Student Result Management System (SRMS)

Khyati Kachhawah , Bashu Bisht , Atharva Nimbalwar

Information Systems, Northeastern University

Software Engineering Systems, Northeastern University

{Kachhawah.k, bisht.b, nimbalwar.a}@northeastern.edu

***Abstract*—**

**The Student Result Management System (SRMS) is an automated web application designed to manage and track the academic performance of students. The system aims to address various challenges encountered by educational institutions, including time-consuming manual processes, inefficient record-keeping, inaccurate and inconsistent grading, difficulty in generating reports, and a lack of transparency. The SRMS allows teachers or administrators to capture and store data on student details, course enrollment, grades, and other relevant information. The system can generate reports and performance analysis that can be easily accessed and shared with administrators.**

**By implementing this system, educational institutions can reduce manual data entry, minimize errors, streamline the process, improve efficiency, and ensure transparency and accountability. The project will provide different users with different levels of authentication and authorization, such as teachers and administrators, with the ability to add, edit, and delete student information, including personal information, course information, and exam results. The system will integrate with a CSV file to generate reports and charts to analyze a student's overall performance, providing meaningful insights from visualizations such as bar, line, pie, and scatter charts. Overall, the Student Result Management System (SRMS) will offer educational institutions a centralized platform to operate student data and improve their management processes.**

**This is done using Eclipse with Java, Java FX, and FXML. Maven dependencies to inject MySQL jar files, MVC Framework with Java. All database operations are created in a separate file, jdbcDao, which takes care of all CRUD operations. Operations and design of this project can be classified into four major parts, which are- registration/ login, management of student results , Activity Log and Admin operations.**

***Keywords— JavaFX, Object Oriented Programming, MVC Framework, JAVA, SQL(JDBC)***

# **I. Problem Description**

The Student Result Management System (SRMS) is a web application designed to automate and streamline academic performance tracking for educational institutions. The system captures and stores student data, including grades and course enrollment, and generates performance reports for analysis. With SRMS, institutions can reduce errors, minimize manual data entry, and improve transparency and accountability. The system is built to support multiple user roles, such as teachers and administrators, with different functionalities and access privileges based on their roles.

# **II. Analysis (Related Work)**

Compared to the traditional method of manual tracking and record-keeping, implementing a web-based Student Result Management System (SRMS) has numerous benefits for educational institutions.

Firstly, it eliminates the need for manual data entry, reducing the risk of errors and inconsistencies in student records. This not only saves time and effort for administrators and teachers but also ensures the accuracy of the data.

Moreover, the SRMS enables real-time tracking of student progress, making it easier for teachers to identify areas where students need improvement and intervene accordingly. The system can generate automated reports that provide a comprehensive analysis of a student's performance, including grades, attendance, and overall progress. This makes it easier for teachers to provide personalized feedback to students and helps them make informed decisions about teaching methods and strategies. The SRMS also promotes transparency and accountability in the management of student data. By storing all the information in a centralized system, administrators can easily monitor student progress, identify any discrepancies or issues, and take corrective measures.

Additionally, the system can provide controlled access to different user roles, such as teachers and administrators, based on their functionalities and access privileges, ensuring data security and privacy.

Also, we have added functionality to generate CSV file from the student records and integrated with charts to display the progress of students in a class.

Overall, the SRMS streamlines the academic performance tracking process, improves efficiency, and provides meaningful insights into student progress. By leveraging the power of technology, educational institutions can improve their management processes, enhance the quality of education, and ultimately contribute to the overall growth and development of their students.

# **III. System Design**

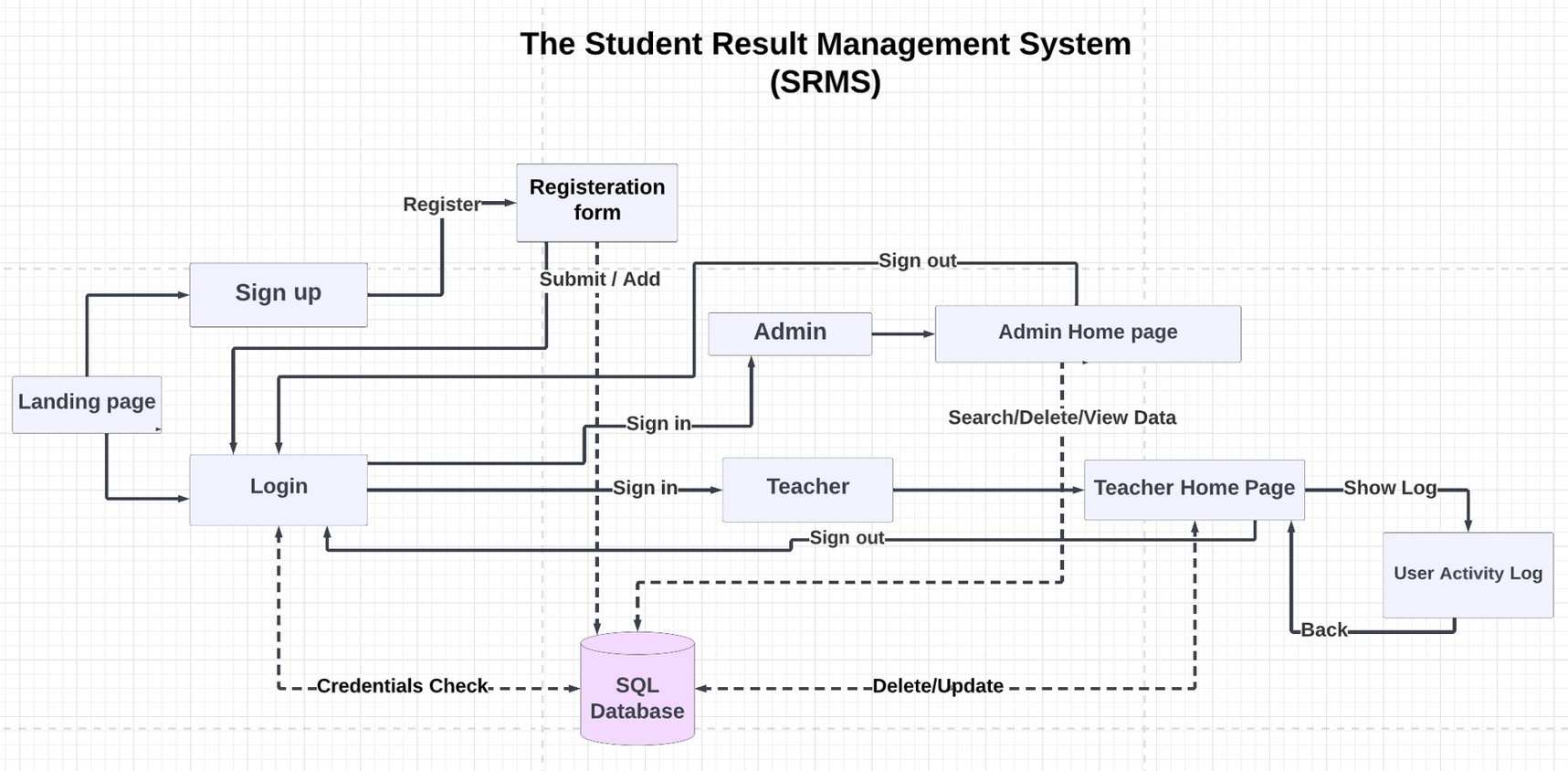


Figure 1. System Architecture

# **IV. Implementation**

*Brief Overview*

The software design pattern known as Model View Controller, or MVC, is used for creating this web application.

A Model view Controller pattern comprises of the following three parts:-

* Model- It represents the data and logic of the application.
* View- It is responsible for displaying the user interface.
* Controller- It acts as an intermediary between the Model and the View.

The Model View Controller architecture is widely adopted due to its ability to separate the presentation layer from the application logic, enabling the modularization of concerns and promoting maintainability. In the Model View Controller design pattern, the Controller acts as an intermediary between the View and the Model, receiving requests from the user interface and using the Model to prepare the necessary data before passing it to the View to generate the output.

*A. Registration/Login*

The login and signup buttons are accessible on the application's home page. New users(teachers) first should register, and hitting the submit button on the registration form prompts the java file to connect to the database and add teacher’s credentials to the User Database; however, logged-in teachers can simply login with their credentials.

In order to successfully log in, the teacher ID must end with "@northeastern.edu". If it does not, an alert is generated indicating that the login has failed. The user's credentials are then checked against the information stored in our SQL database. If there is a match, the teacher is considered to have successfully signed in and will be directed to the User Home page.

*B. User Home/ Post Login*

Once a user has completed the login process, the controller utilizes the Initializable method to present all the dynamic data retrieved from the database and display it on the application. The code overrides this method to ensure that the page is loaded with the correct presentation of labels, buttons, and text.

The teacher can see the dropdown of student names, student enrollment number, age and final grades for options on the display. The admin/teacher also has the authority to delete/update data into these lists.

The database is updated when a plan is added or updated, but when a user clicks the delete button, they are removed from the database system. Data conversion and display are done during page load using the list operator.

*C. Activity Log*

All the teacher's activity for that session is displayed on the screen when the show log button is clicked. Two static variables with the data types Stack and Dequeue were used to accomplish this. We call the push log function, which puts the string into the stack, each time an action is performed.

The pop() method is used to show the log in reverse order, with the most recent activity being displayed on top because the Stack data type is built on first-in-last-out.

We utilize the double-ended queue (deque) data type and push information into the queue before popping the element from the stack to print in order to save the order of the elements in the stack.

Finally, the deque is cleared and all information is pushed when the user presses the back button.

*D. Admin*

In the current application, the sysadmin role is assigned to the administrator, who is the only user without an @northeastern.edu ID. When the "Submit" button is clicked, the user is redirected to the administrator's homepage, where they can search and delete records from the database.

In order to check the input text with our stored data and return all the information, we utilized the LIKE query in our jdbcDao file.

If the admin presses the delete button, the corresponding data of the teacher is deleted from the database, and the admin has the authority to remove any teacher’s records along with the student’s details from the system, while the creation of a new admin account by students or teachers is prohibited since the login details for the admin account are hardcoded.

If there is a need to modify or remove the admin account, it can only be achieved by either updating the hardcoded credentials in the source code or manually altering the admin credentials stored in the database, and upon clicking the sign out button, the admin is redirected to the login page.

*E. Database Connection*

In the application, the jdbcDAO file contains all the database connectivity details and SQL queries. Whenever a Controller needs to interact with the database, it instantiates an object of the JdbcDAO class, which provides various functions such as addRecord, setRecord, getRecord, searchRecord, searchAdminRecord, and DeleteRecord.

To establish a connection with the MySQL database hosted on localhost using port 3306, we utilize the Driver Manager class.

Also, generating CSV file from the database. Using this approach, we can create prepared statements that accept dynamic parameters and avoid the need to write a new query every time we call the corresponding function.

To ensure that the user remains logged in during their session, we implement a User Session class that is invoked by calling the getInstance() function immediately after the user logs in. Upon invocation, the getInstance() function creates a new instance of the UserSession class, which retains the user's username until the setInstance() function is called, resetting the instance value to null, effectively logging the user out. This process repeats with each login session.

The searchAdminRecord function in the JdbcDAO class does not require a database call since the admin's name and password are hardcoded in the codebase and do not rely on database storage. However, for general user login, the application performs a database check, with the register function responsible for adding new user rows to the user\_registration data table. During login, the system verifies user credentials by checking if the SQL select query returns at least one row of data corresponding to the given username and password.

The ood\_project database comprises two independent tables, user\_registration and student\_result, with no foreign keys linking them. The user\_registration table solely contains user credentials (username, email, and password), while all result-related information is stored in the student\_result table. Both tables share the email ID field, enabling the Post login page to display the user's information immediately after logging in. Additionally, the ID fields in these tables are auto-incremented, alleviating the need to specify their values each time a new row is added. Furthermore, the MySQL varchar values are converted to strings in our Java code, thereby encapsulating the project's database information.

*F. Visualization*

The visualization part of the project allows the administrator to view student results by clicking on the "View Data" button. Clicking on this button will fetch the data from the database and generate a CSV file.

After generating the CSV file, a new window for graph visualization will appear. In this window, the administrator can select two parameters for the 'x' and 'y' axes, based on which the graphs will represent the data selected. The available options for the parameters will depend on the data stored in the database.

The project includes several types of graphs for visualization purposes, including:

1. Line chart: A line chart is a graph that displays information as a series of data points connected by straight lines. It is useful for showing trends over time.
2. Pie chart: A pie chart is a circular graph that is divided into slices to represent the proportion of data in each category. It is useful for showing the distribution of data.
3. Bar graph: A bar graph is a graph that represents data with rectangular bars with lengths proportional to the values they represent. It is useful for comparing data across categories.
4. Scatter chart: A scatter chart is a graph that displays the relationship between two variables, with one variable on each axis. It is useful for showing correlations between variables.

The visualization part of the project provides the administrator with a variety of options to explore and analyze the data in the database, making it easier to identify patterns and trends in student performance.

# **V. Evaluation**



1. Home page

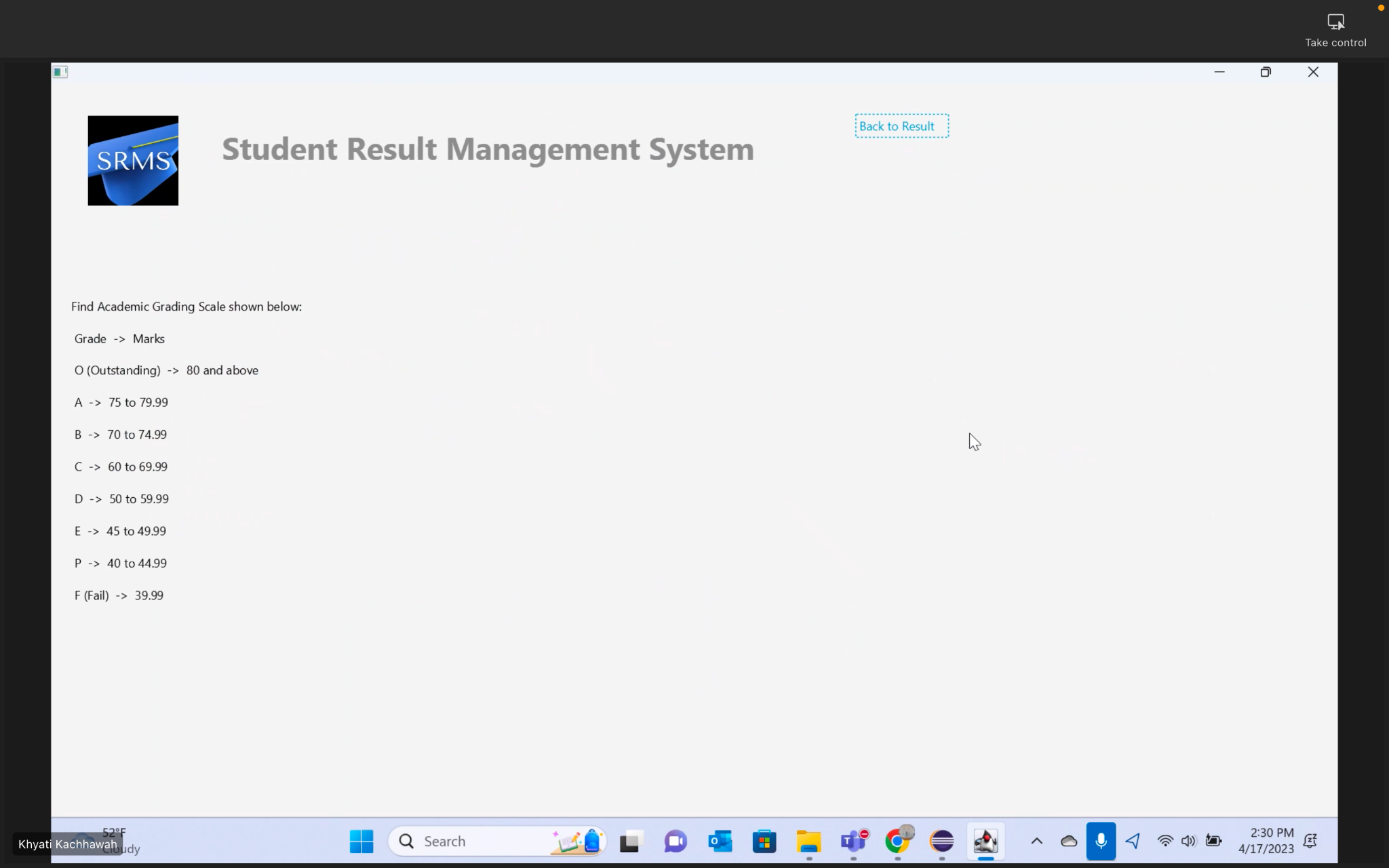
This is the Home page where the user has options to sign-in or register.

Graphical user interface

Description automatically generated

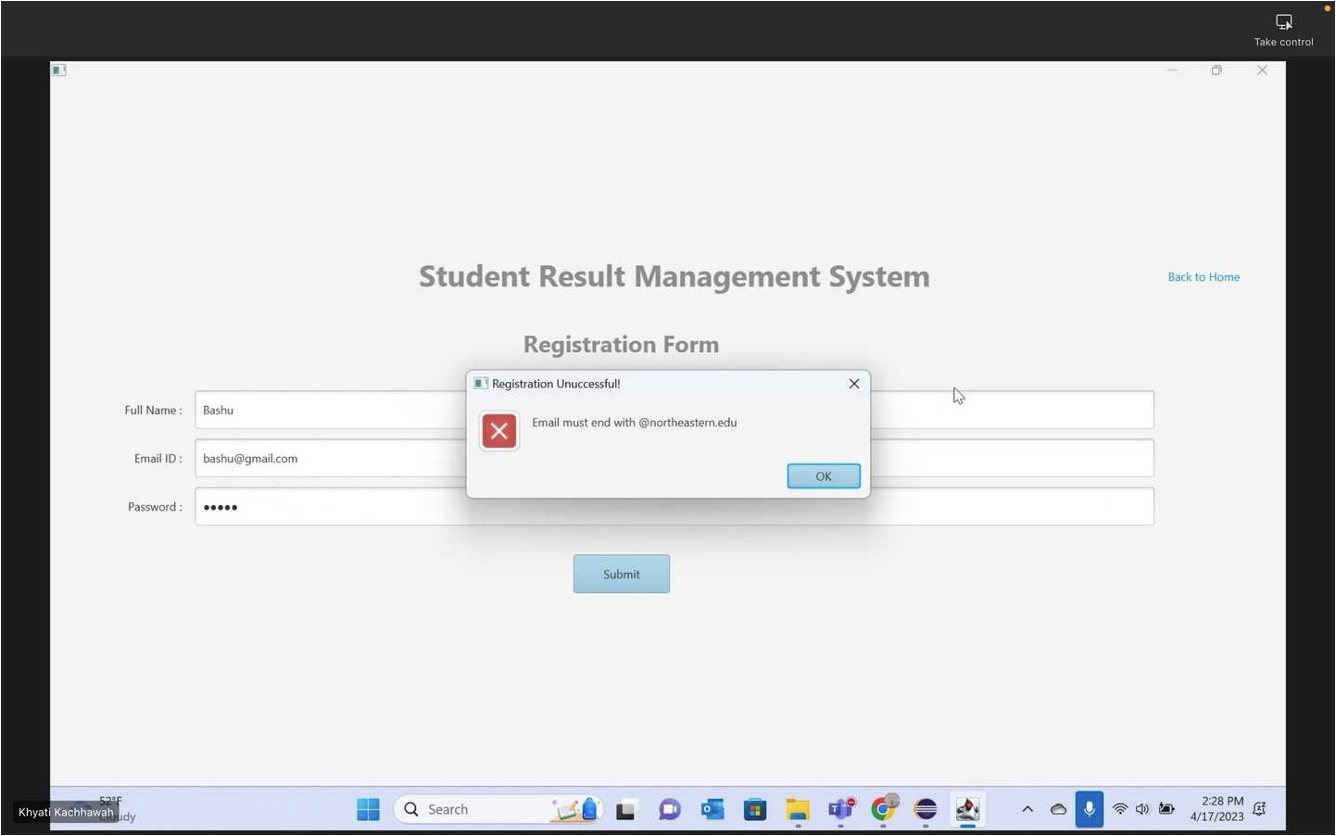
1. Our Team

This is the page showing the developers of this application.



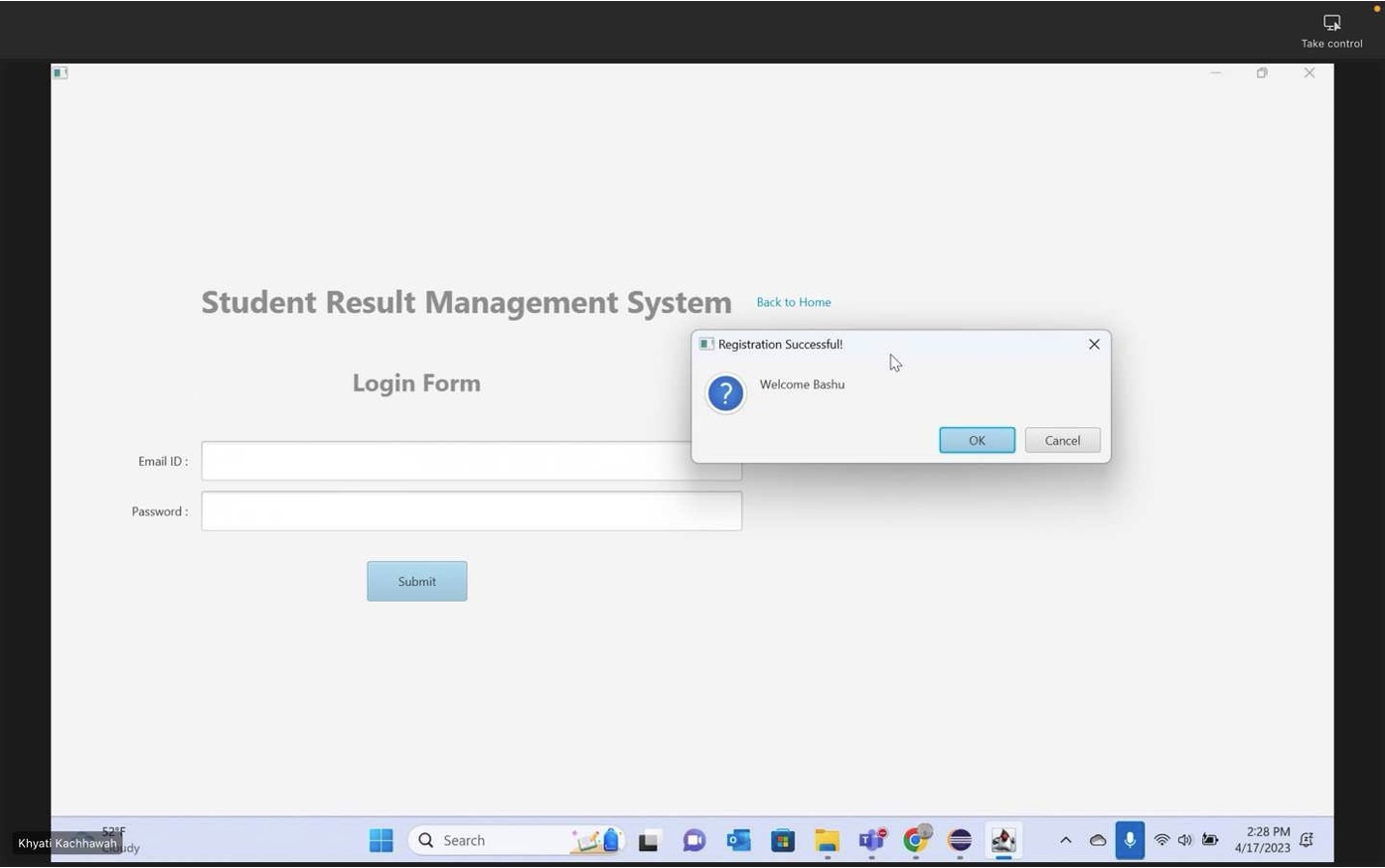
1. Grading Scale

This page shows the representation of grades to their respective percentage values.



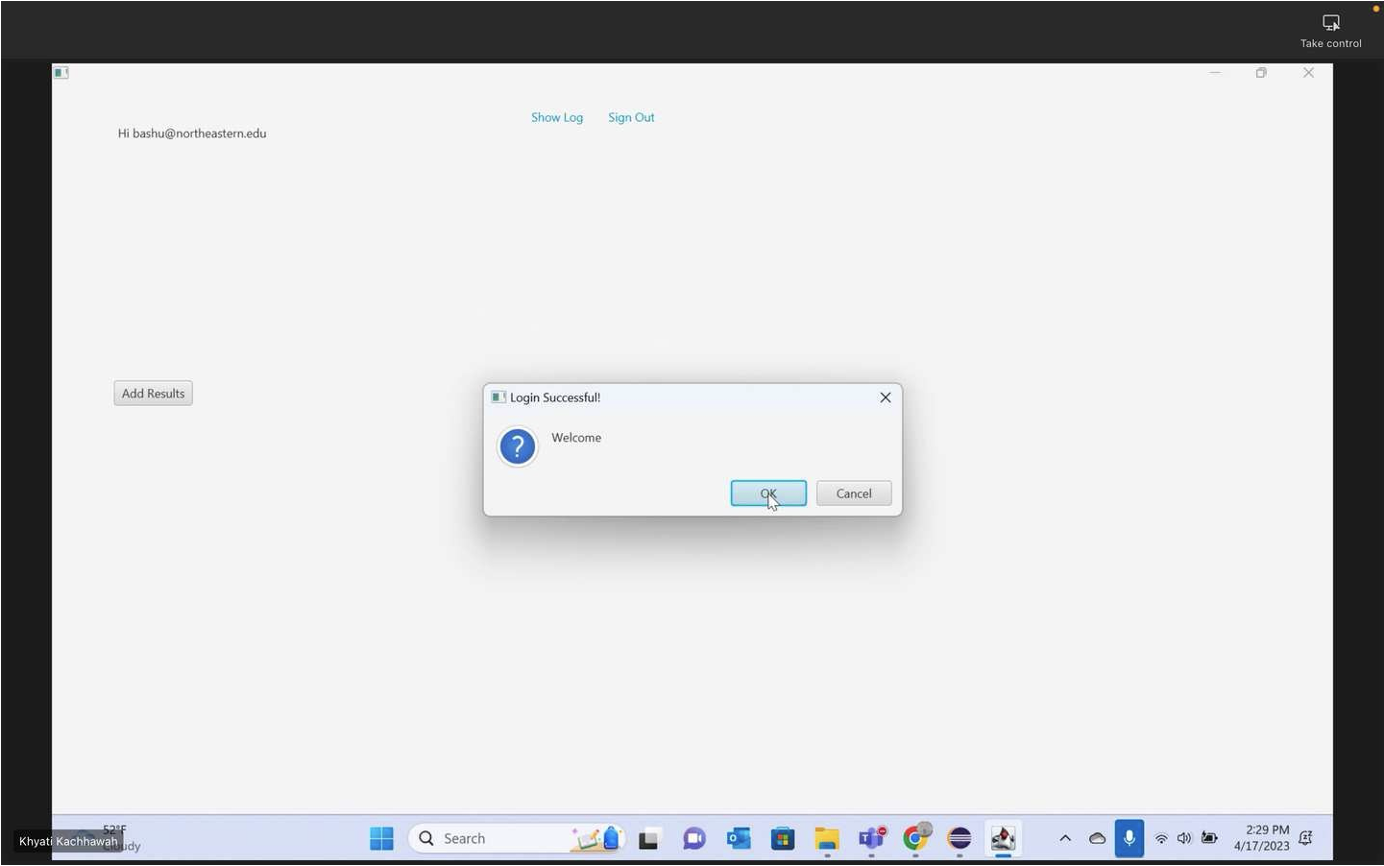
1. Registration page

This page allows new users (teachers) to create an account by entering their personal information and choosing a username and password.



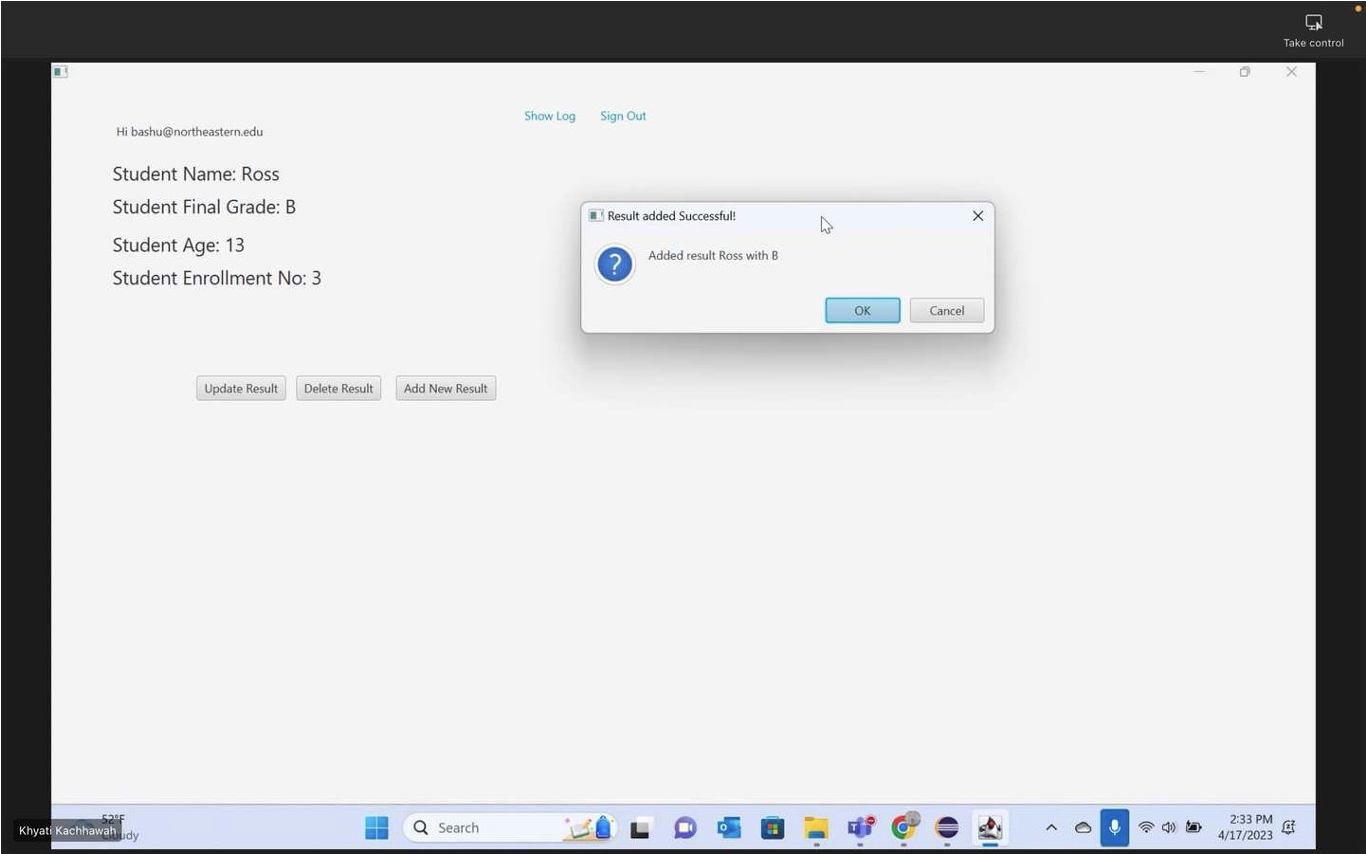
1. Login form

This form enables existing users to enter their login credentials to access the system.



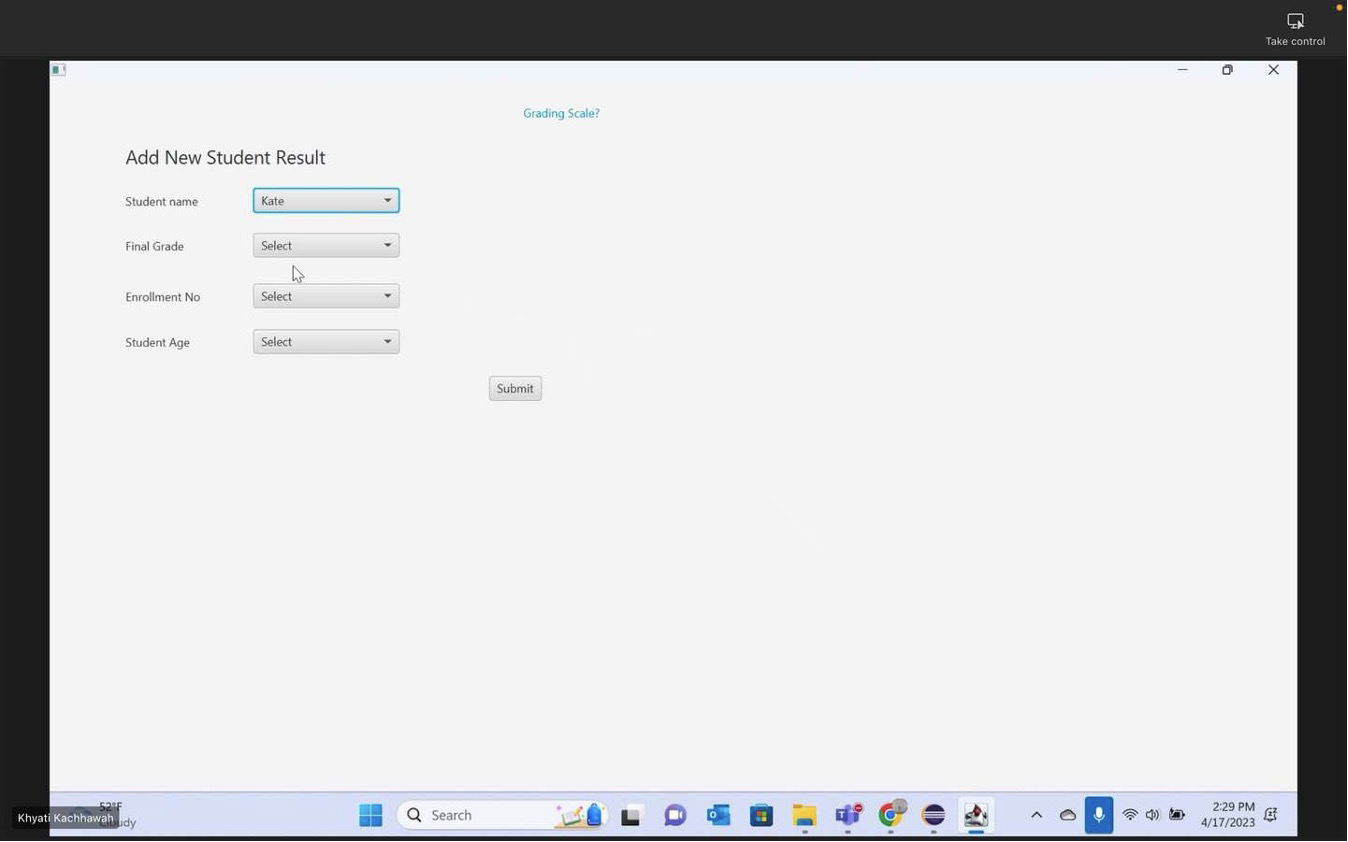
1. User Home

This page displays the options available to the user after they have logged in to add a result.



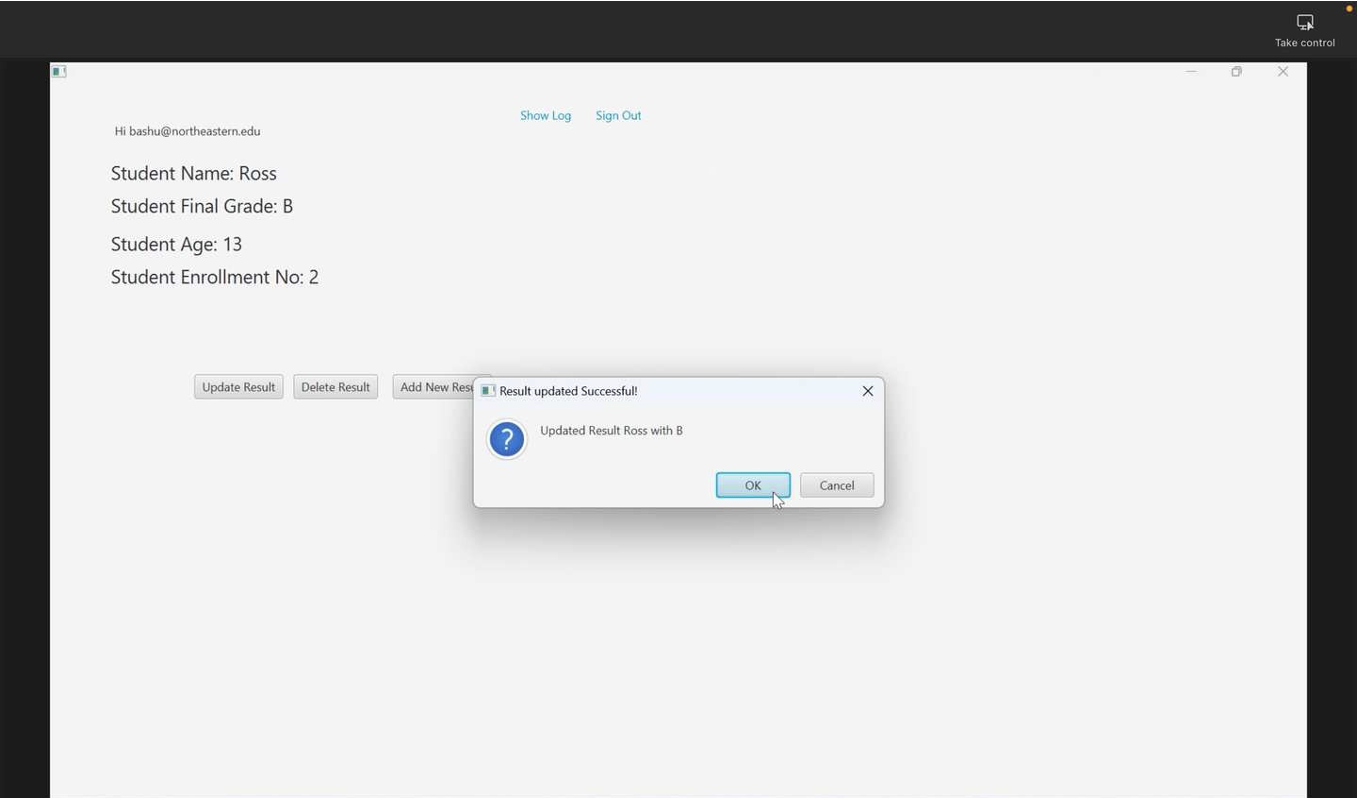
1. Student Record

This page shows the details of a specific student, including their personal information, academic history, and any past results.



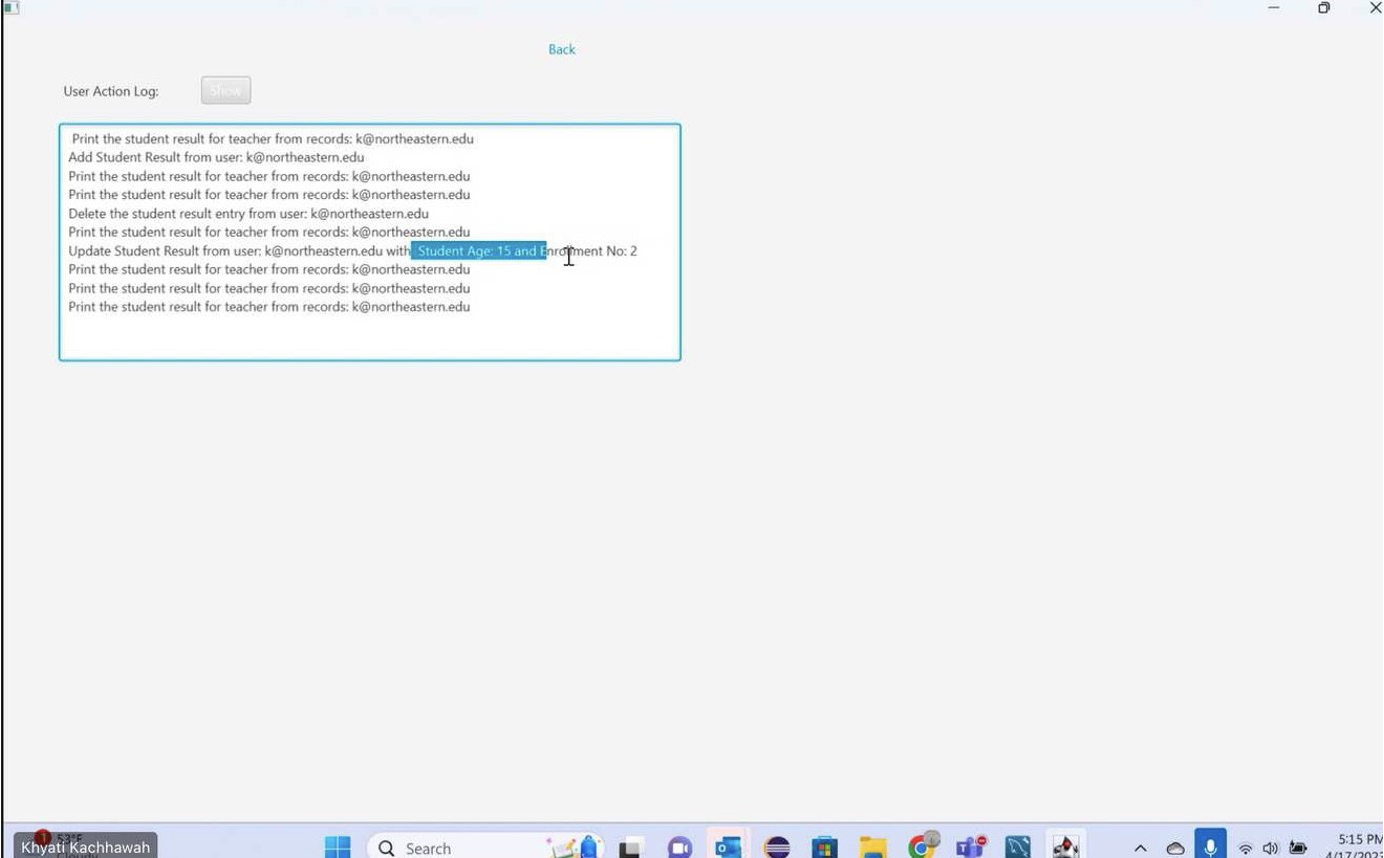
1. Add Result

This page enables the user to input the grades and other personal details for a specific student.



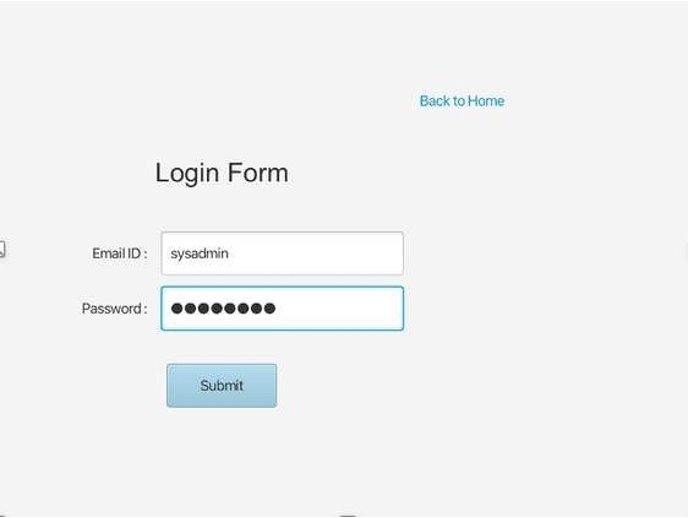
1. Update Result

This page allows the user to edit the details of an existing result for a particular student.



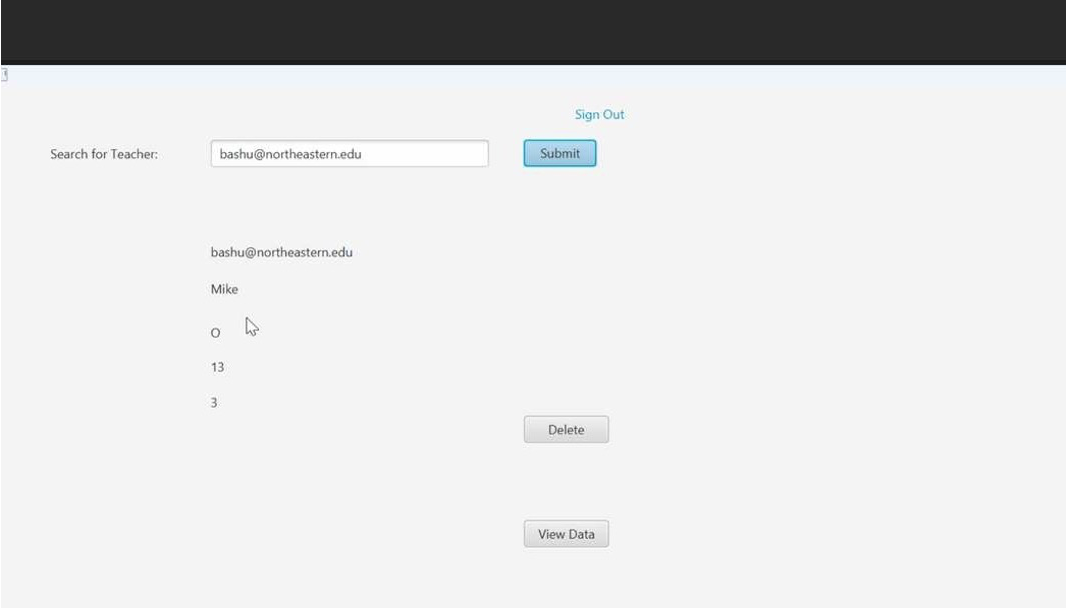
1. Activity Log

This page records all the actions done by teachers in the system, including login/logout, adding/updating results, and other system activities.



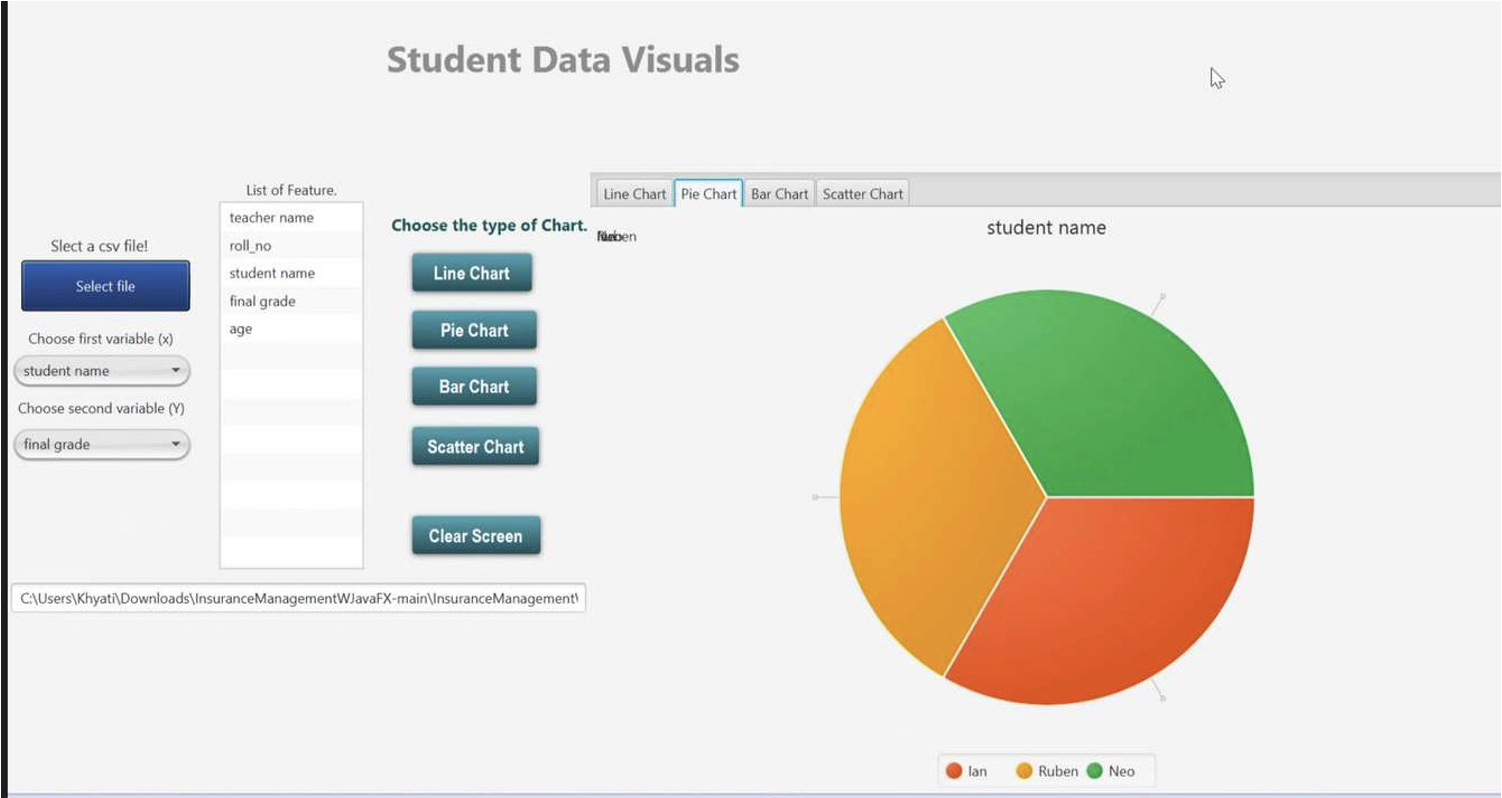
1. Admin Login

This form enables the system administrator to enter their login credentials to access the administrative functions of the system.



1. Admin Home

This page displays the options available to the administrator after they have logged in, such as managing user accounts and viewing activity logs and analyzing student’s graphical reports.



1. Visualization

This page presents the results of a student in a visual format, such as a graph, line, pie and scatter chart, to help understand and analyze the data more easily.

# **VII. Discussion (Reflection)**

On completion of the Student Result Management project, we have successfully built a system that allows for easy management and organization of student data. This project has allowed us to gain a deeper understanding of database management and data manipulation using SQL, as well as the importance of security measures when dealing with sensitive information. While working on the Student Result Management project, we were able to explore the capabilities of SQL and databases. The project involved creating a database to store the information of students and their results, and then designing the application to fetch data from the database as needed. We learned about database design and management, and how to write SQL queries to retrieve specific data from the database.

The use of Java as the main programming language has enabled us to create a modular and extensible system using object-oriented programming (OOP) concepts such as encapsulation, inheritance, and polymorphism. We have also utilized JavaFX to create a user-friendly interface, with the added benefit of Scene Builder for efficient GUI design.

The system's functionality includes the ability to add, update, and delete student information, as well as generate reports and export data in CSV format. The visualization component allows for easy analysis and comparison of student data through line charts, pie charts, bar graphs, and scatter charts. This helped us to better understand the data and identify trends and patterns that might not have been immediately apparent from looking at the raw data.

In addition, the project has emphasized the importance of testing and debugging in software development, as well as the need for clear and concise documentation for future maintenance and updates. Overall, this project has been a valuable learning experience, allowing us to apply theoretical concepts in a practical setting and develop a useful tool for student data management and gain experience in several different areas of software development, including object-oriented programming, database design and management, and data visualization. By working on this project, we were able to deepen our understanding of these concepts and develop practical skills that will be valuable in our future work as developers.

# **VIII. Conclusions and Future Work**

The implementation of the Student Result Management System can bring significant benefits to educational institutions. It can improve the accuracy, efficiency, and security of record-keeping, and provide a better understanding of student performance through data analysis. The system developed in this project offers a user-friendly interface for teachers and administrators to manage and access student data. The data visualization features also provide insights into student performance, making it easier to identify trends and areas that require attention.

Overall, the system developed in this project serves as a good foundation for further improvement and expansion. With more data and feedback from users, it can be refined to better suit the needs of specific institutions. In addition, there is potential to incorporate machine learning algorithms to improve the accuracy of predicting student outcomes based on historical data. The system can also be integrated with other educational software, such as learning management systems, to create a more comprehensive solution for educational institutions. With continuous development and improvement, a student result management system can provide valuable support to educators and contribute to better learning outcomes for students.

While the Student Result Management system provides a solid foundation for managing student records, there is still room for improvement.

One area for future work is the integration of machine learning algorithms to help predict student performance based on past results and other factors. This could help teachers and administrators to identify at-risk students and provide them with the necessary support to succeed, which is presently not a feature of the existing application.

Another area for future work is the implementation of a mobile application for the system, providing students and parents with easy access to their results and other important information. This would enhance the user experience, making it more convenient and accessible for everyone.

Additionally, the system could be expanded to include additional features such as online registration, fee payment, and academic planning tools, making it a comprehensive solution for managing all aspects of student life.

# **IX. Job Assignment**

* Bashu Bisht :
  + Building our database schema, code for Search, Register and login fxml and controllers, GitHub Project Report I, VII
  + Main fxml for display of CSV report generation and project presentation. Project Report IV, IX
  + Managing work on Trello
  + UML Diagram
* Atharva Nimbalwar :
  + System design of the application and the App.java page. Building the PostLogin and log fxml and controllers files. Project Report V, VI
  + Building the data models User, Student Result and UserSession java files. Project Report II, III and VIII.
  + Main Controller for backend Pie chart, bar chart, line chart and scatter chart
  + Building Presentation
* Khyati Kachhawah:
  + Project creation, JDBC connections
  + Generating CSV file from Database, Building the jdbcDAO class, addplan and update plan fxml and java controller
  + Maven configurations, Dependencies addition in pom.xml
  + Teams,terms for grading scale, secondary, primary fxml and java controllers
  + Validations for login forms and result
  + Updating presentation and report

##### **References**

1. <https://developer.oracle.com/technologies/databases.html#articles> : This website provides articles and resources on Oracle databases and related technologies for developers.
2. <https://docs.oracle.com/javaee/6/index.html> :

Java Platform, Enterprise Edition (Java EE) Technical Documentation

1. <https://docs.oracle.com/javafx/2/api/JavaFXfxml> :

provides the API documentation for JavaFX 2.0, a software platform for creating and delivering desktop applications, that can run across a wide variety of devices.