University Management System

Manogna Pallapothu, Padma Anaokar, Yash Pawar

Information Systems, Northeastern University
pallapothu.m@northeastern.edu, anaokar.p@northeastern.edu, pawar.ya@northeastern.edu

ABSTRACT

Our project seeks to address the absence of a dedicated application for efficiently managing essential university functions such as student enrollment, course exploration, and grade updates. The university management system we've developed features distinct user roles – Admin and Student – each assigned specific responsibilities. Admin users can effectively handle tasks such as managing student and course information, updating student grades, and visualizing gender-based enrollment trends through graphical representations on their personalized dashboard. Students, using credentials provided by the admin, can access detailed information about courses, grades, and fellow enrolled students. The system is built upon the JDBC (Java Database Connectivity) driver, a Java API designed for connecting to and querying databases, with all CRUD (Create, Read, Update, Delete) operations executed through JDBC. The application adheres to the Model-View-Controller architecture, where the Model comprises Java classes serving as object/models for data formulation, the View consists of FXML pages displayed based on user actions, and the Controller houses the primary logic for the application.

In comparison to manual record-keeping in books, our application offers significant efficiency gains. By leveraging technology, the university management system streamlines processes related to student enrollment, course management, and grade updates. The system's user roles ensure that authorized personnel can easily handle their respective tasks, reducing the likelihood of errors and enhancing overall productivity. The integration of graphical representations in the admin dashboard allows for quick insights into enrollment trends, providing a visual aid for decision-making. Moreover, the use of the JDBC driver enables seamless and secure interaction with databases, ensuring the integrity and accessibility of data. Ultimately, our application not only automates and centralizes university functions but also contributes to a more efficient and organized academic management system compared to traditional manual record-keeping methods.

I. Problem Description

Manual storage and retrieval of university data through conventional means like Excel sheets and account books is a time-consuming and error-prone process. Additionally, the associated costs of physical record maintenance pose a burden on educational institutions. The absence of a centralized platform leads to the use of multiple applications, causing inefficiencies and data fragmentation. Security concerns arise due to the lack of a streamlined login system for students, compromising the confidentiality of academic information. In the ever-evolving landscape of higher education, the need for an efficient University Management System is paramount. This project aims to develop a comprehensive JavaFX-based system that focuses on essential functionalities, including student CRUD operations, admin tasks, and grades management.

Administrator: The University Management System empowers university administrators to add new courses, including details like the course term and description. Additionally, administrators can perform various tasks such as enrolling students in courses, updating student grades, modifying student information, and accessing graphical representations indicating the total number of enrolled students.

Student: Prioritizing security, the University Management System incorporates a login feature for individual students. Each student receives unique login credentials for system access. Upon logging in, students can check their updated grades, access course details, and obtain information about other students enrolled in the same course. The system records all user actions in the database, ensuring real-time updates to the system based on user interactions.

II. Analysis (Related Work)

The University Management System provides an array of services, including student and course management, grade updates, and the presentation of enrollment trends through graphical representations. The application is

meticulously crafted to ensure seamless performance and a user-friendly interface, drawing inspiration from established management systems like NUbanner and Canvas. Our primary goal was to devise a streamlined design that encompasses essential functionalities for effective university management, catering to the needs of both staff and students and promoting efficient administrative practices.

Furthermore, the system not only enhances user experience through its intuitive design but also aims to provide a comprehensive solution for managing diverse university tasks. By incorporating best practices from other successful management systems, we endeavor to create a robust platform that contributes to a more streamlined and technologically advanced university management environment.

Discoveries: Our strategy in creating the University Management System included capitalizing on functionalities from existing applications like NuBanner, employed for course registration at Northeastern, and Canvas, utilized by students for checking grades and assignments.

Limitations: Despite the unique functionalities of both NuBanner and Canvas, no standalone application existed that amalgamated all the key features from both platforms.

Overview: Our application consolidates some of the most valuable features from both NuBanner and Canvas. This integration allows seamless logins for both students and administrators using their respective credentials, providing an easy-to-use platform for efficient task management.

III. System Design

Algorithm:

Only administrators have the authority to add students, and each student is assigned a unique account. Newly created student accounts default to the password "username123," allowing immediate access to their resources upon login. Administrators are tasked with responsibilities such as adding new courses, releasing grades, updating grades, and enrolling students. Students, have limited access and can only view their own grades (other students' grades remain concealed), see enrolled students in their course (students in other courses are not visible), and access information related to their specific course.

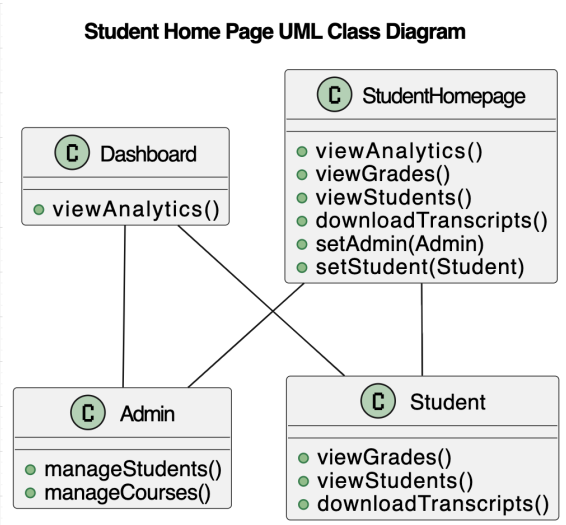
System Architecture: The application is constructed based on the MVC architecture.

Model : MVC involves the processing of business processes and rules while separating it from other layers. The model receives data from the view and returns the outcome of the business process without revealing its operations. The design of the business model is a critical aspect of MVC. It provides guidance on how to use this approach to leverage specific technological components and mitigate technical obstacles.

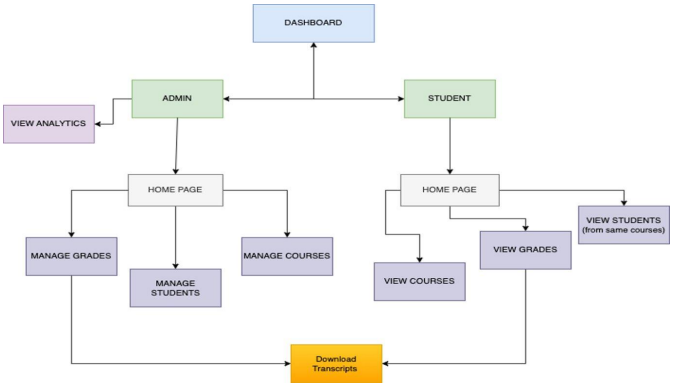
Views: As the size and complexity of an application increase, managing user interfaces and processing interfaces becomes challenging. An application may have multiple views, but the model is responsible for managing the processing of business processes.

Control: The MVC design pattern involves the interaction of the user interface, models, and views to respond to user requests. The responsibilities of the control layer are also clear in distributing tasks and providing recommendations on which models, views, and user requests to use. When a user clicks a link, the control layer receives the request without processing business data, and then passes the user's information to the appropriate model and selects the view that meets the requirements.

UML Diagram :



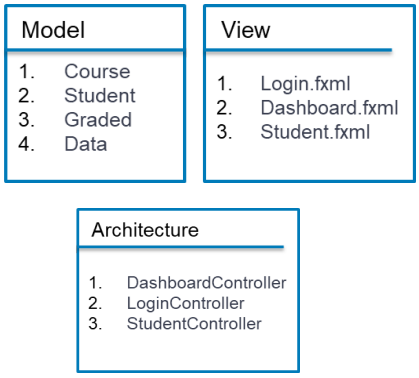
Application Flow:



IV. Implementation

For the development of the University Management System project, we adopted the Model-View-Controller (MVC) architecture. This architectural approach involves the division of application logic into three interconnected components: the model, the view, and the controller. In this structure, the model represents the data and business logic, the view corresponds to the user interface, and the controller serves as a mediator between the model and the view. The controller manages user input, ensuring updates to both the model and view as necessary.

The key motivation behind implementing this design pattern was to tackle the complexity of handling extensive data sets. By breaking down a sizable software application into distinct components, the MVC architecture aims to enhance software development, promoting a more organized and efficient approach to application design.



Controllers:

Functioning as a bridge between the model and the view, the controller manages user input and ensures synchronization between the model and view. In this specific context, the controller comprises different classes, each representing distinct actions on entities. Examples include the Login Controller, Dashboard Controller, and Student Controller, with each class dedicated to handling specific functionalities within the system. This segmented approach allows for a more modular and organized management of actions performed on the underlying entities.

Login Controller :

The Login Controller comprises a "login()" method that triggers upon the user's click on the sign-in button. To validate the user's credentials we create an SQL query, wherein input is taken from the username and password fields on the login page. An "if" statement is employed to verify whether the inputted username and password correspond to those in the database. If they do, the user is directed to their corresponding view page. Otherwise, an error message is shown.

Dashboard Controller :

This controller contains the majority of the application's logic and consists of several methods for performing the CRUD operation on Students, Courses and Grade assignment functionality. Each of these methods is explained as below:

addStudent(): This method would take all the details (input) entered by the admin on the page Dashboard.fxml and add them into the database.

updateStudent(): This method would update the student record in the database by taking the input entered by the admin. It will throw an error if any of the fields related to the student is left blank.

deleteStudent(): This method would delete a particular selected student from the database. It first checks if the data entered by the admin is present in the database or not, and then deletes using the delete query, if present.

addStudentsClear(): On click of "Clear " button, this method would clear all the entered data by setting all the fields to null or empty string.

addCourses(): This method would take all the details inputted on the screen and would add a new course in the system.

availableCourseAdd() : This method would add the course details to the list of available courses.

courseUpdate() : This method would take the course details that are entered by the admin and update it in the database.

courseDelete(): This method would delete a particular course that is selected on the screen from the database.

availableCourseList(): This method would fetch the details of the courses from the database and display it to the user.

availabelStudentsList(): This method would fetch the details of the students from the database and display it to the user.

studentsGrades(): This method would fetch the details of the student's garde from the database and display it to the student.

studentsGradesUpdate(): These methods are responsible for

adding new grades for students, as well as updating existing grades already recorded in the system.

logout(): This function logs out the current user and redirects them to the main login dashboard of the application

Student Controller :
viewGrades(): This function is employed to display a student's recorded grades within the application.

viewStudents(): This method filters and retrieves the students enrolled in a particular course, allowing them to be viewed in the user panel of the application.

Model:
Course: This class contains the setters and getters for the attributes - course(String), description(string), degree(string).

Data: This class contains the setters and getters for the attributes- username(string) and path(String).

Student: This class contains the setters and getters for the attributes- id (int), firstName (String), lastName (String), firstSem (double), secondSem (double), gender (String), dateOfBirth (Date), status (String), image (String), enrollmentYear (String), courseName (String), final(double).

View:
Login.Fxml: The primary dashboard page serves as the entry point where users input their login credentials. Once the user clicks the Login button, the LoginController handles the login operation.

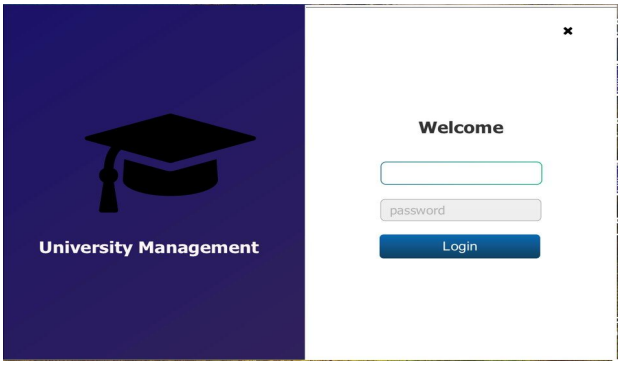
Dashboard.Fxml : This is the view page of the admin panel, where the administrator can perform their designated tasks, including adding, updating, and deleting students and courses. Additionally, the admin can also update student grades from this page.

Student.Fxml : This is the view panel for the student panel, where a student can access information such as their grades, details about their enrolled courses, and information about other students within the system

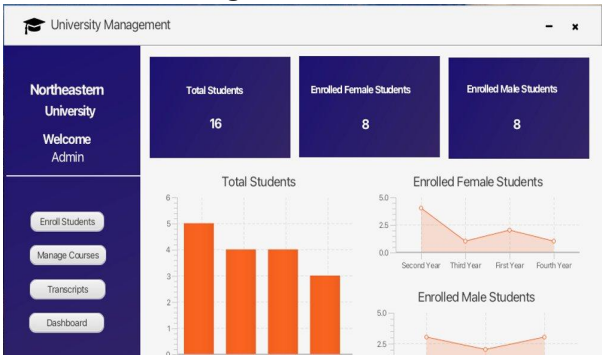
External Libraries :
In order to add visual aids to the application, we incorporated icons from the FontAwesome library and utilized JFreeChart to generate both Bar and Line Chart views within the Dashboard. Additionally, we employed MysqlConnector to establish a connection between the application and database, enabling the performance of CRUD (Create, Read, Update, and Delete) operations, we also used itext for pdf and javaxmail for email notifications.

V. Evaluation

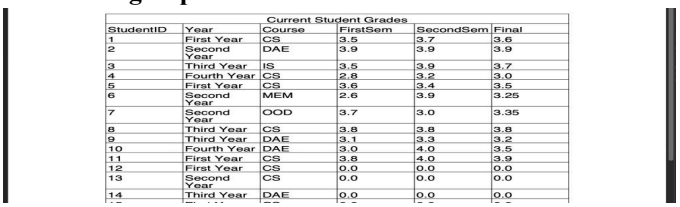
Login page:



Admin successful Login:



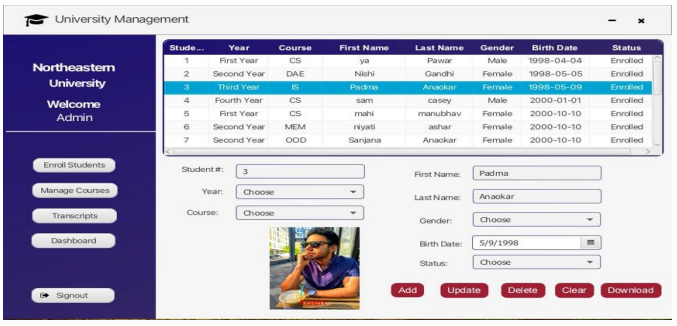
Generating Reports:



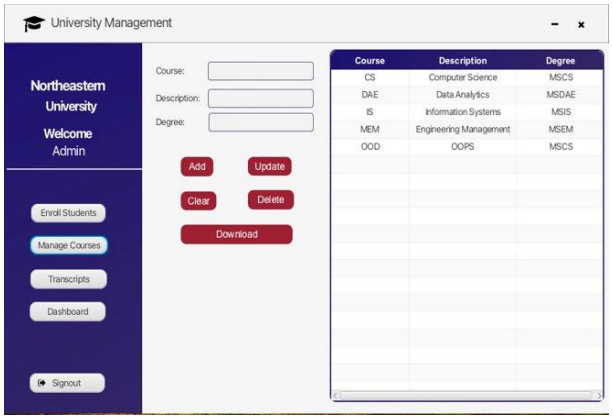
Current Student Grades

StudentID	Year	Course	FirstSem	SecondSem	Final
1	First Year	CS	3.5	3.7	3.6
2	Second Year	DAE	3.9	3.9	3.9
3	Third Year	IS	3.5	3.9	3.7
4	Fourth Year	CS	2.8	3.2	3.0
5	First Year	CS	3.6	3.4	3.5
6	Second Year	MEM	2.6	3.9	3.25
7	Second Year	OOD	3.7	3.0	3.35
8	Third Year	CS	3.8	3.8	3.8
9	Third Year	DAE	3.1	3.3	3.2
10	Fourth Year	DAE	3.0	4.0	3.5
11	First Year	CS	3.8	4.0	3.9
12	First Year	CS	0.0	0.0	0.0
13	Second Year	CS	0.0	0.0	0.0
14	Third Year	DAE	0.0	0.0	0.0

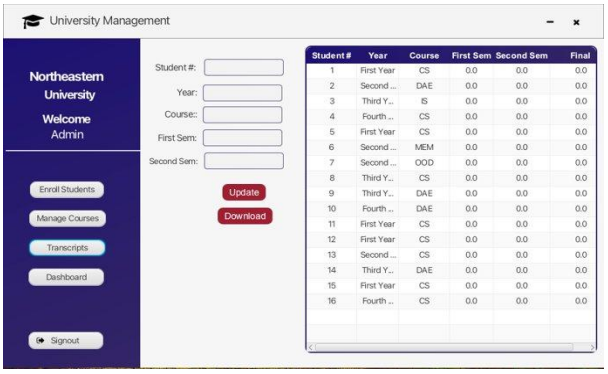
CRUD Student:



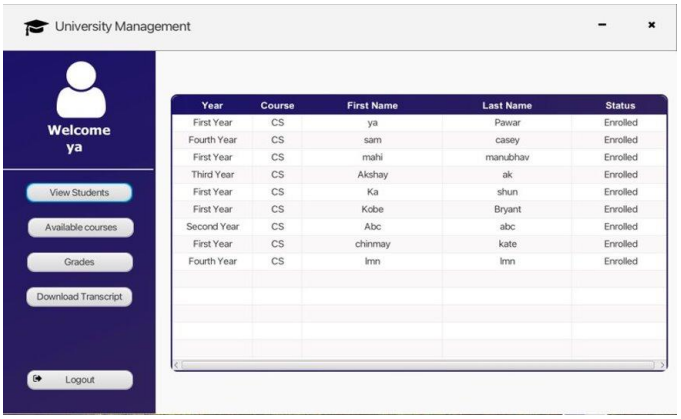
Add Course:

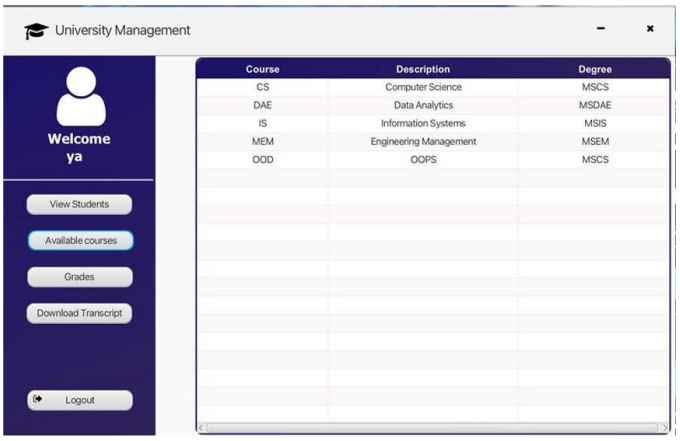


Update grades:



Student view:





VII. Discussion (Reflection)

1. The UI part can be extended with some CSS and other style properties.
2. For the database part, we used JDBC and MySQL to store all the data in the database.
3. For the framework part, we can use MVC or Spring Framework to develop our application more efficiently.

VIII. Conclusions and Future Work

At the end of the project, we came to the conclusion that the MVC architecture could be used to efficiently build the backend of the project.

The front end of the project was completed using Scene Builder. You also learned the importance of JavaFX and how to use it to build advanced applications.

It uses object-oriented principles such as inheritance, abstraction, polymorphism, interfaces, and generics to make application code more efficient. Applied object-oriented principles, including inheritance, abstraction, polymorphism, interfaces, and generics, to enhance code efficiency.

Future Work:

- Including other aspects like attendance management and enrollment management could be implemented.
- Dining Services to check meal-swipes and exchange meal swipes can also be implemented.
- Networking functionalities for students could be incorporated, enabling communication among peers. Features such as chat and messaging could be integrated, facilitating student interaction for online classes and collaborative engagement.

IX. Job Assignment

Padma Anaokar : Database integration, login page, student view page, Add courses page, Downloading Transcripts, Graphs and Analysis

Manogna Pallapothu : UI design, Cascading style sheets, Add student Page, Database entities, CRUD operations, Downloading Courses, Graphs and Analysis

Yash Pawar: System design diagram, System design, Dashboard page, Student Grades page, Crud operations, Email notification, Graphs and Analysis

X. References

[1] A. Kumar, P. Gupta, and R. Singh, "Design and implementation of university management system using Java," 2017 International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, India, 2017, pp. 43-47.

[2] M. H. Raza, M. A. Khan, and M. A. Akbar, "Development of a university management system using Java," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2016, pp. 3514-3519.

[3] S. Das, S. Mitra, and S. Chakraborty, "A Java-based university management system," 2015 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2015, pp. 1212-1216.

[4] N. N. Nweke and V. O. Akpudje, "Design and implementation of a web-based university management system using Java and MySQL," International Journal of Computer Applications, vol. 108, no. 9, pp. 23-29, December 2014.

[5] S. K. Singh and S. S. Rathore, "Development of a Java-based university management system with enhanced security features," International Journal of Computer Science and Mobile Computing, vol. 3, no. 6, pp. 909-917, June 2014.

[6] Oracle Corporation. "JavaFX 8: Introduction by Example." Oracle Corporation, 2014. https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm

[7] Oracle Corporation. "JDBC API Tutorial and Reference." Oracle Corporation, 2019. https://docs.oracle.com/javase/tutorial/jdbc/index.html

[8] Oracle Corporation. "MySQL Connector/J Developer Guide." Oracle Corporation, 2019. https://dev.mysql.com/doc/connector-j/8.0/en/

[9] JFree.org. "JFreeChart Developer Guide." JFree.org, 2023. https://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html

[10] Gofore, Antti. "JavaFX architecture - Part 1: Model-View-Controller (MVC)." Gofore, 2019. https://gofore.com/en/javafx-architecture-part-1-model-view-controller-mvc/

[11] Oracle Corporation. "JavaFX CSS Reference Guide." Oracle Corporation, 2022. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

[12] "Design and Implementation of University Management System based on JavaFX and MySQL" by Xueqin Yang, Junjie Du, and Yushan Qian. Presented at the 2017 International Conference on Computer Science and Artificial Intelligence (CSAI).

[13] "Development of University Management System Using JavaFX and MySQL" by Md. Ariful Islam and Md. Ashiqur Rahman. Presented at the 2019 International Conference on Advances in Electrical, Electronic and Communication Engineering (ICAEECE).

[14] "Design and Implementation of College Student Management System Based on JavaFX" by Hui Li, Li Li, and Jiaquan Li. Presented at the 2018 International Conference on Computer Science, Electronic Engineering, and Education Technology (CEEE).