "Deactivate" button will set the isActive field in the database to false. Conversely, if the student is inactive, clicking the "Activate" button will set the isActive field to true. The system uses the StudentService to call the StudentDAO, which executes the update query to toggle the isActive status in the database.

