THE EGYPTIAN
E-LEARNING UNIVERSITY

الجامعه المصريه للتعليم الكتروني

National Egyptian E-Learning University

Faculty of Computer & Information Technology

By :

Abdelrahman mahrous Mohamed	**15-00469**
Walid Atef Fayze	**20-01029**
Mohamed mamdouh mekawi	**17-00085**
Mekhail Amir Lamey	**20-00041**
Mohamed khalied refaat	**20-01490**
Ramy Hany Lamey	**20-01184**
Rokia Elbadry Mohamed	**19-00511**

Under Supervision of:

DR. Yasser Abdelhamid

(Professor of Computer Engineering and Information Technology Egyptian E-Learning University)

Eng. Dalia Masoud

(Teacher assistant at Computer Engineering and Information Technology Egyptian E-Learning University)

This documentation is submitted as a partial fulfillment of the requirements for the degree of Bachelor of Science in Computer & Information Technology

Table of contents:

Acknowledgement :

Thanks for faculty dean DR/Kamal Safwat Hamza, We would like to express our sincere appreciation and deepest thanks to our advisor Dr.Yasser Abdelhamid (professor of Information system science, faculty of computers and information) Also, thanks a lot for Eng/ Dalia masoud for their continuous support, guidance, and encouragement in achieving this work. This project would not be the way it is without his useful comments and suggestions. Thanks for helpful guidance and valuable assistance throughout the whole work.

Introduction :

A Student Evaluation System is a crucial component of educational institutions, providing a structured approach to assess students' performance, progress, and overall learning experience. Let's explore the key aspects of such a system:

Purpose and Importance:
The primary goal is to evaluate students' academic achievements, skills, and competencies.
It informs instructional improvements, curriculum enhancements, and faculty development.
Enables data-driven decision-making for educational institutions.
Components of the System:
Course Evaluations: Students provide feedback on courses, instructors, and teaching methods.
Peer Reviews: Faculty members assess each other's teaching effectiveness.
Self-Assessment: Students reflect on their own learning and growth.
-360Degree Feedback: Involves input from peers, instructors, and self-assessment.
Benefits:
Quality Enhancement: Identifies strengths and areas for improvement.
Accountability: Holds instructors and institutions accountable.
Student Engagement: Encourages active participation and ownership.
Continuous Improvement: Drives ongoing enhancements in teaching and learnin

Personalized sessions with academic counselors.
Discussing course selection, major choices, and academic progress.
Addressing academic challenges and setting goals.
Group Workshops and Seminars:
Interactive sessions on study skills, time management, and stress management.
Facilitating peer learning and collaboration.
Enhancing students' academic performance.
Career Counseling:
Exploring career paths, job market trends, and industry requirements.
Resume building, interview preparation, and networking strategies.
Connecting students with internship opportunities.
Study Abroad Guidance:
Assisting students in planning study abroad experiences.
Navigating visa requirements, cultural adjustments, and academic expectations.
Encouraging global perspectives

Benefits of the Student Evaluation System:

1. Reduced wait times for appointments to student
2. Enhanced Feedback: A well-designed evaluation system allows students to provide feedback on courses, instructors, and the overall learning experience.
3. This feedback can lead to improvements in teaching methods, curriculum, and student support services.
4. Quality Assurance: Regular evaluations help maintain educational standards. By assessing teaching effectiveness, course content, and student satisfaction, institutions can ensure consistent quality across programs.
5. Individualized Support: Evaluation data can identify struggling students early on. With this information, educators can offer personalized support, such as tutoring, counseling, or study strategies.
6. Curriculum Enhancement: Evaluation results can guide curriculum development
7. Faculty Development: Constructive feedback from students helps instructors improve their teaching skills

Abstract :

• Objective

Abstract Student evaluations of teaching are ubiquitous in the academe as a metric for assessing teaching and frequently used in critical personnel decisions. Yet, there is ample evidence documenting both measurement and equity bias in these assessments. Student Evaluations of Teaching (SETs) have low or no correlation with learning. Furthermore, scholars using different data and different methodologies routinely find that women faculty, faculty of color, and other marginalized groups are subject to a disadvantage in SETs. Extant research on bias on teaching evaluations tend to review only the aspect of the literature most pertinent to that study. In this paper, we review a novel dataset of over 100 articles on bias in student evaluations of teaching and provide a nuanced review of this broad but established literature. We find that women and other marginalized groups do face significant biases in standard evaluations of teaching – however, the effect of gender is conditional upon other factors. We conclude with recommendations for the judicious use of SETs and avenues for future research.

1-**Fair Assessment**: Student evaluation systems aim to provide an equitable and unbiased assessment process. Students should be evaluated based on their actual performance, without external influences.
2-**Feedback and Improvement**: These systems offer constructive feedback to students, highlighting their strengths and areas for growth. Such feedback fosters continuous improvement.
3-**Curriculum Alignment**: Evaluations ensure alignment with educational goals and specified learning outcomes. Assessments should reflect the curriculum's objectives.
4-**Identifying Learning Gaps**: By assessing student understanding and skills, educators can identify gaps and tailor teaching methods accordingly.
5-**Institutional Accountability**: Evaluation results contribute to institutional accountability and drive quality enhancement.

Benefits:

Enhanced Learning Experience: Constructive feedback helps students enhance their learning journey**.**
Quality Assurance: Evaluation results inform curriculum design and teaching effectiveness.
Holistic Assessment: Beyond exams, assessments consider practical skills, critical

thinking, and application.
Competency-Based Assessment Models: Competency-based assessments focus on specific competencies (knowledge, skills, attitudes).

Key features include: Real-World Tasks: Assessments involve practical tasks, projects, and demonstrations.
Application Readiness: Emphasis on practical application prepares students for professional roles.

- # Recommendations for Better Evaluation

There are many well-documented ways that measurement bias and equity bias shape student evaluations of teaching. Given these biases and their role in personnel decisions, many colleges and universities – sometimes at the behest of faculty unions – are reevaluating how these ratings are used. We argue that we need not "throw out the baby with the bathwater" and eliminate the role of student evaluations entirely. Rather, they should be properly contextualized and used with caution. We offer the following six actions that individual faculty members and universities can take to make the use of student ratings more responsible.

- # Overview

Measurement Bias:
SETs do not consistently correlate with student learning outcomes.
The assessment process can be problematic due to biases.
Faculty members' gender, race, and other factors impact evaluation results.
Equity Bias:
Women faculty, faculty of color, and marginalized groups face disadvantages in SETs.
Gender bias interacts with other factors.
Recommendations:
Use SETs judiciously.
Explore alternative evaluation methods.
Address biases in personnel decisions.

1-Introduction: A brief introduction to the concept of student evaluations of teaching (SETs), which are widely used in academia as a metric for assessing teaching quality and are often utilized in critical personnel decisions1.

2-Problem Statement: The identification of issues with SETs, such as measurement and equity bias, which can affect their reliability and fairness. SETs have been found to have low or no correlation with actual student learning1.

3-Literature Review: A summary of the extensive literature documenting the problems with SETs, including biases against certain instructors, particularly women and other marginalized groups1.

4-Methodology: An explanation of the methods used to review the literature, such as analyzing a novel dataset of over 100 articles on bias in SETs1.

5-Findings: The main findings from the literature review, which indicate significant biases in standard evaluations of teaching and the conditional effects of gender and other factors1.

6-Recommendations: Suggestions for ethical reform and the judicious use of SETs, based on the findings of the review1.

7-Conclusion: A closing statement summarizing the need for a critical examination of SETs and the importance

- # Analysis

An analysis of a student evaluation system (SES) typically involves a comprehensive examination of its effectiveness, fairness, and impact on educational outcomes.

Effectiveness: SES is designed to measure teaching quality and provide feedback for instructional improvement. However, studies have shown that student evaluations have low or no correlation with actual student learning, raising concerns about their effectiveness as a measure of teaching quality1.

Fairness: There is ample evidence of measurement and equity bias in SES. Women faculty, faculty of color, and other marginalized groups often face disadvantages in evaluations, which can contribute to disparities in tenure, promotion, and pay decisions1.

Impact: The reliance on SES for critical personnel decisions is widespread, despite the known issues. This can perpetuate inequities within the academic system and affect the career trajectories of those subjected to biases1

Recommendations: To address these issues, researchers suggest ethical reforms and the judicious use of SES. This includes reducing biases and considering alternative

evaluative systems that may provide a more accurate and equitable assessment of teaching effectiveness1.

Conclusion: The analysis underscores the need for a critical reevaluation of SES. Ensuring that evaluations are fair and effectively measure teaching quality is crucial for the integrity of the educational system and the well-being of its educators.

This analysis reflects the growing consensus that while SES can be a valuable tool for feedback, its current implementation requires significant improvement to serve its intended purpose without introducing bias or unfairness

History :

The history of student evaluation systems is a fascinating journey that reflects the evolution of educational practices and the increasing demand for accountability in education .

Early Beginnings: The roots of student evaluation can be traced back to European universities in the 1100s, where scholars evaluated student learning through oral disputes, debates, and arguments relative to the judgment of higher-ranked experts1.

Standardized Testing: The 20th century saw the rise of formal group-administered testing for qualifications and selection, with the development of statistical methods for scoring tests, such as classical test theory and item response theory1.

Technological Advancements: With the advent of personal computing, tests began to be delivered on-screen and through the web with adaptive scoring based on student performance. This era introduced a more sophisticated analysis of test-takers' processes1.

Contemporary Challenges: Despite technological progress, testing has often neglected the psychological, cultural, and contextual factors related to test-taker psychology. Computer testing has also been criticized for neglecting school curriculum and classroom contexts, where most education takes place1.

Future Directions: The field is now on the cusp of significant development as greater emphasis on the psychology of assessment is integrated into testing. The future of educational assessment lies in the integration of psychological theory and research with statistics and technology to support valid and reliable assessment in living classrooms1.

This historical perspective highlights the ongoing efforts to refine student evaluation systems to better align with educational goals and the realities of classroom learning

Motivation :

Motivation plays a crucial role in the context of student evaluation systems. It not only influences how students perceive and engage with the evaluation process but also impacts their overall academic performance and learning outcomes .

The Role of Motivation:

Intrinsic Motivation: Students who are intrinsically motivated tend to engage more deeply with the learning material and are more likely to respond positively to evaluations that align with their interests and goals1.

Extrinsic Motivation: External incentives, such as grades or rewards, can motivate students to perform well in evaluations. However, over-reliance on extrinsic motivators can undermine intrinsic motivation2.

Theories of Motivation in Education:

Expectancy-Value Theory: This theory suggests that students' motivation is influenced by their expectations of success and the value they place on the success1.

Self-Determination Theory: According to this theory, fulfilling the needs for autonomy, competence, and relatedness is essential for fostering intrinsic motivation1.

Achievement Goal Theory: This theory posits that students' approach to learning and evaluation is guided by their goals, which can be oriented towards mastery or performance1.

Implications for Student Evaluation Systems:

Designing Evaluations: Evaluation systems should be designed to enhance intrinsic motivation by providing meaningful feedback and aligning with students' personal and academic goals1.

Self-Assessment: Incorporating self-assessment can promote intrinsic motivation by allowing students to monitor and evaluate their own learning, leading to greater engagement and higher achievement3.

Ethical Considerations: It's important to ensure that the evaluation system is fair and unbiased, as perceived fairness can significantly affect students' motivation and trust in the system1.

Conclusion: A student evaluation system that considers motivational factors can lead to more engaged learning, better academic performance, and a more accurate reflection of students' abilities. By integrating motivational theories into the design and implementation of evaluation systems, educators can create a more supportive and effective educational environment.

This analysis underscores the importance of understanding and incorporating motivational aspects into student evaluation systems to enhance their effectiveness and the educational experience.

Problem statement

A student evaluation system is an essential component of any educational system for several important reasons:

Measuring understanding and achievement: It helps measure the extent to which students understand the curriculum and achieve the required knowledge. Through it, the extent to which the set educational goals have been achieved can be determined.

Identify strengths and weaknesses: The assessment system provides indicators of students' strengths and weaknesses, allowing teachers to customize the support and guidance needed to improve performance.

Improving the educational process: Helps determine the effectiveness of teaching methods and educational programs. Through it, continuous modifications and improvements can be made to ensure the quality of education.

Encouraging interaction and motivation: The evaluation system can be a way to motivate students to put in more effort and interact positively with the educational process, by providing continuous feedback.

Providing data to improve education: The evaluation system provides data and statistics that can be used to analyze academic performance and develop improvement plans based on objective facts and data.

Preparing students for final exams and practical life: Through continuous assessment, students are prepared to deal with final exams and future challenges, whether in study or in working life.

Ensuring fairness and objectivity: It contributes to providing an objective and fair assessment for all students, which enhances the chances of achieving equality in educational opportunities.

Enhancing communication between students, teachers and parents: It contributes to building effective communication channels between all concerned parties, which helps in monitoring students' progress and providing the necessary support.

Determine proficiency levels and educational standards: It helps determine the levels of proficiency required in various academic subjects and ensure the achievement of the required educational standards.

In general, the evaluation system is an integral part of the educational process, as it contributes to improving the quality of education, achieving academic goals, and developing students' abilities in a sustainable manner.

Solution statement

Improving the quality of education: Regular assessment helps identify students' strengths and weaknesses, enabling teachers to adjust teaching methods and provide appropriate support.

Student guidance: Assessment helps guide students towards areas they need to improve, and contributes to the development of their academic and personal skills.

Performance Measurement: The assessment system provides an objective way to measure students' progress and performance, which helps determine the extent to which educational goals have been achieved.

Providing useful data: The evaluation system provides data and statistics that help educational departments make informed decisions about curricula and educational programs.

Enhancing competitiveness: An assessment system can encourage positive competition among students, pushing them to perform better.

Motivating students: Assessment can have a motivational effect on students, as they strive to achieve good results and obtain high grades.

Evaluating teachers' effectiveness: Evaluation helps evaluate teachers' performance and the effectiveness of their teaching methods, which contributes to improving the quality of education.

Improving educational planning: It helps in planning educational activities and programs based on the evaluation results, which enhances the effectiveness of the educational process.

Strengthening communication between concerned parties: The assessment system facilitates communication between teachers, students, and parents, enhancing everyone's understanding of the level of academic achievement and any challenges the student may face.

Organization Report :

Task Name	Purpos	Assigned To	Attendees	Assigning Data	Done
Similar System Survey				**28/10/2023**	
Relational Schema	To construct a basic relational schema of the system	1. Mekhail Amir 2. Walid Atef	All	**31/10/2023**	
Generate Shema	Schema from ERD	1. Rokia Elbadry	All	**2/11/2023**	
Use Case Diagram	• Understand the requirements • Identify the actors • Simplify complexity	1. Abdelrahman Mahrous 2. Mohamed Mamdouh	All	**2/11/2023**	

Sequence diagram	• Explain the interaction between the elements • Design and documentation • Detect potential problems	1. Mohamed Khalied 2. Ramy Hany	All	**2/11/2023**	
User Interface	• Facilitate interaction • provide information • Orient users	1. Abdelrahman Mahrous 2. Mohamed Khalied 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mekhail Amir 7. Mohamed Mamdouh	All	**4/11/2023**	
Design home page		1. Abdelrahman Mahrous 2. Mohamed Khalied 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mekhail Amir 7. Mohamed Mamdouh	All	**8/11/2023**	
Design Login page		1. Abdelrahman Mahrous 2. Mohamed Khalied 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mekhail Amir 7. Mohamed Mamdouh	All	**8/11/2023**	

Design Registration page		1. Abdelrahman Mahrous 2. Mohamed Khalied 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mekhail Amir 7. Mohamed Mamdouh	All	**8/11/2023**	
Design Services pages		1. Abdelrahman Mahrous 2. Mohamed Khalied 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mekhail Amir 7. Mohamed Mamdouh	All	**8/11/2023**	
Design booking page		1. Abdelrahman Mahrous 2. Mohamed Khalied 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mekhail Amir 7. Mohamed Mamdouh	All	**8/11/2023**	
Design doctors pages		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mohamed Khaied 7. Mohamed Mamdouh	All	**8/11/2023**	
Design contact us page		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Rokia Elbadry 6. Mohamed Khaied 7. Mohamed Mamdouh	All	**15/11/2023**	

Turning into front-end using Angular					
Home page form	make a home page to our visitor to define our services	1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Mamdouh Mohamed Khaied 6.	All	**20/2/2024**	
login page form		1. Abdelrahman Mahrous 2. Mekhail Amir	All	**24/2/2024**	
About-us page form		1. Ramy Hany 2. Walid Atef 3. Rokia Elbadry	All	**24/2/2024**	
Add-student page form		1. Mohamed Khaied 2. Mohamed Mamdouh	All	**25/2/2024**	
All Services page form		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Khalied 6. Mohamed Mamdouh	All	**5/3/2024**	
Assignment page form		1. Ramy Hany 2. Walid Atef 3. Mohamed Khalied	All	**17/3/2024**	

Countact-us page form		1. Mekhail Amir 2. Ramy Hany 3. Mohamed Mamdouh 4. Mohamed Khalied	All	**20/3/2024**	
Courses page form		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Walid Atef	All	**26/3/2024**	
Final-grades page form		1. Ramy Hany 2. Walid Atef 3. Mohamed Khalied	All	**15/4/2024**	
Mid-grades page form		1. Abdelrahman Mahrous 2. Mohamed Mamdouh	All		
Profile page form		1. Mekhail Amir 2. Ramy Hany 3. Walid Atef	All	**20/4/2024**	
Quizes page form		1. Ramy Hany 2. Walid Atef 3. Mohamed Khalied 4. Abdelrahman Mahrous	All	**21/4/2024**	
Send-notification page form		1. Mekhail Amir 2. Mohamed Khalied 3. Walid Atef	All	**22/4/2024**	

Services page form		1. Abdelrhman Mahrous 2. Mohamed Mamdoh 3. Mekhail Amir	All	**30/4/2024**	
Show-graph page form		1. Abdelrhman Mahrous 2. Mohamed Mamdoh 3. Mekhail Amir	All	**1/5/2024**	
Student-evaluation page form		1. Ramy Hany 2. Walid Atef 3. Mohamed Khalied	All	**2/5/2024**	
Students page form		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Khalied 6. Mohamed Mamdouh	All	**8/5/2024**	
Services for Student pages		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Khalied 6. Mohamed Mamdouh	All	**10/5/2024**	
Services for Doctor pages		1. Mohamed Mamdouh 2. Walid Atef 3. Mekhail Amir	All	**13/5/2024**	
Services for Academic Advisor pages		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Khalied 6. Mohamed Mamdouh	All	**17/5/2024**	

Back-end using PHP					
Validation					
Validate the system	Validate all modules	1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Khalied 6. Mohamed Mamdouh	All	**25/6/2024**	
Presentation		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Khalied 6. Mohamed Mamdouh	All	**20/2/2023**	
Project documentation		1. Abdelrahman Mahrous 2. Mekhail Amir 3. Ramy Hany 4. Walid Atef 5. Mohamed Khalied 6. Mohamed Mamdouh	All	**9/6/2024**	

Overall description :

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented

Product Perspective :

The product is an educational management system designed to streamline and enhance the interaction between professors, teaching assistants (TAs), academic advisers, and students. The system aims to support academic processes, communication, and resource management within an educational institution.

Product Function :

The primary functions of the system include:

- **Course Management**: Professors and TAs can create, manage, and update course content, schedules, and assignments.
- **Student Management**: Academic advisers and professors can monitor student progress, performance, and provide feedback.
- **Communication Platform**: Facilitates communication between all users through messaging, announcements, and discussion forums.
- **Resource Allocation**: Management of educational resources such as reading materials, lecture notes, and multimedia content.
- **Assessment and Grading**: Tools for creating quizzes, exams, and assignments, as well as grading and providing feedback.
- **Scheduling and Calendar**: Organizes class schedules, office hours, deadlines, and appointments.
- **Data Analytics**: Provides insights into student performance, engagement, and course effectiveness.

User Characteristics :

Professor

Roles:

- Course creator, add new students to their courses,

- view student profiles.
- add midterm and final grades

Needs:

- Efficient course management, student performance tracking, communication with TAs and students.

Technical Proficiency: Moderate to high.

Frequency of Use: Daily to weekly.

Teaching Assistant (TA)

Roles:

- Ability to add quizzes or assignments.
- Conduct behavioral assessment for students support.

Needs:

- Access to course materials, grading tools, student communication.

Technical Proficiency: Moderate to high.

Frequency of Use: Daily.

Academic Adviser

Roles:

- Receive student reports and generate charts illustrating student behavior and issues, along with recommendations for improvement.

Needs:

- Tools for tracking student progress, scheduling meetings, advising on course selection.

Technical Proficiency: Moderate.

Frequency of Use: Weekly.

Student

Roles:

- Learner, assignment submitter, Receive reports and notifications from Academic Advisers.

Needs:

- Access to course materials, submission portals, communication with professors and TAs.

Technical Proficiency: Varies (low to high).

Frequency of Use: Daily.

System Features :

For Professors

- **Course Creation Tools**: Simplified interface for creating and organizing course content.
- **Grading System**: Efficient grading tools with customizable rubrics and feedback options.
- **Student Analytics**: Dashboard displaying student performance metrics.
- **Communication Tools**: Email, forums, and chat options for interacting with students and TAs.

For Teaching Assistants

- **Assignment Management**: Tools for collecting, grading, and returning assignments.
- **Student Interaction**: Platforms for holding office hours and answering student queries.
- **Content Access**: Full access to course materials and teaching resources.

For Academic Advisers

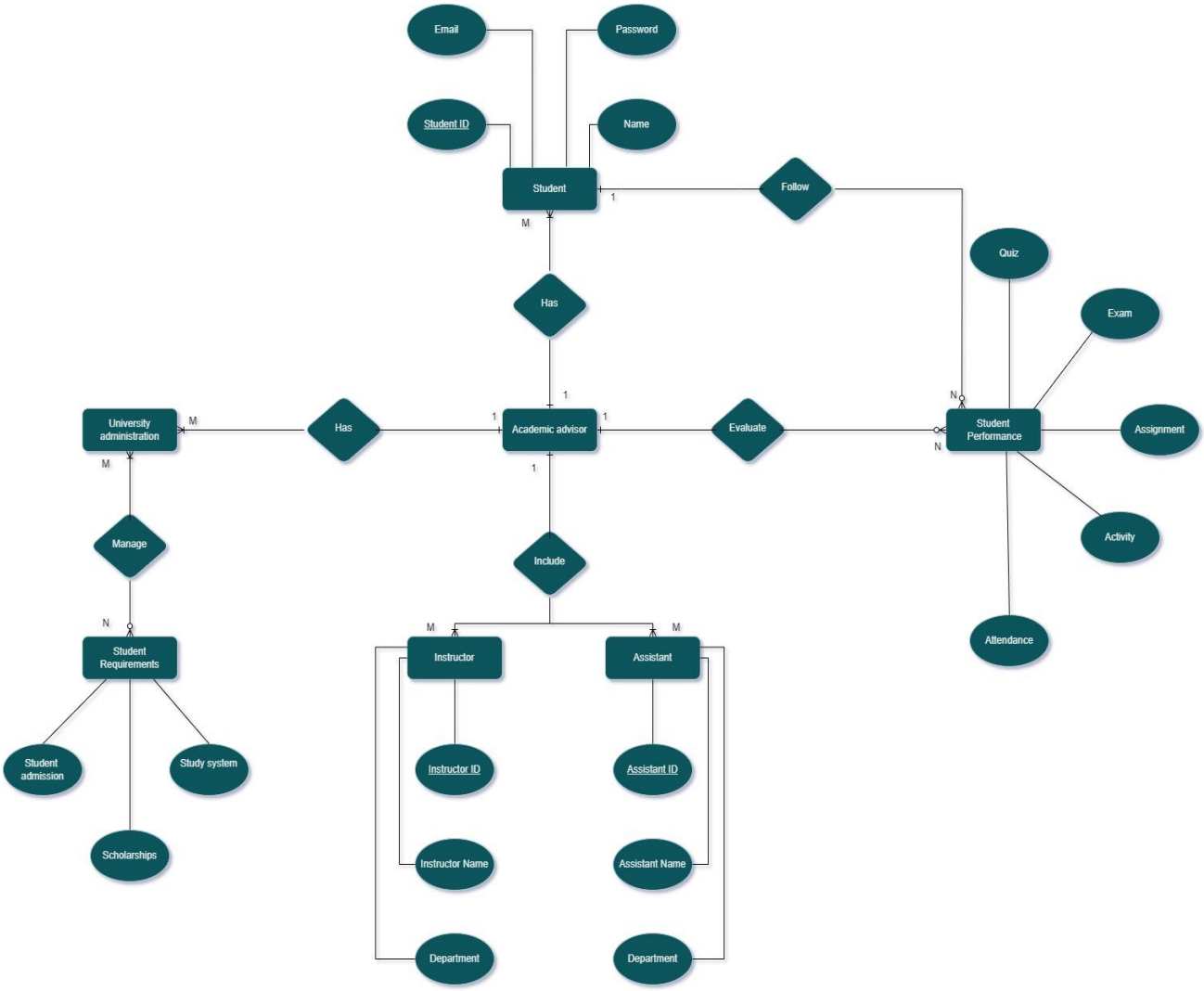
- **Student Profiles**: Comprehensive profiles detailing student academic history and performance.
- **Appointment Scheduling**: Tools for setting up and managing advising sessions.
- **Advising Resources**: Access to academic planning tools and resources.

For Students

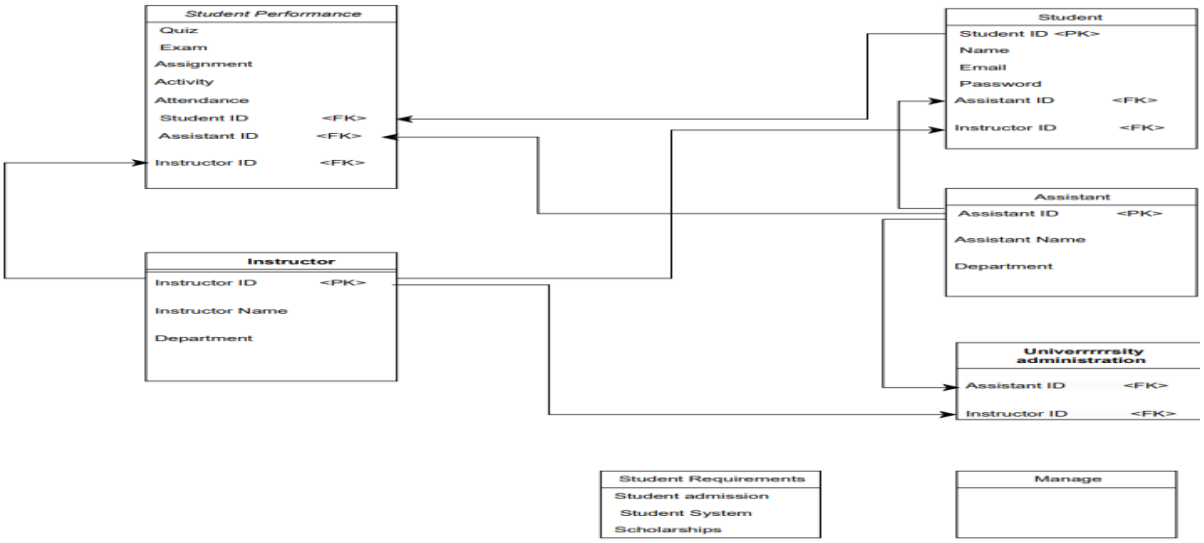
- **Course Materials**: Easy access to all course-related documents and multimedia.
- **Assignment Submission**: User-friendly submission portals with deadline tracking.
- **Progress Tracking**: Personal dashboard showing grades, feedback, and overall performance.

Communication Platforms: Messaging systems for contacting professors, TAs, and peers

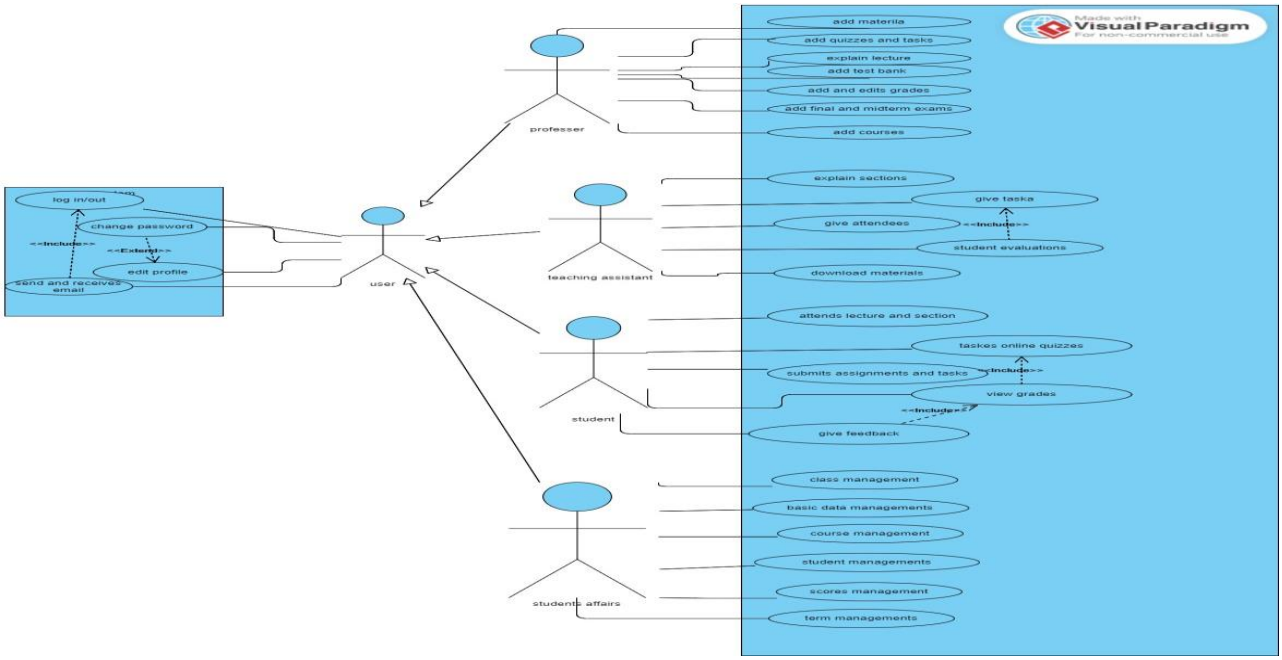
Entity relation Diagram:



Schema :



Use case:



Specific requirements :

Objectives and Goals:

Clearly define the objectives and goals of your student evaluation system. What do you want to achieve? Is it assessing student learning, improving teaching, or enhancing accountability?

Assessment Content:

Incorporate various dimensions into your evaluation system, such as:

- Professional Knowledge and Skills: Assess students' subject-specific expertise.
- Methodological Ability: Evaluate their research and problem-solving skills
- Social Ability: Consider communication, teamwork, and interpersonal skills.
- Emotional Attitude and Values: Assess character, ethics, and emotional intelligence.

Evaluation Methods:

Choose appropriate assessment methods:

- Oral Questioning: Simple and direct.
- Computer-Adaptive Testing: Complex algorithms based on learning progressions.
- Authentic Assessments: Engage students in real-world tasks.
- Higher-Order Thinking Skills: Use a variety of instructional and assessment strategies.

Timeline:

Set up a timeline for evaluation activities

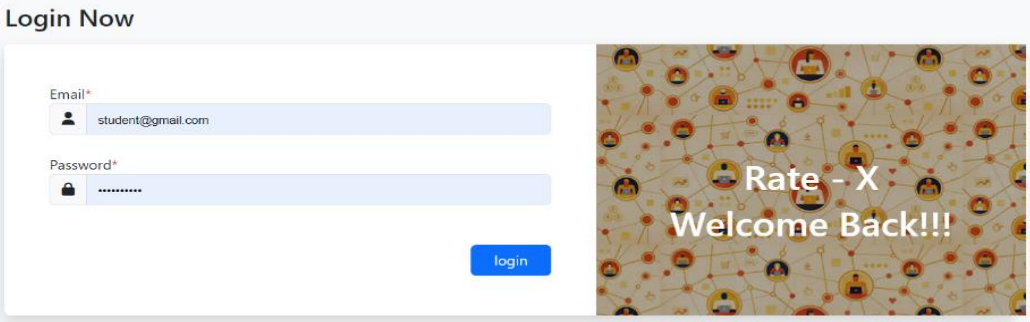
External interface requirements

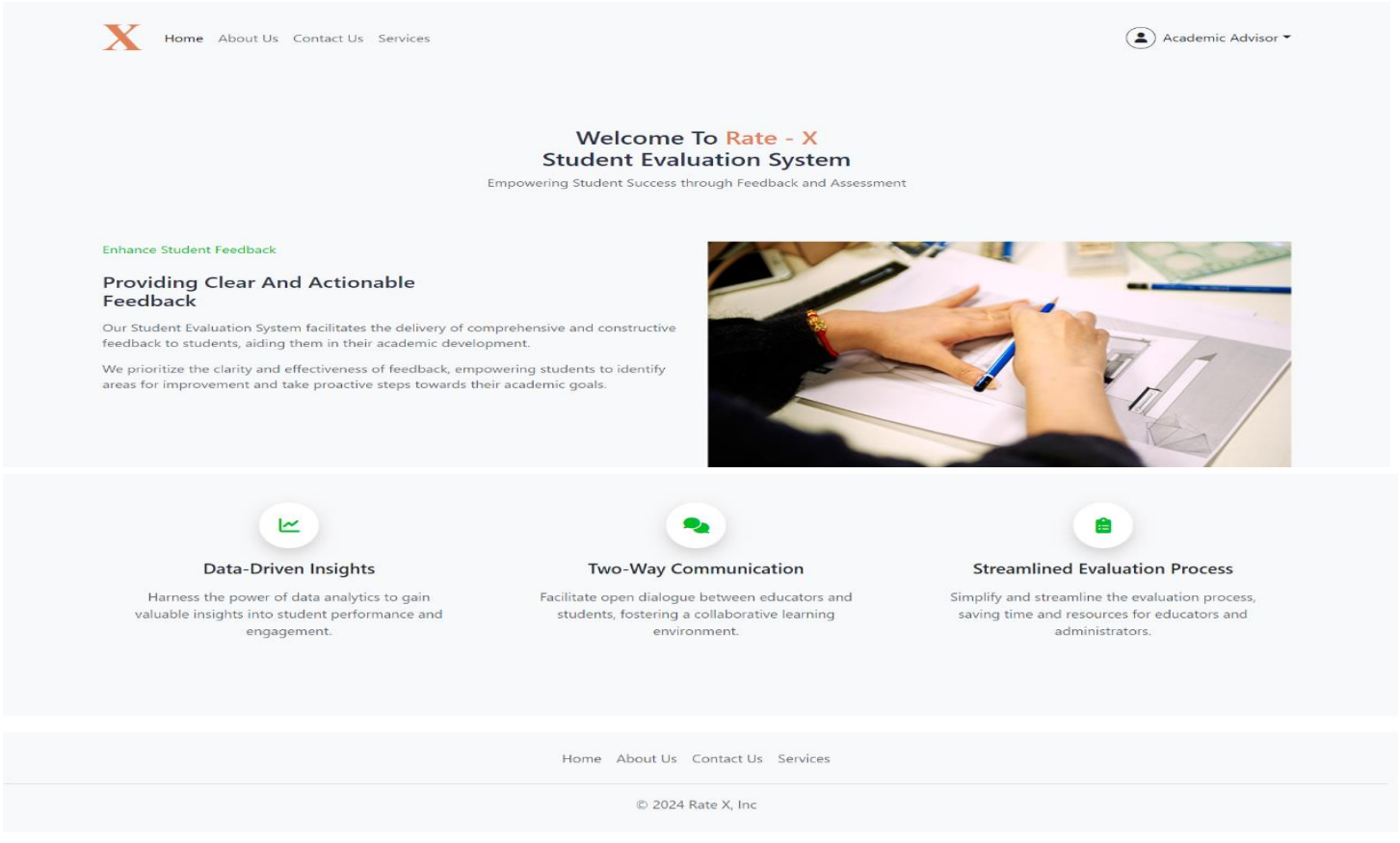
User Interfaces: These define the interaction logic between the software and users. They encompass screen layouts, buttons, and functions on each screen.

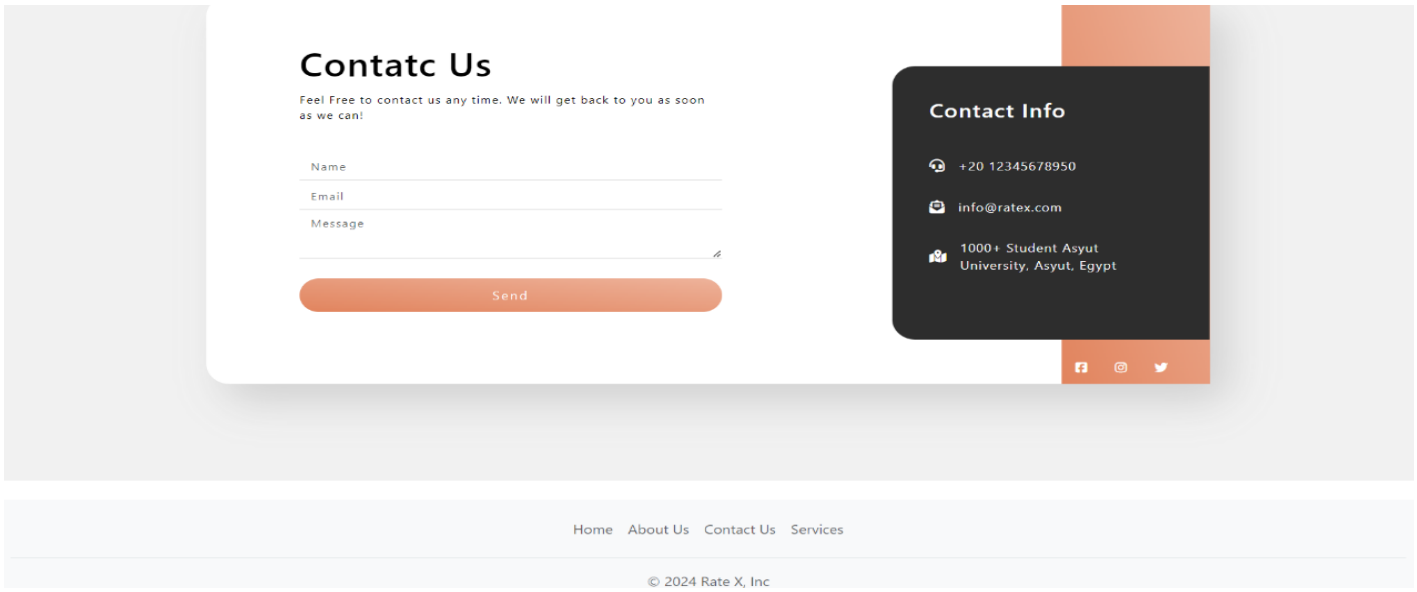
Hardware Interfaces: Describes the devices the software is designed to work with (e.g., computers, tablets, and smartphones).

Software Interfaces: Specifies how the student evaluation system communicates with other software components or systems.

User interface :



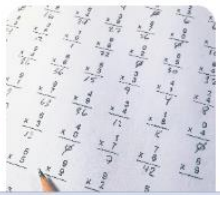




Thanks for being here, Doctor

Your dedication means a lot to us. As we get ready for another round of teaching, I just want to say a quick thanks.

Our Services


All Students

Student Mid Grades

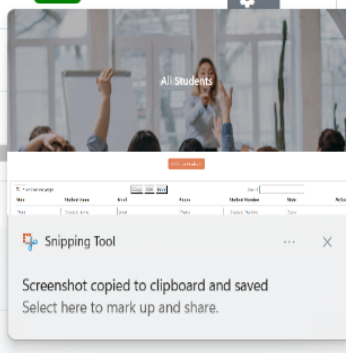
Student Final Grades

Add new Students

10 entries per page Copy CSV Print Search:

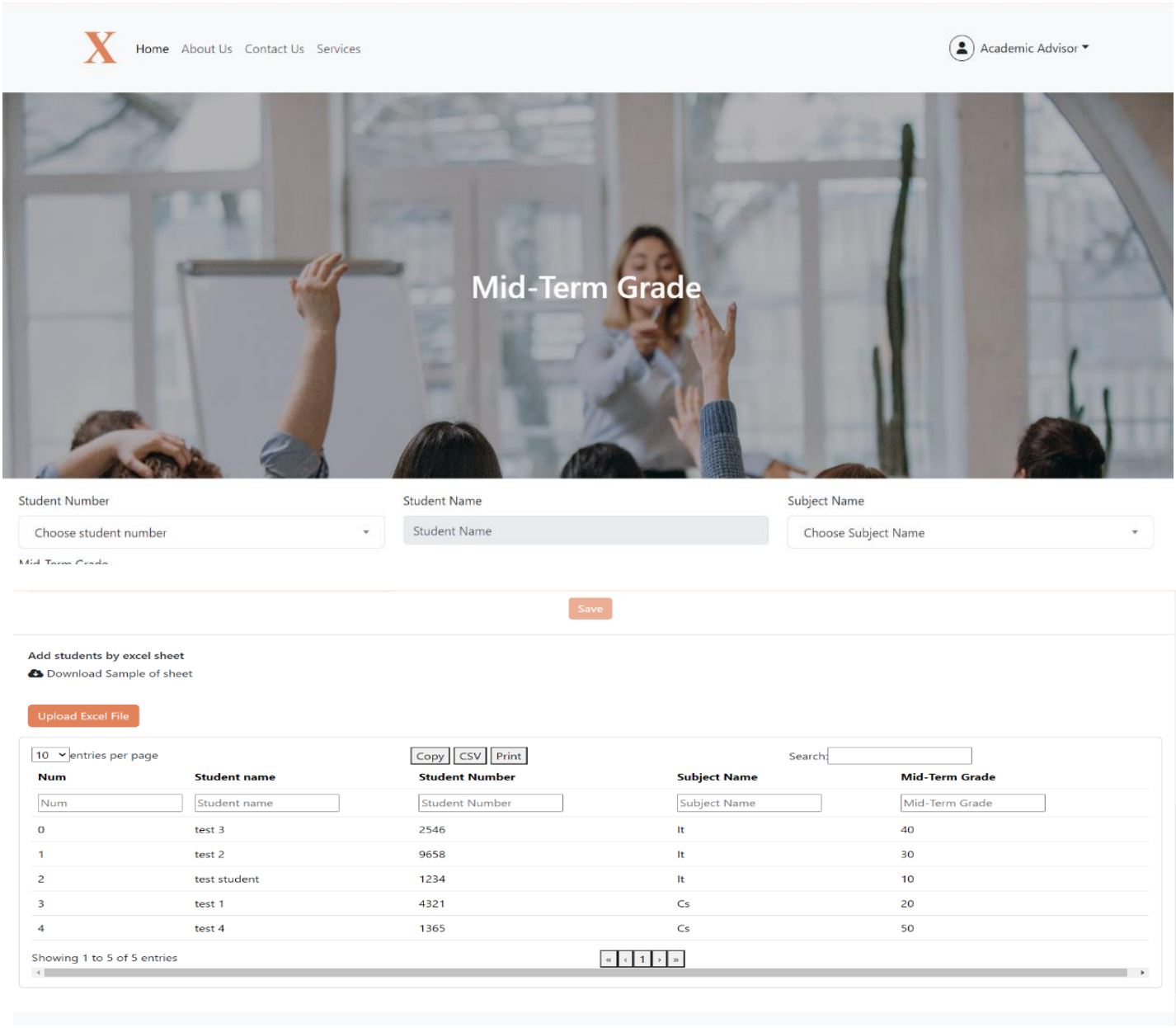
Num	Student name	Email	Phone	Student Number	State	Action
2	test 1	test1@gmail.com	01236547895	4321	Active	
3	test 2	test2@gmail.com	01236547895	9658	Active	
4	test 3	test3@gmail.com	01236547895	2546	Active	
5	test 4	asd@gmail.com	01236547895	1365		

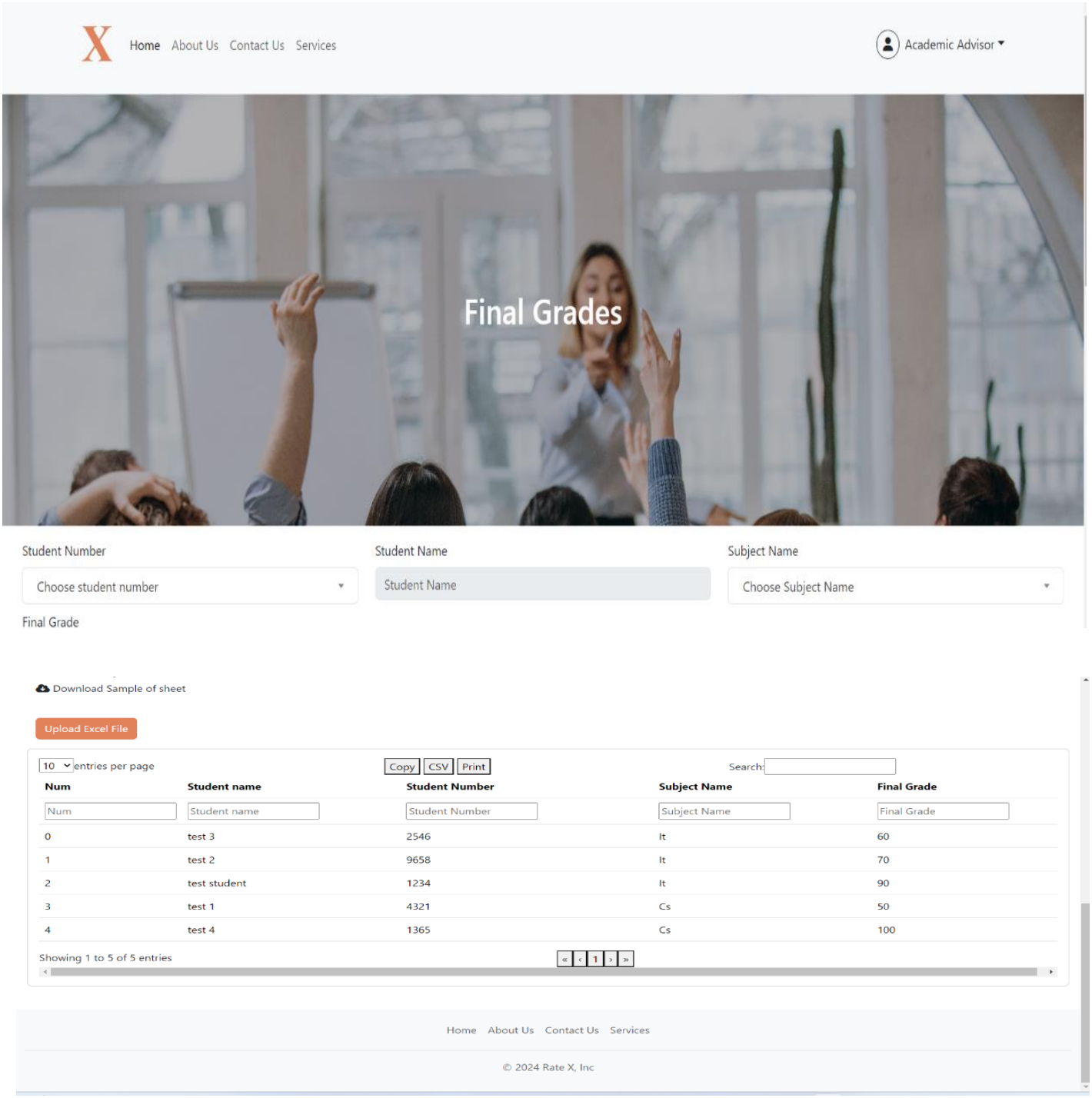
Showing 1 to 5 of 5 entries « ‹ 1 › »

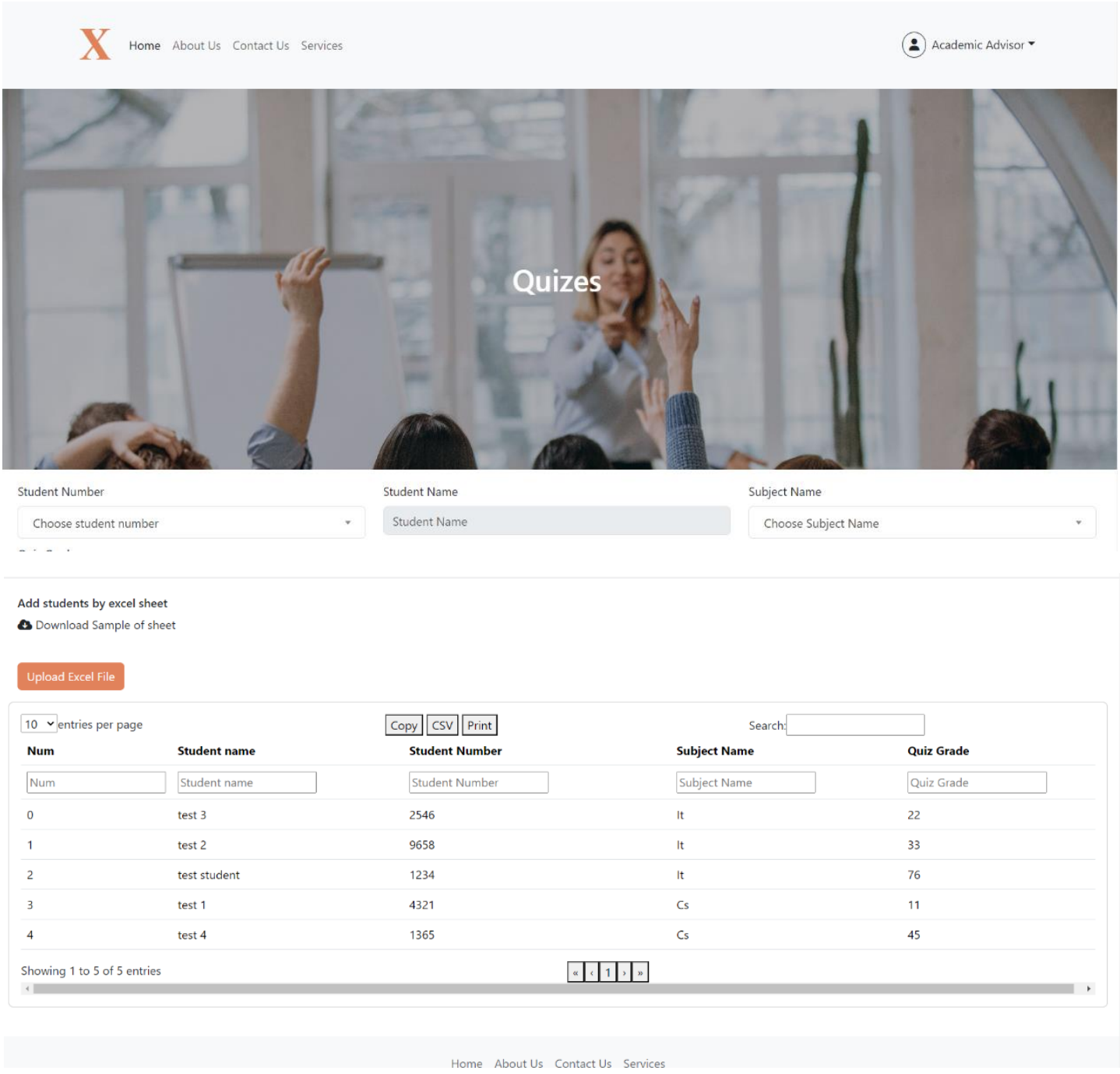


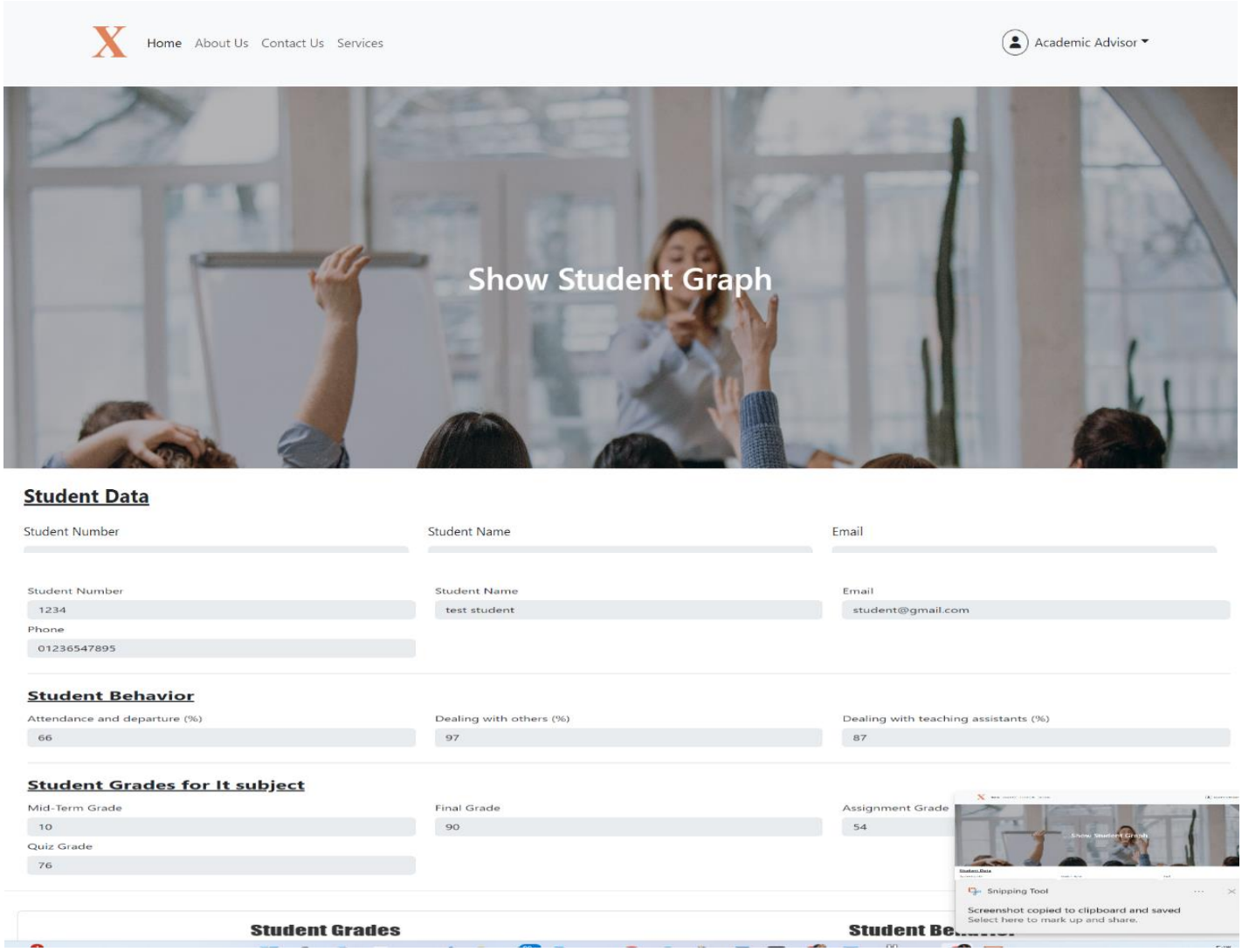
Home About Us Contact Us Services

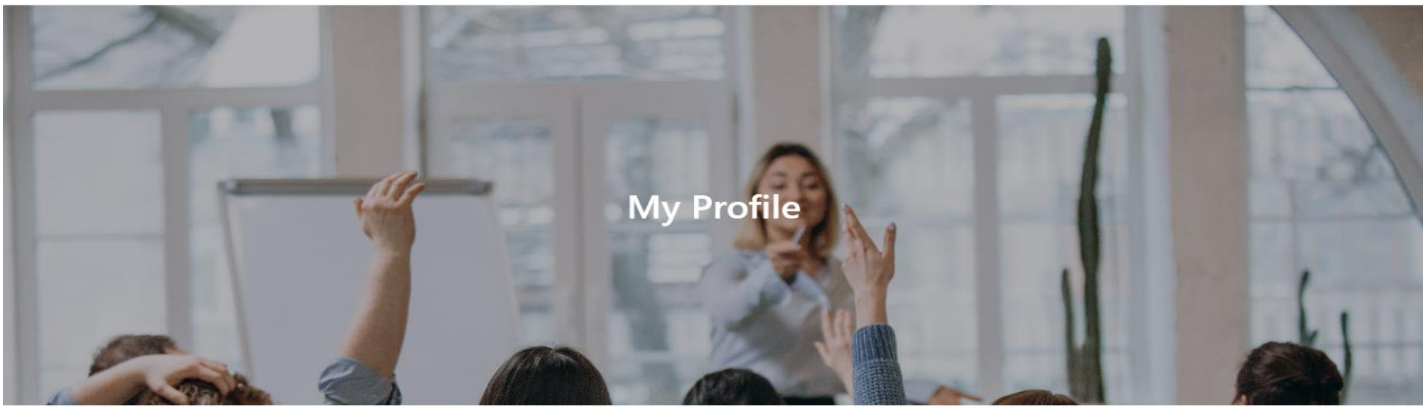
© 2024 Rate X, Inc


Home
About Us
Contact Us
Services
Academic Advisor
Mid-Term Grade
Student Number
Choose student number
Student Name
Student Name
Subject Name
Choose Subject Name
Save
Add students by excel sheet
Download Sample of sheet
Upload Excel File
10 entries per page
Copy
CSV
Print
Search:
Num
Student name
Student Number
Subject Name
Mid-Term Grade
0 test 3 2546 It 40
1 test 2 9658 It 30
2 test student 1234 It 10
3 test 1 4321 Cs 20
4 test 4 1365 Cs 50
Showing 1 to 5 of 5 entries


Home About Us Contact Us Services
Academic Advisor
Final Grades
Student Number
Choose student number
Student Name
Student Name
Subject Name
Choose Subject Name
Final Grade
Download Sample of sheet
Upload Excel File
10 entries per page
Copy CSV Print
Search:
Num
Student name
Student Number
Subject Name
Final Grade
0 test 3 2546 It 60
1 test 2 9658 It 70
2 test student 1234 It 90
3 test 1 4321 Cs 50
4 test 4 1365 Cs 100
Showing 1 to 5 of 5 entries
Home About Us Contact Us Services
© 2024 Rate X, Inc


Home About Us Contact Us Services
Academic Advisor
Quizes
Student Number
Choose student number
Student Name
Student Name
Subject Name
Choose Subject Name
Add students by excel sheet
Download Sample of sheet
Upload Excel File
10 entries per page
Copy CSV Print
Search:
Num
Student name
Student Number
Subject Name
Quiz Grade
0 test 3 2546 It 22
1 test 2 9658 It 33
2 test student 1234 It 76
3 test 1 4321 Cs 11
4 test 4 1365 Cs 45
Showing 1 to 5 of 5 entries
Home About Us Contact Us Services


Home
About Us
Contact Us
Services
Academic Advisor
Show Student Graph
Student Data
Student Number
Student Name
Email
Student Number
1234
Student Name
test student
Email
student@gmail.com
Phone
01236547895
Student Behavior
Attendance and departure (%)
66
Dealing with others (%)
97
Dealing with teaching assistants (%)
87
Student Grades for It subject
Mid-Term Grade
10
Final Grade
90
Assignment Grade
54
Quiz Grade
76
Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share.
Student Grades
Student Grades
Grades degree
Quiz Grade: 19%
Assignment Grade: 13.5%
Mid-Term Grade: 2.5%
Final Grade: 22.5%
CanvasJS.com
Student Behavior
Behavior degree
Dealing with teaching assistants: 2,900%
Dealing with others: 3,233.33333333%
Attendance and departure: 2,200%
CanvasJS.com
Tracks
Choose Track
Notification Message
Enter message to student
Send



My Data

Student Number: 1234

Student Name: test student

Email: student@gmail.com

Phone: 01236547895

My Behavior

Attendance and departure (%): 66

Dealing with others (%): 97

Dealing with teaching assistants (%): 87

My Grades

Student Grades for It subject

Mid-Term Grade: 10

Final Grade: 90

Assignment Grade: 54

Quiz Grade: 76

Methodology :

a series of repeated cycles or iterations. In this approach, a project or task is broken down into smaller, more manageable pieces, and each piece is completed through a series of iterations.

The iterative methodology is often used in software development, where it is also known as iterative and incremental development (IID) or agile development. In this approach, the software is developed through a series of iterations, with each iteration building on the previous one. Each iteration includes planning, design, implementation, testing, and evaluation, and feedback from each iteration is used to inform the next one .

The iterative methodology can also be used in other areas, such as product design, marketing, and project management. In these contexts, the iterative approach involves creating prototypes, testing them, and making modifications based on feedback until the final product meets the desired criteria .

The advantages of the iterative methodology include the ability to respond quickly to feedback, the ability to test and refine ideas before committing to a final product, and the ability to adapt to changing requirements or circumstances. However, it may also require more time and resources than other approaches, and it may not be suitable for all types of projects

Because Iterative methodology is a project management approach that involves breaking down a project into smaller, more manageable stages, and then continuously refining and improving the project in a cyclical manner. It is a popular approach in software development and other areas where the end product is complex and requires frequent adjustments and feedback

The iterative methodology involves the following steps :

- **Planning: In this stage, the project goals and scope are defined, and the project team creates a roadmap for the project .**
- **Requirements .**
- **Analysis**
- **Design: The project team creates a prototype or a draft version of the end product, which is then reviewed and refined based on feedback from stakeholders.**
- **Development: The project team develops the end product based on the refined prototype, and tests it to ensure it meets the project goals.**

• Testing: The end product is tested to identify any issues or bugs, and the project team works to address them.

• Evaluation: The project team evaluates the end product and the process used to develop it, and identifies areas for improvemen.

System architecture :

- **Web-Based System for Student-Project Allocation:**
  This system assists academic staff and students in allocating student projects. It uses a Constraint Programming Solver and follows a LAMP (Linux, Apache, MySQL, and PHP) architecture.
  Key features:
  Allocation: Helps assign projects to students.
  Life Cycle Management: Covers the entire project life cycle, from initial subject writing by teachers to student work evaluation.
- **Student Evaluation System:**
  Developed using PHP and MySQL, this system aims to eliminate manual errors by providing a computerized framework for student evaluation.
- **Component Diagram for Student Evaluation System:**
  A component diagram shows the components, interfaces, ports, and relationships in the system.

Why angular:

Modular Architecture:

Angular is modular architecture allows developers to divide the application into reusable and maintainable modules, which is ideal for complex systems like student evaluation platforms.

Two-Way Data Binding:

Angular is two-way data binding ensures that changes in the user interface automatically reflect in the application data and vice versa. This can simplify the development of dynamic and interactive features, such as real-time student performance updates.

Component-Based Structure:

Angular is component-based structure promotes the development of reusable UI components. This helps in creating a consistent and maintainable codebase, which is crucial for large applications.

Declarative UI:

Angular uses HTML to define the UI of the application, which makes it easier for developers to describe the appearance and behavior of the user interface.

Strong Typing with TypeScript:

Angular is built with TypeScript, which adds static typing to JavaScript. This helps in catching errors early in the development process and improves the overall code quality and maintainability.

Comprehensive Tooling:

Angular comes with a suite of tools like Angular CLI (Command Line Interface) that streamline the development process by providing commands for code generation, testing, and deployment.

Performance Optimization:

Angular provides features like Ahead-of-Time (AOT) compilation and tree shaking to optimize the performance of the application by reducing the size of the code and improving load times.

Why php:

Ease of Use:

PHP is relatively easy to learn and use, making it accessible for developers of varying skill levels. This can speed up the development process, especially for educational institutions or teams with limited resources.

Server-Side Scripting:

PHP is a powerful server-side scripting language, ideal for handling backend tasks such as processing form data, interacting with databases, and managing sessions. This makes it suitable for building dynamic and interactive student evaluation systems.

Database Integration:

PHP seamlessly integrates with various databases, such as MySQL, PostgreSQL, and SQLite. This capability is crucial for a student evaluation system that needs to store, retrieve, and manage large amounts of student data efficiently.

Extensive Frameworks and Libraries:

PHP has a rich ecosystem of frameworks (like Laravel, Symfony, and CodeIgniter) and libraries that can accelerate development, enforce best practices, and add advanced functionality. Frameworks like Laravel offer features such as routing, authentication, and ORM (Object-Relational Mapping), which can be particularly beneficial for building a comprehensive evaluation system.

Cost-Effective:

PHP is an open-source language, meaning there are no licensing fees associated with its use. This makes it a cost-effective option for educational institutions and developers working with limited budgets.

Large Community and Support:

PHP has a large and active community, providing extensive documentation, forums, and tutorials. This community support can be invaluable for troubleshooting, learning, and keeping up with best practices.

Implementation :

Footer html

```
<div class="container-fluid" style="background-color: #f8f9fa;">
    <footer class="py-3 mt-4">
      <ul class="nav justify-content-center border-bottom pb-3 mb-3">
        <li class="nav-item"><a href="home" class="nav-link px-2 text-body-
secondary">Home</a></li>
        <li class="nav-item"><a href="about-us" class="nav-link px-2 text-body-
secondary">About Us</a></li>
        <li class="nav-item"><a href="contact-us" class="nav-link px-2 text-body-
secondary">Contact Us</a></li>
        <li class="nav-item" *ngIf="userType != 'student'"><a href="services"
class="nav-link px-2 text-body-secondary">Services</a></li>
        <!-- <li class="nav-item"><a href="courese" class="nav-link px-2 text-
body-secondary">Courses</a></li> -->
      </ul>
      <p class="text-center text-body-secondary">© 2024 Rate X, Inc</p>
    </footer>
  </div>
```

Footer JS

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-footer',
  templateUrl: './footer.component.html',
  styleUrls: ['./footer.component.css']
})
export class FooterComponent implements OnInit {

  userType: any;
  ngOnInit(): void {
    this.userType = localStorage.getItem('userType')
  }
}
```

Header CSS

```
.img-logo-text{
    font-size: 60px;
    font-family: emoji;
    color: #e1825b;
    font-weight: 900;
}
.fa.fa-user{
    border: 1px solid;
    border-radius: 20px;
    padding: 8px;
}
```

Header HTML

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
    <div class="container">
      <a class="navbar-brand" href="home">
        <!-- <img src="../../../assets/img/logo.png"> -->
        <span class="img-logo-text">X</span>
      </a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          <li class="nav-item">
            <a class="nav-link active" href="home">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="about-us">About Us</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="contact-us">Contact Us</a>
          </li>
          <li class="nav-item" *ngIf="userType != 'student'">
            <a class="nav-link" href="services">Services</a>
          </li>
          <!-- <li class="nav-item">
            <a class="nav-link" href="courese">Courses</a>
```

```
          </li> -->
        </ul>
        <div class="d-flex" style="align-items: center;justify-content: center;">
            <li class="nav-item dropdown" style="list-style-type: none;">
                <a class="nav-link dropdown-toggle" href="#" role="button" data-
bs-toggle="dropdown" aria-expanded="false">
                    <i class="fa fa-user"></i>
                    <span class="ps-2">{{username}}</span>
                </a>
                <ul class="dropdown-menu">
                  <li><a href="profile/{{userId}}" class="dropdown-item"
*ngIf="showMyProfile">My Account</a></li>
                  <li><button class="dropdown-item" (click)="logout()">Log
Out</button></li>
                </ul>
            </li>
            <!-- <input class="form-control me-2" type="search"
placeholder="Search" aria-label="Search"> -->
          <!-- <button class="btn btn-outline-success"
type="submit">Search</button> -->
        </div>
      </div>
    </div>
  </nav>
```

Header JS

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit {

  username: any;
  userId: any;
  userType: any;
  showMyProfile: boolean = false;
  ngOnInit(): void {
      this.username = localStorage.getItem('name')
      this.userId = localStorage.getItem('userId')
      this.userType = localStorage.getItem('userType')
```

```
      if(this.userType === 'student'){
        this.showMyProfile = true
      }else{
        this.showMyProfile = false;
      }
  }

  logout(){
    localStorage.removeItem('name')
    localStorage.removeItem('userId')
    localStorage.removeItem('userType')
    window.location.href = 'login';
  }
}
```

About us CSS

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit {

  username: any;
  userId: any;
  userType: any;
  showMyProfile: boolean = false;
  ngOnInit(): void {
      this.username = localStorage.getItem('name')
      this.userId = localStorage.getItem('userId')
      this.userType = localStorage.getItem('userType')
      if(this.userType === 'student'){
        this.showMyProfile = true
      }else{
        this.showMyProfile = false;
      }
  }

  logout(){
    localStorage.removeItem('name')
```

```
    localStorage.removeItem('userId')
    localStorage.removeItem('userType')
    window.location.href = 'login';
  }
}
```

About us HTML

```
<app-header></app-header>

<section class="section_all bg-light" id="about">
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="section_title_all text-center">
                    <h3 class="font-weight-bold">Welcome To <span
class="logoTitle">Rate - X</span> <span class="text-custom"><br>Student
Evaluation System</span></h3>
                    <p class="section_subtitle mx-auto text-muted">Empowering
Student Success through Feedback and Assessment</p>
                </div>
            </div>
        </div>

        <div class="row vertical_content_manage mt-5">
            <div class="col-lg-6">
                <div class="about_header_main mt-3">
                    <div class="about_icon_box">
                        <p class="text_custom font-weight-bold">Enhance Student
Feedback</p>
                    </div>
                    <h4 class="about_heading text-capitalize font-weight-bold mt-
4">Providing Clear and Actionable Feedback</h4>
                    <p class="text-muted mt-3">Our Student Evaluation System
facilitates the delivery of comprehensive and constructive feedback to students,
aiding them in their academic development.</p>

                    <p class="text-muted mt-3">We prioritize the clarity and
effectiveness of feedback, empowering students to identify areas for improvement
and take proactive steps towards their academic goals.</p>
                </div>
            </div>
            <div class="col-lg-6">
                <div class="img_about mt-3">
```

```
                    <img src="https://i.ibb.co/qpz1hvM/About-us.jpg" alt="About
Us" class="img-fluid mx-auto d-block">
                </div>
            </div>
        </div>

        <div class="row mt-3">
            <div class="col-lg-4">
                <div class="about_content_box_all mt-3">
                    <div class="about_detail text-center">
                        <div class="about_icon">
                            <i class="fas fa-chart-line"></i>
                        </div>
                        <h5 class="text-dark text-capitalize mt-3 font-weight-
bold">Data-Driven Insights</h5>
                        <p class="edu_desc mt-3 mb-0 text-muted">Harness the
power of data analytics to gain valuable insights into student performance and
engagement.</p>
                    </div>
                </div>
            </div>

            <div class="col-lg-4">
                <div class="about_content_box_all mt-3">
                    <div class="about_detail text-center">
                        <div class="about_icon">
                            <i class="fas fa-comments"></i>
                        </div>
                        <h5 class="text-dark text-capitalize mt-3 font-weight-
bold">Two-Way Communication</h5>
                        <p class="edu_desc mb-0 mt-3 text-muted">Facilitate open
dialogue between educators and students, fostering a collaborative learning
environment.</p>
                    </div>
                </div>
            </div>

            <div class="col-lg-4">
                <div class="about_content_box_all mt-3">
                    <div class="about_detail text-center">
                        <div class="about_icon">
                            <i class="fas fa-clipboard-list"></i>
                        </div>
                        <h5 class="text-dark text-capitalize mt-3 font-weight-
bold">Streamlined Evaluation Process</h5>
```

```
                        <p class="edu_desc mb-0 mt-3 text-muted">Simplify and
streamline the evaluation process, saving time and resources for educators and
administrators.</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

<app-footer></app-footer>
```

About us JS

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-about-us',
  templateUrl: './about-us.component.html',
  styleUrls: ['./about-us.component.css']
})
export class AboutUsComponent {

}
```

Add student css

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
```

```
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
```

Add student HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">Add new Students</h1>
    <div class="overlay"></div>
</div>
<div class="row">
    <div class="col-12 text-center">
        <a href="students" class="btn btn-custom m-4">Show Students</a>
    </div>
    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <div class="row" *ngIf="insertedSuccessfully">
                    <div class="alert alert-success">{{msg}}</div>
                </div>
                <div class="row" *ngIf="error">
                    <div class="alert alert-danger">{{msg}}</div>
                </div>
                <form class="row" [formGroup]="stuForm">
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Student Name</label>
                            <input type="text" class="form-control"
placeholder="Enter name" formControlName="name" name="name">
                        </div>
                        <div *ngIf="f['name'].invalid && f['name'].touched"
style="color: red;font-size: 12px;">
                            <div *ngIf="f['name'].hasError('required')">Name is
required.</div>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
```

```
                        <div class="form-group">
                            <label class="form-label">Email</label>
                            <input type="text" class="form-control"
placeholder="Enter email" formControlName="email" name="email">
                        </div>
                        <div *ngIf="f['email'].invalid && f['email'].touched"
style="color: red;font-size: 12px;">
                            <div *ngIf="f['email'].hasError('required')">Email is
required.</div>
                            <div *ngIf="f['email'].hasError('email')">Invalid
email format.</div>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Phone</label>
                            <input type="text" class="form-control"
placeholder="Enter phone" pattern="^01[0-9]{9}$" formControlName="phone"
name="phone">
                        </div>
                        <div *ngIf="f['phone'].invalid && f['phone'].touched"
style="color: red;font-size: 12px;">
                            <div *ngIf="f['phone'].hasError('required')">Phone is
required.</div>
                            <div *ngIf="f['phone'].hasError('pattern')">Invalid
Phone format.</div>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Student Number</label>
                            <input type="text" class="form-control"
placeholder="Enter student number" formControlName="s_number" name="s_number">
                        </div>
                        <div *ngIf="f['s_number'].invalid &&
f['s_number'].touched" style="color: red;font-size: 12px;">
                            <div
*ngIf="f['s_number'].hasError('required')">Student Number is required.</div>
                            <div
*ngIf="f['s_number'].hasError('pattern')">Invalid Student Number format.</div>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Password</label>
```

```
                            <input type="password" class="form-control"
pattern="^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{6,}$" placeholder="Enter student
password" formControlName="password" name="password">
                        </div>
                        <div *ngIf="f['password'].invalid &&
f['password'].touched" style="color: red;font-size: 12px;">
                            <div
*ngIf="f['password'].hasError('required')">Password is required.</div>
                            <div
*ngIf="f['password'].hasError('pattern')">Invalid Password format.</div>
                        </div>
                    </div>
                    <div class="col-12 text-center">
                        <button class="btn btn-custom" (click)="save()"
[disabled]="stuForm.invalid">Save</button>
                    </div>

                    <hr class="mb-4 mt-4">
                </form>

                <form class="row">
                    <div class="col-12">
                        <h6>Add students by excel sheet</h6>
                        <!-- <p><a><i class="fa fa-cloud-download"></i> <span>
Download Sample of excel sheet</span></a></p> -->
                        <p>
                            <a style="cursor: pointer;"
(click)="downloadExcel()"><i class="fa fa-cloud-download"></i> Download Sample of
sheet</a>
                            <!-- <a (click)="downloadExcel()"
href="../../../assets/sample.xlsx"><i class="fa fa-cloud-download"></i> Download
Sample of sheet</a> -->
                        </p>
                    </div>
                    <div class="col-4 mt-4">
                        <label class="btn btn-custom" for="uploadExcel">Upload
Excel File</label>
                        <!-- <input type="file" hidden id="uploadExcel"
accept=".xlsx, application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"> -->
                        <input type="file" hidden id="uploadExcel"
(change)="onFileChange($event)" accept=".xlsx, application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet">
```

```
                        <!-- <input type="file" hidden id="uploadExcel"
accept=".xlsx, application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"> -->
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Add student JS

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ApiService } from 'src/app/api.service';
import * as XLSX from 'xlsx';

@Component({
  selector: 'app-add-student',
  templateUrl: './add-student.component.html',
  styleUrls: ['./add-student.component.css']
})
export class AddStudentComponent implements OnInit {

  stuForm: FormGroup;

  constructor(private api: ApiService,private fb: FormBuilder){
    this.stuForm = this.fb.group({
      name: ['',[Validators.required]],
      email: ['',[Validators.required,Validators.email]],
      phone: ['',[Validators.required]],
      s_number: ['', [Validators.required, Validators.pattern('^[0-9]{4,}$')]],
      password: ['',[Validators.required]],
    })
  }

  ngOnInit(): void {

  }

  get f(){
    return this.stuForm.controls
  }
```

```
  insertedSuccessfully: boolean = false;
  error: boolean = false;
  msg: any;
  save(){
    console.log(this.stuForm)
    if(this.stuForm.valid){
      this.api.insert_student(this.stuForm.value).subscribe({ next:(res: any) =>
{
          console.log(res)
          if(res['message'] == 'Student and user data inserted successfully.'){
            this.insertedSuccessfully = true;
            this.msg = 'Student and user data inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
          }else if(res['message'] == 'Email already exists.'){
            this.error = true;
            this.msg = 'Email already exists.';
            setTimeout(() => {
              this.error = false;
            }, 2000);
          }
      }})
    }
  }

  // download excel
  downloadExcel() {
    this.api.fetchStudents().subscribe((response: any) => {
      if (response.status === 'success') {
        this.api.exportToExcel2(response.data);
      } else {
        console.error('Failed to fetch students data');
      }
    }, error => {
      console.error('Error fetching students data', error);
    });
  }

  // import excel
  onFileChange(evt: any) {
```

```
  const target: DataTransfer = <DataTransfer>(evt.target);
  if (target.files.length !== 1) throw new Error('Cannot use multiple files');
  const reader: FileReader = new FileReader();
  reader.onload = (e: any) => {
    const bstr: string = e.target.result;
    const wb: XLSX.WorkBook = XLSX.read(bstr, { type: 'binary' });
    const wsname: string = wb.SheetNames[0];
    const ws: XLSX.WorkSheet = wb.Sheets[wsname];
    const data = <any[][]>(XLSX.utils.sheet_to_json(ws, { header: 1 }));

    // Process the data and send it to the backend
    const students = this.processData(data);
    this.uploadStudents(students);
  };
  reader.readAsBinaryString(target.files[0]);
}

processData(data: any[][]): any[] {
  const students = [];
  const phonePattern = /^01[0-9]{9}$/;

  for (let i = 1; i < data.length; i++) { // Assuming the first row is header
    const row = data[i];
    const student = {
      name: row[0],
      phone: row[1],
      email: row[2],
      s_number: row[3],
      password: row[4]
    };
console.log(row[i])
    // Validate the phone number
    if (!phonePattern.test(student.phone)) {
      console.warn(`Invalid phone number for student: ${student.name}`);
      continue; // Skip invalid phone numbers
    }

    students.push(student);
  }
  return students;
}

uploadStudents(students: any[]) {
  console.log(students)
  this.api.import_students({ students }).subscribe(
```

```
      (response: any) => {
        console.log(response)
        if(response['message'] == 'Students data processed successfully.'){
          this.insertedSuccessfully = true;
            this.msg = 'Student and user data inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
        }
      },
      error => console.log(error)
    );
  }
}
```

All services css

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ApiService } from 'src/app/api.service';
import * as XLSX from 'xlsx';

@Component({
  selector: 'app-add-student',
  templateUrl: './add-student.component.html',
  styleUrls: ['./add-student.component.css']
})
export class AddStudentComponent implements OnInit {

  stuForm: FormGroup;

  constructor(private api: ApiService,private fb: FormBuilder){
    this.stuForm = this.fb.group({
      name: ['',[Validators.required]],
      email: ['',[Validators.required,Validators.email]],
      phone: ['',[Validators.required]],
      s_number: ['', [Validators.required, Validators.pattern('^[0-9]{4,}$')]],
      password: ['',[Validators.required]],
    })
  }

  ngOnInit(): void {
```

```
  }

  get f(){
    return this.stuForm.controls
  }



  insertedSuccessfully: boolean = false;
  error: boolean = false;
  msg: any;
  save(){
    console.log(this.stuForm)
    if(this.stuForm.valid){
      this.api.insert_student(this.stuForm.value).subscribe({ next:(res: any) =>
{
          console.log(res)
          if(res['message'] == 'Student and user data inserted successfully.'){
            this.insertedSuccessfully = true;
            this.msg = 'Student and user data inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
          }else if(res['message'] == 'Email already exists.'){
            this.error = true;
            this.msg = 'Email already exists.';
            setTimeout(() => {
              this.error = false;
            }, 2000);
          }
      }})
    }
  }

  // download excel
  downloadExcel() {
    this.api.fetchStudents().subscribe((response: any) => {
      if (response.status === 'success') {
        this.api.exportToExcel2(response.data);
      } else {
        console.error('Failed to fetch students data');
      }
    }, error => {
      console.error('Error fetching students data', error);
```

```
  });
}

// import excel
onFileChange(evt: any) {
  const target: DataTransfer = <DataTransfer>(evt.target);
  if (target.files.length !== 1) throw new Error('Cannot use multiple files');
  const reader: FileReader = new FileReader();
  reader.onload = (e: any) => {
    const bstr: string = e.target.result;
    const wb: XLSX.WorkBook = XLSX.read(bstr, { type: 'binary' });
    const wsname: string = wb.SheetNames[0];
    const ws: XLSX.WorkSheet = wb.Sheets[wsname];
    const data = <any[][]>(XLSX.utils.sheet_to_json(ws, { header: 1 }));

    // Process the data and send it to the backend
    const students = this.processData(data);
    this.uploadStudents(students);
  };
  reader.readAsBinaryString(target.files[0]);
}

processData(data: any[][]): any[] {
  const students = [];
  const phonePattern = /^01[0-9]{9}$/;

  for (let i = 1; i < data.length; i++) { // Assuming the first row is header
    const row = data[i];
    const student = {
      name: row[0],
      phone: row[1],
      email: row[2],
      s_number: row[3],
      password: row[4]
    };
console.log(row[i])
    // Validate the phone number
    if (!phonePattern.test(student.phone)) {
      console.warn(`Invalid phone number for student: ${student.name}`);
      continue; // Skip invalid phone numbers
    }

    students.push(student);
  }
  return students;
```

```
  }

  uploadStudents(students: any[]) {
    console.log(students)
    this.api.import_students({ students }).subscribe(
      (response: any) => {
        console.log(response)
        if(response['message'] == 'Students data processed successfully.'){
          this.insertedSuccessfully = true;
            this.msg = 'Student and user data inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
        }
      },
      error => console.log(error)
    );
  }
}
```

All services html

```
<app-header></app-header>

    <section class="we-offer-area text-center bg-gray">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <div class="site-heading text-center">
                        <h2>What we <span>Offer</span></h2>
                        <h4><span>Rate - X</span> Student Evaluation System</h4>
                    </div>
                </div>
            </div>
            <div class="row our-offer-items less-carousel">

                <div class="col-md-4 col-sm-6 equal-height">
                    <a href="students">
                        <div class="item">
                            <i class="fas fa-users"></i>
                            <h4>All Students</h4>
                            <p>
```

```
                            Manage and view information for all enrolled
students in one centralized platform.
                        </p>
                    </div>
                </a>
            </div>

            <div class="col-md-4 col-sm-6 equal-height">
                <a href="add-students">
                    <div class="item">
                        <i class="fas fa-user-plus"></i>
                        <h4>Add Students</h4>
                        <p>
                            Seamlessly add new students to the system with
our user-friendly interface.
                        </p>
                    </div>
                </a>
            </div>

            <div class="col-md-4 col-sm-6 equal-height" *ngIf="userType !=
'teacher' && userType != 'advisor'">
                <a href="mid-grade">
                    <div class="item">
                        <i class="fas fa-eraser"></i>
                        <h4>Mid Grades</h4>
                        <p>
                            Evaluate and record mid-term grades with ease
using our comprehensive grading system.
                        </p>
                    </div>
                </a>
            </div>

            <div class="col-md-4 col-sm-6 equal-height" *ngIf="userType !=
'teacher' && userType != 'advisor'">
                <a href="final-grade">
                    <div class="item">
                        <i class="fas fa-tasks"></i>
                        <h4>Final Grades</h4>
                        <p>
                            Efficiently calculate and assign final grades for
courses with our intuitive grading tool.
                        </p>
                    </div>
```

```
                    </a>
                </div>

                <div class="col-md-4 col-sm-6 equal-height" *ngIf="userType !=
'doctor' && userType != 'advisor'">
                    <a href="quizes">
                        <div class="item">
                            <i class="fas fa-file-pen"></i>
                            <h4>Quizes</h4>
                            <p>
                                Create and manage quizzes effortlessly with our
quiz management feature.
                            </p>
                        </div>
                    </a>
                </div>

                <div class="col-md-4 col-sm-6 equal-height" *ngIf="userType !=
'doctor' && userType != 'advisor'">
                    <a href="assignment">
                        <div class="item">
                            <i class="fas fa-folder-open"></i>
                            <h4>Assignments</h4>
                            <p>
                                Assign and track student assignments seamlessly
through our assignment tracking system.
                            </p>
                        </div>
                    </a>
                </div>

                <div class="col-md-4 col-sm-6 equal-height" *ngIf="userType !=
'doctor' && userType != 'advisor'">
                    <a href="student-evaluations">
                        <div class="item">
                            <i class="fas fa-recycle"></i>
                            <h4>Student evaluations</h4>
                            <p>
                                Conduct comprehensive student evaluations to
assess learning outcomes and measure student proficiency.
                            </p>
                        </div>
                    </a>
                </div>
```

```
            </div>
        </div>
    </section>

<app-footer></app-footer>
```

All services angular

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-all-services',
  templateUrl: './all-services.component.html',
  styleUrls: ['./all-services.component.css']
})
export class AllServicesComponent implements OnInit {

  userType: any;
  ngOnInit(): void {
    this.userType = localStorage.getItem('userType')
  }
}
```

Assignment css

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
```

```
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
```

Assignment HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">Assignment</h1>
    <div class="overlay"></div>
</div>
<div class="row m-3" *ngIf="insertedSuccessfully">
    <div class="alert alert-success">{{msg}}</div>
</div>
<div class="row m-3" *ngIf="error">
    <div class="alert alert-danger">{{msg}}</div>
</div>
<div class="row">
    <form class="col-12" [formGroup]="stuForm">
        <div class="row m-3">
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Number</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose student number" (valueChanged)="changeStudent($event)"
formControlName="s_id">
                        <option selected disabled value="0">Choose student
number</option>
                        <option *ngFor="let num of data"
[value]="num.id">{{num.s_number}}</option>
                    </ng-select2>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Name</label>
                    <input type="text" class="form-control"
value="{{studentName}}" placeholder="Student Name" readonly disabled>
```

```
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Subject Name</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose Subject Name" (valueChanged)="changeTrack($event)"
formControlName="track_id">
                        <option selected disabled value="0">Choose Subject
Name</option>
                        <option *ngFor="let track of tracks"
[value]="track.id">{{track.name}}</option>
                    </ng-select2>
                    <!-- <input type="text" class="form-control"
placeholder="Enter subject name"> -->
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Assignment Grade</label>
                    <input type="text" class="form-control" placeholder="Enter
grade" name="assignment" formControlName="assignment">
                </div>
            </div>
            <div class="col-12 text-center">
                <button class="btn btn-custom" [disabled]="stuForm.invalid"
(click)="save()">Save</button>
            </div>
        </div>

        <hr class="mb-4 mt-4">

        <div class="row mx-3">
            <div class="col-12">
                <h6>Add students by excel sheet</h6>
                <p>
                    <a style="cursor: pointer;" (click)="downloadExcel()"><i
class="fa fa-cloud-download"></i> Download Sample of sheet</a>
                </p>
            </div>
            <div class="col-4 mt-4">
                <label class="btn btn-custom" for="uploadExcel">Upload Excel
File</label>
```

```
                <input type="file" hidden id="uploadExcel" accept=".xlsx,
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
(change)="onFileChange($event)">
            </div>
        </div>
    </form>


    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <div class="table-responsive">
                    <table class="table w-100 bordered" id="example">
                        <thead>
                            <tr>
                                <th>Num</th>
                                <th>Student name</th>
                                <th>Student Number</th>
                                <th>Subject Name</th>
                                <th>Assignment Grade</th>
                                <!-- <th>Action</th> -->
                            </tr>
                        </thead>
                        <tbody>
                            <tr *ngFor="let stu of studentsGrades;let i = index">
                                <td>{{i}}</td>
                                <td>{{stu.student_name}}</td>
                                <td>{{stu.student_number}}</td>
                                <td>{{stu.track_name}}</td>
                                <td>{{stu.assignment}}</td>
                                <!-- <td>
                                    <button class="btn btn-danger"><i class="fa
fa-trash"></i></button>
                                </td> -->
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Assignment Angular

```
import { Component, OnInit } from '@angular/core';
import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { Select2OptionData } from 'ng-select2';
import { Options } from 'select2';
import { ApiService } from 'src/app/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import * as saveAs from 'file-saver';
import * as XLSX from 'xlsx';

@Component({
  selector: 'app-assignment',
  templateUrl: './assignment.component.html',
  styleUrls: ['./assignment.component.css']
})
export class AssignmentComponent implements OnInit {

  public options!: Options;
  data: any;
  tracks: any;
  stuForm: FormGroup;
  stuExcelForm: FormGroup;
  studentsGrades: any;
  gradeType!: string;

  constructor(private api: ApiService,private fb: FormBuilder){
    this.stuForm = this.fb.group({
      s_id:['0',[Validators.required]],
      track_id:['0',[Validators.required]],
      assignment:['',[Validators.required]]
    })

    this.stuExcelForm = this.fb.group({
      s_id:[''],
      track_id:[''],
      assignment:['']
    })
```

```
    this.api.get_student('').subscribe({ next: (res: any) => {
      this.data = res.data;
      // console.log(this.data)
    }})

    this.api.get_tracks().subscribe({ next: (res: any) => {
      this.tracks = res.data;
      // console.log(this.tracks)
    }})

    this.api.get_all_grads().subscribe({next: (res: any) => {
      this.studentsGrades = res.data;
      console.log(this.studentsGrades)
    }})
  }

  ngOnInit(): void {
    setTimeout(()=>{
      $('#example thead tr').clone(true).addClass('filters').appendTo('#example
thead');

      var table=$('#example').DataTable({
        "orderCellsTop": true,
        "pagingType": "full_numbers",
        scrollCollapse: true,
        "search": true,
        "lengthMenu": [[10, 25, 50,100, -1], [10, 25, 50,100, "All"]],
        "dom":  "<'row'<'col-md-4'l><'col-md-4'B><'col-md-4'f>><'row'<'col-md-
12't>><'row'<'col-md-6'i><'col-md-6'p>>",
        initComplete: function (api:any) {

       api.aoColumns.forEach(function(aoColumn:any) {

      var cell = $('.filters th').eq(
        aoColumn.idx
    );
    var title = $(cell).text();
    $(cell).html('<input type="text" placeholder="' + title + '"  class="' +
title + '" />');
     if(title == "Phone"){
      $(cell).html('<input type="number" placeholder="Phone" pattern="[0-9]{11}"
maxlength="11" onkeypress="if(this.value.length==11) return false; return /[0-
9]/i.test(event.key)"' + title + '" />');
     }
     if(title == "Action"){
```

```
      $(cell).html('<input type="number" style="display:none;"');
     }
    $('input',cell).off('keyup change').on('keyup change', function (e:any) {
                    e.stopPropagation();

            var regexr = '({search})';
$(this).parents('th').find('select').val();
            var cursorPosition = e.selectionStart;

                  var code = e.keyCode || e.which;
                  if (code == 13) {
                    console.log($(this),aoColumn);

                    table .column( aoColumn.idx )
                        .search(
                          $(this).val() != ''
                                      ? regexr.replace('{search}', '((' +
$(this).val() + '))')
                                      : '',
                                      $(this).val() != '',
                                      $(this).val() == ''
                              )
                              .draw();
                          }
                  });
              });
            },
        });
    },200);

    this.options = {
      multiple: false,
      closeOnSelect: true,
      width: '100%'
    };

    this.gradeType = 'assignment';
  }

  studentName: any;
  changeStudent(e: any){
    this.api.get_student(e).subscribe({next: (res: any) => {
      console.log(res.data)
      this.studentName = res.data.name
    }})
```

```
  }

  changeTrack(e: any){}

  insertedSuccessfully: boolean = false;
  error: boolean = false;
  msg: any;
  save(){
    console.log(this.stuForm)
    if(this.stuForm){
      this.api.insert_grades(this.stuForm.value).subscribe({next: (res: any) => {
        if(res['message'] == 'Grade inserted successfully.'){
          this.insertedSuccessfully = true;
          this.msg = 'Grade inserted successfully.';
          setTimeout(() => {
            this.insertedSuccessfully = false;
            window.location.reload();
          }, 2000);
        }else if(res['message'] == 'Failed to insert grade.'){
          this.error = true;
          this.msg = 'Failed to insert grade.';
          setTimeout(() => {
            this.error = false;
          }, 2000);
        }
      }})
    }
  }

  // download excel
  downloadExcel() {
    this.api.fetchStudentsWithSpecificGrade(this.gradeType).subscribe((response:
any) => {
      if (response.status === 'success') {
        this.exportToExcel(response.data);
      } else {
        console.error('Failed to fetch students data');
      }
    }, error => {
      console.error('Error fetching students data', error);
    });
  }

  exportToExcel(students: any[]) {
    // Ensure only the relevant grade field is included
```

```
    const updatedStudents = students.map(student => ({
      ...student,
      [this.gradeType]: student[this.gradeType] || ''
    }));

    const ws: XLSX.WorkSheet = XLSX.utils.json_to_sheet(updatedStudents);
    const wb: XLSX.WorkBook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(wb, ws, 'Students');
    const wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'array' });
    saveAs(new Blob([wbout], { type: 'application/octet-stream' }),
`students_${this.gradeType}.xlsx`);
  }
  onFileChange(evt: any) {
    const target: DataTransfer = <DataTransfer>(evt.target);
    if (target.files.length !== 1) throw new Error('Cannot use multiple files');
    const reader: FileReader = new FileReader();
    reader.onload = (e: any) => {
      const bstr: string = e.target.result;
      const wb: XLSX.WorkBook = XLSX.read(bstr, { type: 'binary' });
      const wsname: string = wb.SheetNames[0];
      const ws: XLSX.WorkSheet = wb.Sheets[wsname];
      const data = <any[][]>(XLSX.utils.sheet_to_json(ws, { header: 1 }));

      // Process the data and send it to the backend
      const students = this.processData(data);
      this.uploadStudents(students);
    };
    reader.readAsBinaryString(target.files[0]);
  }

  processData(data: any[][]): any[] {
    const students = [];

    for (let i = 1; i < data.length; i++) { // Assuming the first row is header
      const row = data[i];
      const student = {
        student_id: row[0],
        student_name: row[1],
        student_number: row[2],
        student_email: row[3],
        subject_id: row[4],
        subject_name: row[5],
        assignment: row[6]
      };
```

```
      students.push(student);
    }
    return students;
  }

  uploadStudents(students: any[]) {
    for (let i = 0; i < students.length; i++) {
      const student = students[i];
      const formData = {
        s_id: student.student_id,
        track_id: student.subject_id,
        assignment: student.assignment
      };

      // Send the individual student data to the API
      this.api.insert_grades(formData).subscribe({
        next: (res: any) => {
          console.log(res)
          if (res['message'] == 'Grade inserted successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle success if needed
            // console.log('Grade inserted successfully for student:',
student.student_name);

          }else if (res['message'] == 'Grade updated successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade updated successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }else if (res['message'] == 'Failed to insert grade.') {
            this.error = true;
            this.msg = 'Failed to insert grade.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
```

```
            }, 2000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }
        },
        error: (err) => {
          // Handle error if needed
          console.error('Error occurred while inserting grade for student:',
student.student_name, err);
        }
      });
    }
  }
}
```

Contact us CSS

```
body{
    color:#fff
  }
  .right_conatct_social_icon{
    background: linear-gradient(to top right, #e1825b -5%, #eeb39b 100%);
  }
  .contact_us{
      background-color: #f1f1f1;
      padding: 120px 0px;
  }

  .contact_inner{
      background-color: #fff;
      position: relative;
      box-shadow: 20px 22px 44px #cccc;
      border-radius: 25px;
  }
  .contact_field{
      padding: 60px 340px 90px 100px;
  }
  .right_conatct_social_icon{
      height: 100%;
  }

  .contact_field h3{
     color: #000;
```

```
    font-size: 40px;
    letter-spacing: 1px;
    font-weight: 600;
    margin-bottom: 10px
}
.contact_field p{
    color: #000;
    font-size: 13px;
    font-weight: 400;
    letter-spacing: 1px;
    margin-bottom: 35px;
}
.contact_field .form-control{
    border-radius: 0px;
    border: none;
    border-bottom: 1px solid #ccc;
}
.contact_field .form-control:focus{
    box-shadow: none;
    outline: none;
    border-bottom: 2px solid #1325e8;
}
.contact_field .form-control::placeholder{
    font-size: 13px;
    letter-spacing: 1px;
}

.contact_info_sec {
    position: absolute;
    background-color: #2d2d2d;
    right: 1px;
    top: 18%;
    height: 340px;
    width: 340px;
    padding: 40px;
    border-radius: 25px 0 0 25px;
    color: #fff;
}
.contact_info_sec h4{
    letter-spacing: 1px;
    padding-bottom: 15px;
}

.info_single{
    margin: 30px 0px;
```

```
}
.info_single i{
    margin-right: 15px;
}
.info_single span{
    font-size: 14px;
    letter-spacing: 1px;
}

button.contact_form_submit {
    background: linear-gradient(to top right, #e1825b -5%, #eeb39b 100%);
    border: none;
    color: #fff;
    padding: 10px 15px;
    width: 100%;
    margin-top: 25px;
    border-radius: 35px;
    cursor: pointer;
    font-size: 14px;
    letter-spacing: 2px;
}
.socil_item_inner li{
    list-style: none;
}
.socil_item_inner li a{
    color: #fff;
    margin: 0px 15px;
    font-size: 14px;
}
.socil_item_inner{
    padding-bottom: 10px;
}

.map_sec{
    padding: 50px 0px;
}
.map_inner h4, .map_inner p{
    color: #000;
    text-align: center
}
.map_inner p{
    font-size: 13px;
}
.map_bind{
   margin-top: 50px;
```

```
    border-radius: 30px;
    overflow: hidden;
  }
```

Contact us HTML

```
<app-header></app-header>

<section class="contact_us">
    <div class="container">
        <div class="row">
            <div class="col-md-10 offset-md-1">
                <div class="contact_inner">
                    <div class="row">
                        <div class="col-md-10">
                            <div class="contact_form_inner">
                                <div class="contact_field">
                                    <h3>Contatc Us</h3>
                                    <p>Feel Free to contact us any time. We will
get back to you as soon as we can!</p>
                                    <form>
                                        <input type="text" class="form-control
form-group" placeholder="Name" />
                                        <input type="text" class="form-control
form-group" placeholder="Email" />
                                        <textarea class="form-control form-group"
placeholder="Message"></textarea>
                                        <button
class="contact_form_submit">Send</button>
                                    </form>
                                </div>
                            </div>
                        </div>
                        <div class="col-md-2">
                            <div class="right_conatct_social_icon d-flex align-
items-end">
                                <div class="socil_item_inner d-flex">
                                    <li><a href="#"><i class="fab fa-facebook-
square"></i></a></li>
                                    <li><a href="#"><i class="fab fa-
instagram"></i></a></li>
                                    <li><a href="#"><i class="fab fa-
twitter"></i></a></li>
                                </div>
```

```
                        </div>
                    </div>
                </div>
                <div class="contact_info_sec">
                    <h4>Contact Info</h4>
                    <div class="d-flex info_single align-items-center">
                        <i class="fas fa-headset"></i>
                        <span>+20 12345678950</span>
                    </div>
                    <div class="d-flex info_single align-items-center">
                        <i class="fas fa-envelope-open-text"></i>
                        <span>info@ratex.com</span>
                    </div>
                    <div class="d-flex info_single align-items-center">
                        <i class="fas fa-map-marked-alt"></i>
                        <span>1000+ Student Asyut University, Asyut,
Egypt</span>
                    </div>

                </div>
            </div>
        </div>
    </div>
</section>

<app-footer></app-footer>
```

Contact us angular

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-contact-us',
  templateUrl: './contact-us.component.html',
  styleUrls: ['./contact-us.component.css']
})
export class ContactUsComponent {

}
```

Final grades CSS

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
}
```

Final grades HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">Final Grades</h1>
    <div class="overlay"></div>
</div>
<div class="row m-3" *ngIf="insertedSuccessfully">
    <div class="alert alert-success">{{msg}}</div>
</div>
<div class="row m-3" *ngIf="error">
    <div class="alert alert-danger">{{msg}}</div>
</div>
<div class="row">
    <form class="col-12" [formGroup]="stuForm">
```

```
        <div class="row m-3">
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Number</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose student number" (valueChanged)="changeStudent($event)"
formControlName="s_id">
                        <option selected disabled value="0">Choose student
number</option>
                        <option *ngFor="let num of data"
[value]="num.id">{{num.s_number}}</option>
                    </ng-select2>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Name</label>
                    <input type="text" class="form-control"
value="{{studentName}}" placeholder="Student Name" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Subject Name</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose Subject Name" (valueChanged)="changeTrack($event)"
formControlName="track_id">
                        <option selected disabled value="0">Choose Subject
Name</option>
                        <option *ngFor="let track of tracks"
[value]="track.id">{{track.name}}</option>
                    </ng-select2>
                    <!-- <input type="text" class="form-control"
placeholder="Enter subject name"> -->
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Final Grade</label>
                    <input type="text" class="form-control" placeholder="Enter
grade" name="final" formControlName="final">
                </div>
            </div>
            <div class="col-12 text-center">
```

```
                <button class="btn btn-custom" [disabled]="stuForm.invalid"
(click)="save()">Save</button>
            </div>
        </div>

        <hr class="mb-4 mt-4">

        <div class="row mx-3">
            <div class="col-12">
                <h6>Add students by excel sheet</h6>
                <p>
                    <a style="cursor: pointer;" (click)="downloadExcel()"><i
class="fa fa-cloud-download"></i> Download Sample of sheet</a>
                </p>
            </div>
            <div class="col-4 mt-4">
                <label class="btn btn-custom" for="uploadExcel">Upload Excel
File</label>
                <input type="file" hidden id="uploadExcel" accept=".xlsx,
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
(change)="onFileChange($event)">
            </div>
        </div>
    </form>


    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <div class="table-responsive">
                    <table class="table w-100 bordered" id="example">
                        <thead>
                            <tr>
                                <th>Num</th>
                                <th>Student name</th>
                                <th>Student Number</th>
                                <th>Subject Name</th>
                                <th>Final Grade</th>
                                <!-- <th>Action</th> -->
                            </tr>
                        </thead>
                        <tbody>
                            <tr *ngFor="let stu of studentsGrades;let i = index">
                                <td>{{i}}</td>
                                <td>{{stu.student_name}}</td>
```

```
                                <td>{{stu.student_number}}</td>
                                <td>{{stu.track_name}}</td>
                                <td>{{stu.final}}</td>
                                <!-- <td>
                                    <button class="btn btn-danger"><i class="fa
fa-trash"></i></button>
                                </td> -->
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Final grade angular

```
import { Component, OnInit } from '@angular/core';
import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { Select2OptionData } from 'ng-select2';
import { Options } from 'select2';
import { ApiService } from 'src/app/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import * as saveAs from 'file-saver';
import * as XLSX from 'xlsx';
@Component({
  selector: 'app-final-grades',
  templateUrl: './final-grades.component.html',
  styleUrls: ['./final-grades.component.css']
})
export class FinalGradesComponent implements OnInit {

  public options!: Options;
  data: any;
  tracks: any;
  stuForm: FormGroup;
```

```
  stuExcelForm: FormGroup;
  studentsGrades: any;
  gradeType!: string;

  constructor(private api: ApiService,private fb: FormBuilder){
    this.stuForm = this.fb.group({
      s_id:['0',[Validators.required]],
      track_id:['0',[Validators.required]],
      final:['',[Validators.required]]
    })

    this.stuExcelForm = this.fb.group({
      s_id:[''],
      track_id:[''],
      final:['']
    })

    this.api.get_student('').subscribe({ next: (res: any) => {
      this.data = res.data;
      // console.log(this.data)
    }})

    this.api.get_tracks().subscribe({ next: (res: any) => {
      this.tracks = res.data;
      // console.log(this.tracks)
    }})

    this.api.get_all_grads().subscribe({next: (res: any) => {
      this.studentsGrades = res.data;
      console.log(this.studentsGrades)
    }})
  }

  ngOnInit(): void {
    setTimeout(()=>{
      $('#example thead tr').clone(true).addClass('filters').appendTo('#example
thead');

      var table=$('#example').DataTable({
        "orderCellsTop": true,
        "pagingType": "full_numbers",
        scrollCollapse: true,
        "search": true,
        "lengthMenu": [[10, 25, 50,100, -1], [10, 25, 50,100, "All"]],
```

```
        "dom":  "<'row'<'col-md-4'l><'col-md-4'B><'col-md-4'f>><'row'<'col-md-
12't>><'row'<'col-md-6'i><'col-md-6'p>>",
        initComplete: function (api:any) {

       api.aoColumns.forEach(function(aoColumn:any) {

      var cell = $('.filters th').eq(
        aoColumn.idx
    );
    var title = $(cell).text();
    $(cell).html('<input type="text" placeholder="' + title + '"  class="' +
title + '" />');
     if(title == "Phone"){
      $(cell).html('<input type="number" placeholder="Phone" pattern="[0-9]{11}"
maxlength="11" onkeypress="if(this.value.length==11) return false; return /[0-
9]/i.test(event.key)"' + title + '" />');
     }
     if(title == "Action"){
      $(cell).html('<input type="number" style="display:none;"');
     }
    $('input',cell).off('keyup change').on('keyup change', function (e:any) {
                    e.stopPropagation();

            var regexr = '({search})';
$(this).parents('th').find('select').val();
            var cursorPosition = e.selectionStart;

                  var code = e.keyCode || e.which;
                  if (code == 13) {
                    console.log($(this),aoColumn);

                    table .column( aoColumn.idx )
                        .search(
                          $(this).val() != ''
                                      ? regexr.replace('{search}', '((' +
$(this).val() + '))')
                                      : '',
                                      $(this).val() != '',
                                      $(this).val() == ''
                              )
                              .draw();
                          }
                  });
              });
            },
```

```
      });
  },200);

  this.options = {
    multiple: false,
    closeOnSelect: true,
    width: '100%'
  };

  this.gradeType = 'final';
}

studentName: any;
changeStudent(e: any){
  this.api.get_student(e).subscribe({next: (res: any) => {
    console.log(res.data)
    this.studentName = res.data.name
  }})
}

changeTrack(e: any){}

insertedSuccessfully: boolean = false;
error: boolean = false;
msg: any;
save(){
  console.log(this.stuForm)
  if(this.stuForm){
    this.api.insert_grades(this.stuForm.value).subscribe({next: (res: any) => {
      if(res['message'] == 'Grade inserted successfully.'){
        this.insertedSuccessfully = true;
        this.msg = 'Grade inserted successfully.';
        setTimeout(() => {
          this.insertedSuccessfully = false;
          window.location.reload();
        }, 2000);
      }else if(res['message'] == 'Failed to insert grade.'){
        this.error = true;
        this.msg = 'Failed to insert grade.';
        setTimeout(() => {
          this.error = false;
        }, 2000);
      }
    }})
  }
```

```
  }

  // download excel
  downloadExcel() {
    this.api.fetchStudentsWithSpecificGrade(this.gradeType).subscribe((response:
any) => {
      if (response.status === 'success') {
        this.exportToExcel(response.data);
      } else {
        console.error('Failed to fetch students data');
      }
    }, error => {
      console.error('Error fetching students data', error);
    });
  }

  exportToExcel(students: any[]) {
    // Ensure only the relevant grade field is included
    const updatedStudents = students.map(student => ({
      ...student,
      [this.gradeType]: student[this.gradeType] || ''
    }));

    const ws: XLSX.WorkSheet = XLSX.utils.json_to_sheet(updatedStudents);
    const wb: XLSX.WorkBook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(wb, ws, 'Students');
    const wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'array' });
    saveAs(new Blob([wbout], { type: 'application/octet-stream' }),
`students_${this.gradeType}.xlsx`);
  }
  onFileChange(evt: any) {
    const target: DataTransfer = <DataTransfer>(evt.target);
    if (target.files.length !== 1) throw new Error('Cannot use multiple files');
    const reader: FileReader = new FileReader();
    reader.onload = (e: any) => {
      const bstr: string = e.target.result;
      const wb: XLSX.WorkBook = XLSX.read(bstr, { type: 'binary' });
      const wsname: string = wb.SheetNames[0];
      const ws: XLSX.WorkSheet = wb.Sheets[wsname];
      const data = <any[][]>(XLSX.utils.sheet_to_json(ws, { header: 1 }));

      // Process the data and send it to the backend
      const students = this.processData(data);
      this.uploadStudents(students);
    };
```

```
    reader.readAsBinaryString(target.files[0]);
  }

  processData(data: any[][]): any[] {
    const students = [];

    for (let i = 1; i < data.length; i++) { // Assuming the first row is header
      const row = data[i];
      const student = {
        student_id: row[0],
        student_name: row[1],
        student_number: row[2],
        student_email: row[3],
        subject_id: row[4],
        subject_name: row[5],
        final: row[6]
      };

      students.push(student);
    }
    return students;
  }

  uploadStudents(students: any[]) {
    for (let i = 0; i < students.length; i++) {
      const student = students[i];
      const formData = {
        s_id: student.student_id,
        track_id: student.subject_id,
        final: student.final
      };

      // Send the individual student data to the API
      this.api.insert_grades(formData).subscribe({
        next: (res: any) => {
          console.log(res)
          if (res['message'] == 'Grade inserted successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle success if needed
```

```
            // console.log('Grade inserted successfully for student:',
student.student_name);

          }else if (res['message'] == 'Grade updated successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade updated successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }else if (res['message'] == 'Failed to insert grade.') {
            this.error = true;
            this.msg = 'Failed to insert grade.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }
        },
        error: (err) => {
          // Handle error if needed
          console.error('Error occurred while inserting grade for student:',
student.student_name, err);
        }
      });
    }
  }

}
```

Home CSS

```
.overlay{
    background-color: #00000038;
    width: 100%;
    position: absolute;
    height: 100%;
    z-index: 1;
}
.carousel-inner {
    height: 600px;
}
.carousel-caption{
    bottom: 35rem;
    z-index: 999;
}
.card{
    height: 100%;
}
.btn-custom{
    background-color: #e1825b;
    color: #fff;
}
```

Home HTML

```
      <app-header></app-header>


<div id="carouselExampleCaptions" class="carousel slide">
    <div class="carousel-indicators">
      <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
      <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide-to="1" aria-label="Slide 2"></button>
      <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-
slide-to="2" aria-label="Slide 3"></button>
    </div>
    <div class="carousel-inner">
        <div class="overlay"></div>
      <div class="carousel-item active">
        <img src="../../../assets/img/female-speaker-giving-presentation-hall-
university-workshop-audience-conference-hall.jpg" class="d-block w-100"
alt="...">
```

```
        <div class="carousel-caption d-none d-md-block">
            <h5>Enhance Your Presentation Skills</h5>
            <p>Learn how to captivate your audience and deliver powerful
presentations that leave a lasting impact.</p>
        </div>
      </div>
      <div class="carousel-item">
        <img src="../../../assets/img/successful-computer-gadget-digital-
close.jpg" class="d-block w-100" alt="...">
        <div class="carousel-caption d-none d-md-block">
            <h5>Master Digital Technologies</h5>
            <p>Unlock the potential of modern digital tools and gadgets to
streamline your workflow and boost productivity.</p>
        </div>
      </div>
      <div class="carousel-item">
        <img src="../../../assets/img/laptop-book-pile-desk.jpg" class="d-block
w-100" alt="...">
        <div class="carousel-caption d-none d-md-block">
            <h5>Explore Online Learning Opportunities</h5>
            <p>Discover a wealth of knowledge at your fingertips with our
extensive collection of online courses and resources.</p>
        </div>
      </div>
    </div>
    <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleCaptions" data-bs-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Previous</span>
    </button>
    <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleCaptions" data-bs-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Next</span>
    </button>
</div>
<div class="container mt-5">
    <h1 class="text-center mb-5">Our Features and Services</h1>
    <div class="row">
        <div class="col-md-3">
            <div class="card course-card">
                <img src="../../../assets/img/Online world-rafiki.png"
class="card-img-top w-50 m-auto" alt="Course 1">
                <div class="card-body">
                    <h5 class="card-title">Courses</h5>
```

```
                    <p class="card-text">Courses of Information Technology and
Business Administration</p>
                </div>
                <div class="card-footer text-center p-3">
                    <a href="courses" class="btn btn-custom">Enroll Now</a>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card course-card">
                <img src="../../../assets/img/Data analysis-rafiki.png"
class="card-img-top w-50 m-auto" alt="Course 2">
                <div class="card-body">
                    <h5 class="card-title">History and Facts</h5>
                    <p class="card-text">About the history of the university</p>
                </div>
                <div class="card-footer text-center p-3">
                    <a href="#" class="btn btn-custom">Enroll Now</a>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card course-card">
                <img src="../../../assets/img/Student stress-rafiki.png"
class="card-img-top w-50 m-auto" alt="Course 3">
                <div class="card-body">
                    <h5 class="card-title">Student Dashboard</h5>
                    <p class="card-text">About the students, activities and
courses</p>
                </div>
                <div class="card-footer text-center p-3">
                    <a href="#" class="btn btn-custom">Enroll Now</a>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card course-card">
                <img src="../../../assets/img/Instruction manual-rafiki.png"
class="card-img-top w-50 m-auto" alt="Course 3">
                <div class="card-body">
                    <h5 class="card-title">Instructor Dashboard</h5>
                    <p class="card-text">About the Instructors and courses</p>
                </div>
                <div class="card-footer text-center p-3">
                    <a href="#" class="btn btn-custom">Enroll Now</a>
```

```
                </div>
            </div>
        </div>
    </div>
</div>

<div class="p-5 mb-4 mt-4 bg-light rounded-3">
    <div class="container py-5">
        <h1 class="display-5 fw-bold">Rate - X</h1>
        <p class="col-md-8 fs-4">Student Evaluation System is going to be used as
a tool for academic advisor to help students who have academic difficulties.</p>
        <!-- <button class="btn btn-primary btn-lg" type="button">Example
button</button> -->
    </div>
</div>

<div class="container">
    <div class="row align-items-md-stretch">
        <div class="col-md-6">
            <div class="h-100 p-5 text-white bg-dark rounded-3">
                <h2>Enhance Your Learning Experience</h2>
                <p>Transform your learning journey by customizing your study
environment. Tailor background colors and text styles to suit your preferences,
fostering a personalized and engaging learning atmosphere.</p>
                <button class="btn btn-outline-light" type="button">Get
Started</button>
            </div>
        </div>
        <div class="col-md-6">
            <div class="h-100 p-5 bg-light border rounded-3">
                <h2>Optimize Student Feedback</h2>
                <p>Optimize the clarity and effectiveness of student feedback
with our comprehensive evaluation system. By incorporating borders, our platform
ensures that feedback is clearly defined, making it easier for students to
understand and act upon. Empower students with clear and actionable insights.</p>
                <button class="btn btn-outline-secondary" type="button">Explore
Features</button>
            </div>
        </div>
    </div>
</div>

<app-footer></app-footer>
```

Home angular

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { HomeComponent } from './home.component';

describe('HomeComponent', () => {
  let component: HomeComponent;
  let fixture: ComponentFixture<HomeComponent>;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [HomeComponent]
    });
    fixture = TestBed.createComponent(HomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Login CSS

```
a {
    text-decoration: none;
}
.login-page {
    width: 100%;
    height: 100vh;
    display: inline-block;
    display: flex;
    align-items: center;
}
.form-right i {
    font-size: 100px;
}
.form-right.h-100.text-white.text-center{
    background-image: url('../../../assets/img/images (1).jpg') !important;
    background-size: inherit;
```

```
    background-position: center;
    background-repeat: repeat;
}
.color-overlay{
    background-color: #00000052;
    width: 100%;
    height: 100%;
    position: absolute;
}
.fs-1{
    position: relative;
    z-index: 99999;
}
.pt-5{  padding-top: 30% !important;
}
```

Login HTML

```
<div class="login-page bg-light">
    <div class="container">
        <div class="row">
            <div class="col-lg-10 offset-lg-1">
              <h3 class="mb-3">Login Now</h3>
                <div class="bg-white shadow rounded">
                    <div class="row">
                        <div class="col-md-7 pe-0">
                            <div class="form-left h-100 py-5 px-5">
                                <form [formGroup]="loginForm"
(ngSubmit)="login()" class="row g-4">
                                    <div class="col-12 alert alert-danger"
*ngIf="showError">

                                        {{errorMsg}}
                                    </div>
                                    <div class="col-12">
                                        <label>Email<span class="text-
danger">*</span></label>

                                        <div class="input-group">
                                            <div class="input-group-text"><i
class="fa-solid fa-user"></i></div>

                                            <input type="text" class="form-
control" placeholder="Enter Username" name="email" formControlName="email">
                                        </div>
```

```
                                        <div class="danger"
*ngIf="loginForm.get('email')?.hasError('required') &&
loginForm.get('email')?.touched">
                                            Email is required
                                        </div>
                                    </div>

                                    <div class="col-12">
                                        <label>Password<span class="text-
danger">*</span></label>
                                        <div class="input-group">
                                            <div class="input-group-text"><i
class="fa fa-lock"></i></div>
                                            <input type="password" class="form-
control" placeholder="Enter Password" name="password" formControlName="password">
                                        </div>
                                        <div class="danger"
*ngIf="loginForm.get('password')?.hasError('required') &&
loginForm.get('password')?.touched">
                                            Password is required
                                        </div>
                                    </div>

                                    <!-- <div class="col-sm-6">
                                        <div class="form-check">
                                            <input class="form-check-input"
type="checkbox" id="inlineFormCheck">
                                            <label class="form-check-label"
for="inlineFormCheck">Remember me</label>
                                        </div>
                                    </div> -->

                                    <!-- <div class="col-sm-6">
                                        <a href="#" class="float-end text-
primary">Forgot Password?</a>
                                    </div> -->

                                    <div class="col-12">
                                        <button type="submit" class="btn btn-
primary px-4 float-end mt-4">login</button>
                                    </div>
                                </form>
                            </div>
                        </div>
```

```
                        <div class="col-md-5 ps-0 d-none d-md-block p-0"
style="position: relative;">
                            <div class="form-right h-100 text-white text-center">
                                <div class="color-overlay"></div>
                                <h2 class="fs-1 pt-5">Rate - X</h2>
                                <h2 class="fs-1">Welcome Back!!!</h2>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Login angular

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ApiService } from 'src/app/api.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  loginAccounts = [
    {
      name: 'Doctor',
      email: 'doctor@gmail.com',
      password: 'doctor123'
    },
    {
      name: 'Teaching assistant',
      email: 'teachingassistant@gmail.com',
      password: 'teachingassistant123'
    },
    {
      name: 'Academic Advisor',
      email: 'academicadvisor@gmail.com',
      password: 'academicadvisor123'
    },
    {
```

```
    name: 'Student',
    email: 'student@gmail.com',
    password: 'student123'
  }
];
userType: any;
userId: any;

loginForm: FormGroup;
constructor(private formBuilder: FormBuilder,private api: ApiService){
  this.loginForm = this.formBuilder.group({
    email: ['',[Validators.required]],
    password: ['',[Validators.required]]
  })
}

ngOnInit(): void {

}

errorMsg: any;
showError: boolean = false;
login() {
  if (this.loginForm.valid) {
    const loginData = this.loginForm.value;
    this.api.login(loginData).subscribe({
      next: (res: any) => {
        console.log(res);
        localStorage.setItem('userType',res.type)
        localStorage.setItem('userId',res.student_id)
        localStorage.setItem('name',res.name)
        if (res.success) {
          // Handle successful login
          window.location.href = 'home'
          console.log("Login successful");
        } else {
          this.errorMsg = res.message;
          this.showError = true;
          setTimeout(() => {
            this.showError = false;
          }, 2000);
        }
      },
      error: (err: any) => {
        console.error(err);
```

```
          this.errorMsg = "An error occurred during login.";
          this.showError = true;
          setTimeout(() => {
            this.showError = false;
          }, 2000);
        }
      });
    } else {
      this.errorMsg = 'Please fill in all required fields.';
      this.showError = true;
      setTimeout(() => {
        this.showError = false;
      }, 2000);
    }
  }

  checkPassword(e: any){
    console.log(e.target.value)
  }
}
```

Mid grades CSS

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
```

```
    height: 100%;
    position: absolute;
    background-color: #00000047;
}
```

Mid grade HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">Mid-Term Grade</h1>
    <div class="overlay"></div>
</div>
<div class="row m-3" *ngIf="insertedSuccessfully">
    <div class="alert alert-success">{{msg}}</div>
</div>
<div class="row m-3" *ngIf="error">
    <div class="alert alert-danger">{{msg}}</div>
</div>
<div class="row">
    <form [formGroup]="stuForm" class="col-12">
        <div class="row m-3">
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Number</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose student number" (valueChanged)="changeStudent($event)"
formControlName="s_id">
                        <option selected disabled value="0">Choose student
number</option>
                        <option *ngFor="let num of data"
[value]="num.id">{{num.s_number}}</option>
                    </ng-select2>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Name</label>
                    <input type="text" class="form-control"
value="{{studentName}}" placeholder="Student Name" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
```

```
                    <label class="form-label">Subject Name</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose Subject Name" (valueChanged)="changeTrack($event)"
formControlName="track_id">
                        <option selected disabled value="0">Choose Subject
Name</option>
                        <option *ngFor="let track of tracks"
[value]="track.id">{{track.name}}</option>
                    </ng-select2>
                    <!-- <input type="text" class="form-control"
placeholder="Enter subject name"> -->
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Mid-Term Grade</label>
                    <input type="text" class="form-control" placeholder="Enter
grade" name="midterm" formControlName="midterm">
                </div>
            </div>
            <div class="col-12 text-center">
                <button class="btn btn-custom" [disabled]="stuForm.invalid"
(click)="save()">Save</button>
            </div>
        </div>

        <hr class="mb-4 mt-4">

        <div class="row mx-3">
            <div class="col-12">
                <h6>Add students by excel sheet</h6>
                <p>
                    <a style="cursor: pointer;" (click)="downloadExcel()"><i
class="fa fa-cloud-download"></i> Download Sample of sheet</a>
                </p>
            </div>
            <div class="col-4 mt-4">
                <label class="btn btn-custom" for="uploadExcel">Upload Excel
File</label>
                <input type="file" hidden id="uploadExcel" accept=".xlsx,
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
(change)="onFileChange($event)">
            </div>
        </div>
    </form>
```

```
    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <div class="table-responsive">
                    <table class="table w-100 bordered" id="example">
                        <thead>
                            <tr>
                                <th>Num</th>
                                <th>Student name</th>
                                <th>Student Number</th>
                                <th>Subject Name</th>
                                <th>Mid-Term Grade</th>
                                <!-- <th>Action</th> -->
                            </tr>
                        </thead>
                        <tbody>
                            <tr *ngFor="let stu of studentsGrades;let i = index">
                                <td>{{i}}</td>
                                <td>{{stu.student_name}}</td>
                                <td>{{stu.student_number}}</td>
                                <td>{{stu.track_name}}</td>
                                <td>{{stu.midterm}}</td>
                                <!-- <td>
                                    <button class="btn btn-danger"><i class="fa
fa-trash"></i></button>
                                </td> -->
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Mid grade Angular

```
import { Component, OnInit } from '@angular/core';
import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { Select2OptionData } from 'ng-select2';
import { Options } from 'select2';
import { ApiService } from 'src/app/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import * as saveAs from 'file-saver';
import * as XLSX from 'xlsx';

@Component({
  selector: 'app-mid-grades',
  templateUrl: './mid-grades.component.html',
  styleUrls: ['./mid-grades.component.css']
})
export class MidGradesComponent implements OnInit {

  public options!: Options;
  data: any;
  tracks: any;
  stuForm: FormGroup;
  stuExcelForm: FormGroup;
  studentsGrades: any;
  gradeType!: string;


  constructor(private api: ApiService,private fb: FormBuilder){
    this.stuForm = this.fb.group({
      s_id:['0',[Validators.required]],
      track_id:['0',[Validators.required]],
      midterm:['',[Validators.required]]
    })

    this.stuExcelForm = this.fb.group({
      s_id:[''],
      track_id:[''],
      midterm:['']
    })
```

```
    this.api.get_student('').subscribe({ next: (res: any) => {
      this.data = res.data;
      // console.log(this.data)
    }})

    this.api.get_tracks().subscribe({ next: (res: any) => {
      this.tracks = res.data;
      // console.log(this.tracks)
    }})

    this.api.get_all_grads().subscribe({next: (res: any) => {
      this.studentsGrades = res.data;
      console.log(this.studentsGrades)
    }})
  }

  ngOnInit(): void {
    setTimeout(()=>{
      $('#example thead tr').clone(true).addClass('filters').appendTo('#example
thead');

      var table=$('#example').DataTable({
        "orderCellsTop": true,
        "pagingType": "full_numbers",
        scrollCollapse: true,
        "search": true,
        "lengthMenu": [[10, 25, 50,100, -1], [10, 25, 50,100, "All"]],
        "dom":  "<'row'<'col-md-4'l><'col-md-4'B><'col-md-4'f>><'row'<'col-md-
12't>><'row'<'col-md-6'i><'col-md-6'p>>",
        initComplete: function (api:any) {

       api.aoColumns.forEach(function(aoColumn:any) {

      var cell = $('.filters th').eq(
        aoColumn.idx
    );
    var title = $(cell).text();
    $(cell).html('<input type="text" placeholder="' + title + '"  class="' +
title + '" />');
     if(title == "Phone"){
      $(cell).html('<input type="number" placeholder="Phone" pattern="[0-9]{11}"
maxlength="11" onkeypress="if(this.value.length==11) return false; return /[0-
9]/i.test(event.key)"' + title + '" />');
     }
```

```
     if(title == "Action"){
      $(cell).html('<input type="number" style="display:none;"');
     }
    $('input',cell).off('keyup change').on('keyup change', function (e:any) {
                    e.stopPropagation();

            var regexr = '({search})';
$(this).parents('th').find('select').val();
            var cursorPosition = e.selectionStart;

                  var code = e.keyCode || e.which;
                  if (code == 13) {
                    console.log($(this),aoColumn);

                    table .column( aoColumn.idx )
                        .search(
                          $(this).val() != ''
                                      ? regexr.replace('{search}', '((' +
$(this).val() + '))')
                                      : '',
                                      $(this).val() != '',
                                      $(this).val() == ''
                              )
                              .draw();
                          }
                  });
              });
            },
        });
    },200);

    this.options = {
      multiple: false,
      closeOnSelect: true,
      width: '100%'
    };

    this.gradeType = 'midterm';
  }

  studentName: any;
  changeStudent(e: any){
    this.api.get_student(e).subscribe({next: (res: any) => {
      console.log(res.data)
      this.studentName = res.data.name
```

```
    }})
  }

  changeTrack(e: any){}

  insertedSuccessfully: boolean = false;
  error: boolean = false;
  msg: any;
  save(){
    console.log(this.stuForm)
    if(this.stuForm){
      this.api.insert_grades(this.stuForm.value).subscribe({next: (res: any) => {
        if(res['message'] == 'Grade inserted successfully.'){
          this.insertedSuccessfully = true;
          this.msg = 'Grade inserted successfully.';
          setTimeout(() => {
            this.insertedSuccessfully = false;
            window.location.reload();
          }, 2000);
        }else if(res['message'] == 'Failed to insert grade.'){
          this.error = true;
          this.msg = 'Failed to insert grade.';
          setTimeout(() => {
            this.error = false;
          }, 2000);
        }
      }})
    }
  }

  // download excel
  downloadExcel() {
    this.api.fetchStudentsWithSpecificGrade(this.gradeType).subscribe((response:
any) => {
      if (response.status === 'success') {
        this.exportToExcel(response.data);
      } else {
        console.error('Failed to fetch students data');
      }
    }, error => {
      console.error('Error fetching students data', error);
    });
  }

  exportToExcel(students: any[]) {
```

```
    // Ensure only the relevant grade field is included
    const updatedStudents = students.map(student => ({
      ...student,
      [this.gradeType]: student[this.gradeType] || ''
    }));

    const ws: XLSX.WorkSheet = XLSX.utils.json_to_sheet(updatedStudents);
    const wb: XLSX.WorkBook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(wb, ws, 'Students');
    const wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'array' });
    saveAs(new Blob([wbout], { type: 'application/octet-stream' }),
`students_${this.gradeType}.xlsx`);
  }
  onFileChange(evt: any) {
    const target: DataTransfer = <DataTransfer>(evt.target);
    if (target.files.length !== 1) throw new Error('Cannot use multiple files');
    const reader: FileReader = new FileReader();
    reader.onload = (e: any) => {
      const bstr: string = e.target.result;
      const wb: XLSX.WorkBook = XLSX.read(bstr, { type: 'binary' });
      const wsname: string = wb.SheetNames[0];
      const ws: XLSX.WorkSheet = wb.Sheets[wsname];
      const data = <any[][]>(XLSX.utils.sheet_to_json(ws, { header: 1 }));

      // Process the data and send it to the backend
      const students = this.processData(data);
      this.uploadStudents(students);
    };
    reader.readAsBinaryString(target.files[0]);
  }

  processData(data: any[][]): any[] {
    const students = [];

    for (let i = 1; i < data.length; i++) { // Assuming the first row is header
      const row = data[i];
      const student = {
        student_id: row[0],
        student_name: row[1],
        student_number: row[2],
        student_email: row[3],
        subject_id: row[4],
        subject_name: row[5],
        midterm: row[6]
      };
```

```
      students.push(student);
    }
    return students;
  }

  uploadStudents(students: any[]) {
    for (let i = 0; i < students.length; i++) {
      const student = students[i];
      const formData = {
        s_id: student.student_id,
        track_id: student.subject_id,
        midterm: student.midterm
      };

      // Send the individual student data to the API
      this.api.insert_grades(formData).subscribe({
        next: (res: any) => {
          if (res['message'] == 'Grade inserted successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle success if needed
            // console.log('Grade inserted successfully for student:',
student.student_name);

          } else if (res['message'] == 'Failed to insert grade.') {
            this.error = true;
            this.msg = 'Failed to insert grade.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }
        },
        error: (err) => {
          // Handle error if needed
          console.error('Error occurred while inserting grade for student:',
student.student_name, err);
```

```
        }
      });
    }
  }
}
```

Profile CSS

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
}
.heading{
    font-size: 25px;
    font-weight: 700;
    text-decoration-line: underline;
}
```

Profile HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">My Profile</h1>
    <div class="overlay"></div>
</div>
<div class="row">
    <form class="col-12">
        <div class="row m-3">
            <div class="col-12 mb-3">
                <span class="heading">My Data</span>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Number</label>
                    <input type="text" class="form-control" [value]="s_number"
readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Name</label>
                    <input type="text" class="form-control" [value]="name"
readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Email</label>
                    <input type="text" class="form-control" [value]="email"
readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Phone</label>
                    <input type="text" class="form-control" [value]="phone"
readonly disabled>
                </div>
            </div>
            <div class="col-12">
                <hr class="mb-4 mt-4">
            </div>
```

```
            <div class="col-12 mb-3">
                <span class="heading">My Behavior</span>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Attendance and departure
(%)</label>
                    <input type="number" class="form-control"
value="{{attendance}}" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Dealing with others (%)</label>
                    <input type="number" class="form-control"
value="{{dealing_other}}" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Dealing with teaching assistants
(%)</label>
                    <input type="number" class="form-control"
value="{{dealing_teach}}" readonly disabled>
                </div>
            </div>

            <div class="col-12">
                <hr class="mb-4 mt-4">
            </div>
            <div class="col-12 mb-3">
                <span class="heading">My Grades</span>
            </div>
            <div class="col-12">
                <div class="row" *ngFor="let item of stuGradesData;let i =
index;">
                    <div class="col-12 mb-3">
                        <span class="heading">Student Grades for
{{trackNames[i]}} subject</span>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Mid-Term Grade</label>
                            <input type="number" class="form-control"
value="{{item.midterm}}" readonly disabled>
```

```
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Final Grade</label>
                            <input type="number" class="form-control"
value="{{item.final}}" readonly disabled>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Assignment Grade</label>
                            <input type="number" class="form-control"
value="{{item.assignment}}" readonly disabled>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Quiz Grade</label>
                            <input type="number" class="form-control"
value="{{item.quiz}}" readonly disabled>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <hr class="mb-4 mt-4">
    </form>

    <div class="col-12" *ngIf="isDataLoaded">
        <div class="card m-3">
            <div class="card-body">
                <div class="row">
                    <div class="col-md-6">
                        <canvasjs-chart [options]="chartOptions"
[styles]="{width: '100%', height:'360px'}"></canvasjs-chart>
                    </div>
                    <div class="col-md-6">
                        <canvasjs-chart [options]="chartOptions2"
[styles]="{width: '100%', height:'360px'}"></canvasjs-chart>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

```
    <div class="col-12">
        <div class="card m-3" *ngFor="let msg of messagesData">
            <div class="card-body">
                <div class="row">
                    <div class="col-md-6 mb-2">
                        <div class="form-group">
                            <label class="form-label">Tracks</label>
                            <input type="text" class="form-control"
value="{{msg.track_name}}" readonly disabled>
                        </div>
                    </div>
                    <div class="col-md-6 mb-2">
                        <div class="form-group">
                            <label class="form-label">Notification
Message</label>
                            <textarea rows="3" readonly disabled class="form-
control">{{msg.message}}</textarea>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Profile Angular

```
import { Component, OnInit } from '@angular/core';
import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { Select2OptionData } from 'ng-select2';
import { Options } from 'select2';
import { ActivatedRoute } from '@angular/router';
import { ApiService } from 'src/app/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
```

```
  selector: 'app-profile',
  templateUrl: './profile.component.html',
  styleUrls: ['./profile.component.css']
})
export class ProfileComponent implements OnInit {

  public exampleData!: Array<Select2OptionData>;
  public options!: Options;
  id: any;
  stuData: any;
  stuGradesData: any;
  stuBehaviorData: any;
  tracksData: any;

  // student data
  s_number: any;
  phone: any;
  name: any;
  email: any;

  // student grades
  final: any;
  midterm: any;
  assignment: any;
  quiz: any;
  trackNames: any = [];
  messagesData:  any;

  // student behavior
  dealing_teach: any;
  dealing_other: any;
  attendance: any;

  // total grades
  totalFinal=0;
  totalMidterm=0;
  totalAssignment=0;
  totalQuiz=0;
  //
  totalDealing_teach=0;
  totalDealing_other=0;
  totalAttendance=0;

  isDataLoaded = false;
```

```
  notifyForm: FormGroup;

  constructor(private route: ActivatedRoute,private api: ApiService,private fb:
FormBuilder){
    this.id = this.route.snapshot.paramMap.get('id');

    this.notifyForm = this.fb.group({
      student_id:[''],
      assest_id:[''],
      message:['',[Validators.required]],
      track_id:['',[Validators.required]]
    })
  }

  notifications: any
  tracks: any
  errorMsg: any;
  ngOnInit(): void {
    console.log(this.id)

    this.api.get_notifications(this.id).subscribe({
      next: (res: any) => {
        console.log(res.notifications)
        this.messagesData = res.notifications;
        if (res.success) {
          this.notifications = res.notifications[0].message;
          this.tracks = res.notifications[0].track_name;
        } else {
          this.errorMsg = res.message;
        }
      },
      error: (err: any) => {
        console.error(err);
        this.errorMsg = 'An error occurred while fetching notifications.';
      }
    });


    this.api.get_student(this.id).subscribe({next: (res: any) => {
      this.stuData = res.data;
      this.s_number = this.stuData.s_number;
      this.phone = this.stuData.phone;
      this.name = this.stuData.name;
      this.email = this.stuData.email;
      // console.log('data ',this.stuData)
```

```
  }})
  this.api.get_student_behaviour(this.id).subscribe({next: (res: any) => {
    this.stuBehaviorData = res.data;
    this.dealing_teach = this.stuBehaviorData.dealing_teach;
    this.dealing_other = this.stuBehaviorData.dealing_other;
    this.attendance = this.stuBehaviorData.attendance;

    //
    this.totalDealing_teach = parseInt(this.dealing_teach)
    this.totalDealing_other = parseInt(this.dealing_other)
    this.totalAttendance = parseInt(this.attendance)
    // console.log(this.stuBehaviorData)
    // this.updateCharts();
  }})

  this.api.get_tracks().subscribe({
    next: (res: any) => {
      this.tracksData = res.data;
      // console.log(this.tracksData);

      // Construct payload with all track IDs
      const trackIds = this.tracksData.map((track: any) => track.id);
      const payload = {
        student_id: this.id,
        track_ids: trackIds
      };

      // Send payload to get_grads API
      this.api.get_grads(payload).subscribe({
        next: (res: any) => {
          this.stuGradesData = res.data;
          this.final = this.stuGradesData.final;
          this.midterm = this.stuGradesData.midterm;
          this.assignment = this.stuGradesData.assignment;
          this.quiz = this.stuGradesData.quiz;
          for(let i =0; i < this.stuGradesData.length ; i++){
            this.trackNames.push(this.stuGradesData[i].track_name)
            this.totalFinal += this.stuGradesData[i].final;
            this.totalMidterm += this.stuGradesData[i].midterm;
            this.totalAssignment += this.stuGradesData[i].assignment;
            this.totalQuiz += this.stuGradesData[i].quiz;
          }
          // console.log('grades', this.stuGradesData);
          // console.log('grades', this.trackNames);
          // console.log('final = '+this.totalFinal);
```

```
            // console.log('mid = '+this.totalMidterm);
            // console.log('ass = '+this.totalAssignment);
            // console.log('quiz = '+this.totalQuiz);
            this.updateCharts();
          }
        });
      }
    });

    // setTimeout(() => {
      // console.log('atten = '+this.totalAttendance);
      // console.log('other = '+this.totalDealing_other);
      // console.log('teach = '+this.totalDealing_teach);
    // }, 500);
    // this.api.get_tracks().subscribe({next: (res: any) => {
    //   this.tracksData = res.data;
    //   console.log(this.tracksData)
    // }})

    // const payload = {
    //   student_id: this.id,
    //   track_id: 1
    // };
    // this.api.get_grads(payload).subscribe({next: (res: any) => {
    //   this.stuGradesData = res.data;
    //   this.final = this.stuGradesData.final;
    //   this.midterm = this.stuGradesData.midterm;
    //   this.assignment = this.stuGradesData.assignment;
    //   this.quiz = this.stuGradesData.quiz;
    //   console.log('grades ',this.stuGradesData)
    // }})

      this.options = {
        multiple: true,
        closeOnSelect: false,
        width: '100%'
      };
  }

  updateCharts() {
    this.chartOptions.data[0].dataPoints = [
      { name: "Mid-Term Grade", y: isNaN(this.totalMidterm / 4) ? 0 :
this.totalMidterm / 4 },
      { name: "Final Grade", y: isNaN(this.totalFinal / 4) ? 0 : this.totalFinal
/ 4 },
```

```
      { name: "Assignment Grade", y: isNaN(this.totalAssignment / 4) ? 0 :
this.totalAssignment / 4 },
      { name: "Quiz Grade", y: isNaN(this.totalQuiz / 4) ? 0 : this.totalQuiz / 4
}
    ];

    this.chartOptions2.data[0].dataPoints = [
      { name: "Attendance and departure", y: isNaN((this.totalAttendance / 3) *
100) ? 0 : (this.totalAttendance / 3) * 100 },
      { name: "Dealing with others", y: isNaN((this.totalDealing_other / 3) *
100) ? 0 : (this.totalDealing_other / 3) * 100 },
      { name: "Dealing with teaching assistants", y:
isNaN((this.totalDealing_teach / 3) * 100) ? 0 : (this.totalDealing_teach / 3) *
100 }
    ];
    // console.log(this.chartOptions.data[0])
    this.isDataLoaded = true;
  }

  chartOptions = {
    animationEnabled: true,
    theme: "light",
    exportEnabled: false,
    title: {
      text: "Student Grades"
    },
    subtitles: [{
      text: "Grades degree"
    }],
    data: [{
      type: "pie",
      indexLabel: "{name}: {y}%",
      dataPoints: [] as { name: string; y: number; }[]
    }]
  }

  chartOptions2 = {
    animationEnabled: true,
    theme: "light",
    exportEnabled: false,
    title: {
      text: "Student Behavior"
    },
    subtitles: [{
      text: "Behavior degree"
```

```
    }],
    data: [{
      type: "pie",
      indexLabel: "{name}: {y}%",
      dataPoints: [] as { name: string; y: number; }[]
    }]
  }

  tracksIds: any = [];
  changeStudent(e: any){
    this.tracksIds = e;
    console.log(this.tracksIds)
  }

}
```

Quizez CSS

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
}
```

Quizzes HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">Quizes</h1>
    <div class="overlay"></div>
</div>
<div class="row m-3" *ngIf="insertedSuccessfully">
    <div class="alert alert-success">{{msg}}</div>
</div>
<div class="row m-3" *ngIf="error">
    <div class="alert alert-danger">{{msg}}</div>
</div>
<div class="row">
    <form class="col-12" [formGroup]="stuForm">
        <div class="row m-3">
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Number</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose student number" (valueChanged)="changeStudent($event)"
formControlName="s_id">
                        <option selected disabled value="0">Choose student
number</option>
                        <option *ngFor="let num of data"
[value]="num.id">{{num.s_number}}</option>
                    </ng-select2>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Name</label>
                    <input type="text" class="form-control"
value="{{studentName}}" placeholder="Student Name" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Subject Name</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose Subject Name" (valueChanged)="changeTrack($event)"
formControlName="track_id">
                        <option selected disabled value="0">Choose Subject
Name</option>
```

```
                        <option *ngFor="let track of tracks"
[value]="track.id">{{track.name}}</option>
                    </ng-select2>
                    <!-- <input type="text" class="form-control"
placeholder="Enter subject name"> -->
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Quiz Grade</label>
                    <input type="text" class="form-control" placeholder="Enter
grade" name="quiz" formControlName="quiz">
                </div>
            </div>
            <div class="col-12 text-center">
                <button class="btn btn-custom" [disabled]="stuForm.invalid"
(click)="save()">Save</button>
            </div>
        </div>

        <hr class="mb-4 mt-4">

        <div class="row mx-3">
            <div class="col-12">
                <h6>Add students by excel sheet</h6>
                <p>
                    <a style="cursor: pointer;" (click)="downloadExcel()"><i
class="fa fa-cloud-download"></i> Download Sample of sheet</a>
                </p>
            </div>
            <div class="col-4 mt-4">
                <label class="btn btn-custom" for="uploadExcel">Upload Excel
File</label>
                <input type="file" hidden id="uploadExcel" accept=".xlsx,
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
(change)="onFileChange($event)">
            </div>
        </div>
    </form>


    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <div class="table-responsive">
```

```
                    <table class="table w-100 bordered" id="example">
                        <thead>
                            <tr>
                                <th>Num</th>
                                <th>Student name</th>
                                <th>Student Number</th>
                                <th>Subject Name</th>
                                <th>Quiz Grade</th>
                                <!-- <th>Action</th> -->
                            </tr>
                        </thead>
                        <tbody>
                            <tr *ngFor="let stu of studentsGrades;let i = index">
                                <td>{{i}}</td>
                                <td>{{stu.student_name}}</td>
                                <td>{{stu.student_number}}</td>
                                <td>{{stu.track_name}}</td>
                                <td>{{stu.quiz}}</td>
                                <!-- <td>
                                    <button class="btn btn-danger"><i class="fa
fa-trash"></i></button>
                                </td> -->
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Quizzes Angular

```
import { Component, OnInit } from '@angular/core';
import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { Select2OptionData } from 'ng-select2';
import { Options } from 'select2';
```

```
import { ApiService } from 'src/app/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import * as saveAs from 'file-saver';
import * as XLSX from 'xlsx';

@Component({
  selector: 'app-quizes',
  templateUrl: './quizes.component.html',
  styleUrls: ['./quizes.component.css']
})
export class QuizesComponent implements OnInit {

  public options!: Options;
  data: any;
  tracks: any;
  stuForm: FormGroup;
  stuExcelForm: FormGroup;
  studentsGrades: any;
  gradeType!: string;

  constructor(private api: ApiService,private fb: FormBuilder){
    this.stuForm = this.fb.group({
      s_id:['0',[Validators.required]],
      track_id:['0',[Validators.required]],
      quiz:['',[Validators.required]]
    })

    this.stuExcelForm = this.fb.group({
      s_id:[''],
      track_id:[''],
      quiz:['']
    })

    this.api.get_student('').subscribe({ next: (res: any) => {
      this.data = res.data;
      // console.log(this.data)
    }})

    this.api.get_tracks().subscribe({ next: (res: any) => {
      this.tracks = res.data;
      // console.log(this.tracks)
    }})

    this.api.get_all_grads().subscribe({next: (res: any) => {
      this.studentsGrades = res.data;
```

```
      console.log(this.studentsGrades)
    }})
  }

  ngOnInit(): void {
    setTimeout(()=>{
      $('#example thead tr').clone(true).addClass('filters').appendTo('#example
thead');

      var table=$('#example').DataTable({
        "orderCellsTop": true,
        "pagingType": "full_numbers",
        scrollCollapse: true,
        "search": true,
        "lengthMenu": [[10, 25, 50,100, -1], [10, 25, 50,100, "All"]],
        "dom":  "<'row'<'col-md-4'l><'col-md-4'B><'col-md-4'f>><'row'<'col-md-
12't>><'row'<'col-md-6'i><'col-md-6'p>>",
        initComplete: function (api:any) {

       api.aoColumns.forEach(function(aoColumn:any) {

      var cell = $('.filters th').eq(
        aoColumn.idx
    );
    var title = $(cell).text();
    $(cell).html('<input type="text" placeholder="' + title + '"  class="' +
title + '" />');
     if(title == "Phone"){
      $(cell).html('<input type="number" placeholder="Phone" pattern="[0-9]{11}"
maxlength="11" onkeypress="if(this.value.length==11) return false; return /[0-
9]/i.test(event.key)"' + title + '" />');
     }
     if(title == "Action"){
      $(cell).html('<input type="number" style="display:none;"');
     }
    $('input',cell).off('keyup change').on('keyup change', function (e:any) {
                    e.stopPropagation();

            var regexr = '({search})';
$(this).parents('th').find('select').val();
            var cursorPosition = e.selectionStart;

                  var code = e.keyCode || e.which;
                  if (code == 13) {
                    console.log($(this),aoColumn);
```

```
                    table .column( aoColumn.idx )
                        .search(
                          $(this).val() != ''
                                      ? regexr.replace('{search}', '((((' +
$(this).val() + '))))')
                                      : '',
                                      $(this).val() != '',
                                      $(this).val() == ''
                                )
                                .draw();
                          }
                    });
              });
            },
        });
    },200);

    this.options = {
      multiple: false,
      closeOnSelect: true,
      width: '100%'
    };

    this.gradeType = 'quiz';
  }

  studentName: any;
  changeStudent(e: any){
    this.api.get_student(e).subscribe({next: (res: any) => {
      console.log(res.data)
      this.studentName = res.data.name
    }})
  }

  changeTrack(e: any){}

  insertedSuccessfully: boolean = false;
  error: boolean = false;
  msg: any;
  save(){
    console.log(this.stuForm)
    if(this.stuForm){
      this.api.insert_grades(this.stuForm.value).subscribe({next: (res: any) => {
        if(res['message'] == 'Grade inserted successfully.'){
```

```
          this.insertedSuccessfully = true;
          this.msg = 'Grade inserted successfully.';
          setTimeout(() => {
            this.insertedSuccessfully = false;
            window.location.reload();
          }, 2000);
        }else if(res['message'] == 'Failed to insert grade.'){
          this.error = true;
          this.msg = 'Failed to insert grade.';
          setTimeout(() => {
            this.error = false;
          }, 2000);
        }
      })
    }
  }

  // download excel
  downloadExcel() {
    this.api.fetchStudentsWithSpecificGrade(this.gradeType).subscribe((response:
any) => {
      if (response.status === 'success') {
        this.exportToExcel(response.data);
      } else {
        console.error('Failed to fetch students data');
      }
    }, error => {
      console.error('Error fetching students data', error);
    });
  }

  exportToExcel(students: any[]) {
    // Ensure only the relevant grade field is included
    const updatedStudents = students.map(student => ({
      ...student,
      [this.gradeType]: student[this.gradeType] || ''
    }));

    const ws: XLSX.WorkSheet = XLSX.utils.json_to_sheet(updatedStudents);
    const wb: XLSX.WorkBook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(wb, ws, 'Students');
    const wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'array' });
    saveAs(new Blob([wbout], { type: 'application/octet-stream' }),
`students_${this.gradeType}.xlsx`);
  }
```

```
onFileChange(evt: any) {
  const target: DataTransfer = <DataTransfer>(evt.target);
  if (target.files.length !== 1) throw new Error('Cannot use multiple files');
  const reader: FileReader = new FileReader();
  reader.onload = (e: any) => {
    const bstr: string = e.target.result;
    const wb: XLSX.WorkBook = XLSX.read(bstr, { type: 'binary' });
    const wsname: string = wb.SheetNames[0];
    const ws: XLSX.WorkSheet = wb.Sheets[wsname];
    const data = <any[][]>(XLSX.utils.sheet_to_json(ws, { header: 1 }));

    // Process the data and send it to the backend
    const students = this.processData(data);
    this.uploadStudents(students);
  };
  reader.readAsBinaryString(target.files[0]);
}

processData(data: any[][]): any[] {
  const students = [];

  for (let i = 1; i < data.length; i++) { // Assuming the first row is header
    const row = data[i];
    const student = {
      student_id: row[0],
      student_name: row[1],
      student_number: row[2],
      student_email: row[3],
      subject_id: row[4],
      subject_name: row[5],
      quiz: row[6]
    };

    students.push(student);
  }
  return students;
}

uploadStudents(students: any[]) {
  for (let i = 0; i < students.length; i++) {
    const student = students[i];
    const formData = {
      s_id: student.student_id,
      track_id: student.subject_id,
      quiz: student.quiz
```

```
      };

      // Send the individual student data to the API
      this.api.insert_grades(formData).subscribe({
        next: (res: any) => {
          console.log(res)
          if (res['message'] == 'Grade inserted successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle success if needed
            // console.log('Grade inserted successfully for student:',
student.student_name);

          }else if (res['message'] == 'Grade updated successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade updated successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }else if (res['message'] == 'Failed to insert grade.') {
            this.error = true;
            this.msg = 'Failed to insert grade.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }
        },
        error: (err) => {
          // Handle error if needed
          console.error('Error occurred while inserting grade for student:',
student.student_name, err);
        }
      });
```

```
    }
  }
}
```

Send notification Angular

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-send-notification',
  templateUrl: './send-notification.component.html',
  styleUrls: ['./send-notification.component.css']
})
export class SendNotificationComponent {

}
```

Services CSS

```
.services {
    padding: 3rem;
    font-family: sans-serif;
  }

  .container2 {
    margin: 0;
    padding: 0;
    list-style: none;
    display: grid;
    grid-template-columns: repeat(5, 1fr) [list-start] auto [list-end];
    position: relative;
    grid-template-rows: 4fr repeat(4, 1fr) 4fr;
    gap: 1rem;
  }

  li {
    text-decoration: underline;
    grid-column: list;
  }

  .container2 li:first-child {
```

```
  text-decoration: none;
}

li a {
  font-size: clamp(1rem, 3vw, 1.5rem);
  color: #000;
  text-decoration: none;
}

.container-inner {
  max-width: 100%;
  height: 100%;
  object-fit: cover;
  position: absolute;
  left: 0;
  top: 0;
  right: 0;
  bottom: 0;
  transition: 0.5s;
  border-radius: 20px;
  max-height: 300px;
  opacity: 0.75;
}

.container-inner:not(:hover, :focus) {
  animation: z-index-fix 0.5s;
}

.container2 li .container-inner {
  grid-row: 1/7;
  grid-column: var(--column);
  align-self: center;
}

li:has(:hover, :focus) a {
  background: rgba(50, 250, 200, 1);
}

li:has(:hover, :focus) .container-inner {
  transform: scale(1.1);
  max-height: 100%;
  z-index: 2;
  box-shadow: 50px 50px 50px 0px rgba(0, 0, 0, 0.2);
  opacity: 1;
}
```

```
  @keyframes z-index-fix {
    0%,
    100% {
      z-index: 1;
    }
  }
  .btn-custom{
    background-color: #e1825b;
    color: #fff;
}
```

Services HTML

```
<app-header></app-header>

<div class="container">
    <div class="row align-items-center">
        <div class="col-md-6">
            <img src="../../../assets/img/doctor_image.jpg" alt="Doctor Image"
class="img-fluid mb-4 w-100">
        </div>
        <div class="col-md-6">
            <div class="service-section">
                <h2>Thanks for being here, Doctor</h2>
                <p>Your dedication means a lot to us. As we get ready for another
round of teaching, I just want to say a quick thanks.</p>
            </div>
        </div>
    </div>
</div>

<div class='services'>
    <ul class='container2'>
      <li>
        <h2>Our Services</h2>
      </li>
      <li style='--column: 1/2'><a href="students">All
Students</a><a  href="students"><img class="container-inner"
src='../../../assets/img/students.jpg'></a></li>
      <li style='--column: 2/3'><a href="mid-grade">Student Mid Grades</a><a
href="mid-grade"><img class="container-inner" src='../../../assets/img/mid-
grade.jpg'></a></li>
```

```
      <li style='--column: 3/4'><a href="final-grade">Student Final Grades</a><a
href="final-grade"><img class="container-inner" src='../../../assets/img/final-
grade.jpg'></a></li>
      <li style='--column: 4/5' *ngIf="userType != 'doctor'"><a
href="quizes">Student Quizes </a><a href="quizes"><img class="container-inner"
src='../../../assets/img/quiz-grade.jpg'></a></li>
    </ul>
    <a class="btn btn-custom" href="all-services">Show More</a>
  </div>

<app-footer></app-footer>
```

Services Angular

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-services',
  templateUrl: './services.component.html',
  styleUrls: ['./services.component.css']
})
export class ServicesComponent implements OnInit {

  userType: any;
  ngOnInit(): void {
    this.userType = localStorage.getItem('userType')
  }
}
```

Show graph CSS

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
```

```
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
}
.heading{
    font-size: 25px;
    font-weight: 700;
    text-decoration-line: underline;
}
```

Show graph HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">Show Student Graph</h1>
    <div class="overlay"></div>
</div>
<div class="row">
    <form class="col-12">
        <div class="row m-3">
            <div class="col-12 mb-3">
                <span class="heading">Student Data</span>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Number</label>
                    <input type="text" class="form-control" [value]="s_number"
readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Name</label>
```

```
                    <input type="text" class="form-control" [value]="name"
readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Email</label>
                    <input type="text" class="form-control" [value]="email"
readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Phone</label>
                    <input type="text" class="form-control" [value]="phone"
readonly disabled>
                </div>
            </div>
            <div class="col-12">
                <hr class="mb-4 mt-4">
            </div>
            <div class="col-12 mb-3">
                <span class="heading">Student Behavior</span>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Attendance and departure
(%)</label>
                    <input type="number" class="form-control"
value="{{attendance}}" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Dealing with others (%)</label>
                    <input type="number" class="form-control"
value="{{dealing_other}}" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Dealing with teaching assistants
(%)</label>
                    <input type="number" class="form-control"
value="{{dealing_teach}}" readonly disabled>
```

```
                </div>
            </div>

            <div class="col-12">
                <hr class="mb-4 mt-4">
            </div>
            <div class="col-12">
                <div class="row" *ngFor="let item of stuGradesData;let i =
index;">
                    <div class="col-12 mb-3">
                        <span class="heading">Student Grades for
{{trackNames[i]}} subject</span>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Mid-Term Grade</label>
                            <input type="number" class="form-control"
value="{{item.midterm}}" readonly disabled>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Final Grade</label>
                            <input type="number" class="form-control"
value="{{item.final}}" readonly disabled>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Assignment Grade</label>
                            <input type="number" class="form-control"
value="{{item.assignment}}" readonly disabled>
                        </div>
                    </div>
                    <div class="col-md-4 mb-2">
                        <div class="form-group">
                            <label class="form-label">Quiz Grade</label>
                            <input type="number" class="form-control"
value="{{item.quiz}}" readonly disabled>
                        </div>
                    </div>
                </div>
            </div>
        </div>
```

```
        <hr class="mb-4 mt-4">


    </form>


    <div class="col-12" *ngIf="isDataLoaded">
        <div class="card m-3">
            <div class="card-body">
                <div class="row">
                    <div class="col-md-6">
                        <canvasjs-chart [options]="chartOptions"
[styles]="{width: '100%', height:'360px'}"></canvasjs-chart>
                    </div>
                    <div class="col-md-6">
                        <canvasjs-chart [options]="chartOptions2"
[styles]="{width: '100%', height:'360px'}"></canvasjs-chart>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <div class="row m-3" *ngIf="insertedSuccessfully">
        <div class="alert alert-success">{{msg}}</div>
    </div>
    <div class="row m-3" *ngIf="error">
        <div class="alert alert-danger">{{msg}}</div>
    </div>
    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <form class="row" [formGroup]="notifyForm">
                    <div class="col-md-6 mb-2">
                        <div class="form-group">
                            <label class="form-label">Tracks</label>
                            <ng-select2 class="form-control" [options]="options"
placeholder="Choose Track" (valueChanged)="changeStudent($event)"
formControlName="track_id">
                                <option *ngFor="let item of tracksData"
[value]="item.id">{{item.name}}</option>
                            </ng-select2>
                            <!-- <ng-select2 class="form-control"
[data]="exampleData" [options]="options"></ng-select2> -->
                        </div>
```

```
                    </div>
                    <div class="col-md-6 mb-2">
                        <div class="form-group">
                            <label class="form-label">Notification
Message</label>
                            <textarea placeholder="Enter message to student"
rows="3" class="form-control" formControlName="message"></textarea>
                        </div>
                    </div>
                    <div class="col-12 text-center">
                        <button class="btn btn-custom" (click)="sendNotify()"
[disabled]="notifyForm.invalid">Send</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Show graph Angular

```
import { Component, OnInit } from '@angular/core';
import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { Select2OptionData } from 'ng-select2';
import { Options } from 'select2';
import { ActivatedRoute } from '@angular/router';
import { ApiService } from 'src/app/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-show-graph',
  templateUrl: './show-graph.component.html',
  styleUrls: ['./show-graph.component.css']
})
export class ShowGraphComponent implements OnInit{

  public exampleData!: Array<Select2OptionData>;
```

```
  public options!: Options;
  id: any;
  stuData: any;
  stuGradesData: any;
  stuBehaviorData: any;
  tracksData: any;

  // student data
  s_number: any;
  phone: any;
  name: any;
  email: any;

  // student grades
  final: any;
  midterm: any;
  assignment: any;
  quiz: any;
  trackNames: any = [];

  // student behavior
  dealing_teach: any;
  dealing_other: any;
  attendance: any;

  // total grades
  totalFinal=0;
  totalMidterm=0;
  totalAssignment=0;
  totalQuiz=0;
  //
  totalDealing_teach=0;
  totalDealing_other=0;
  totalAttendance=0;

  isDataLoaded = false;

  notifyForm: FormGroup;


  constructor(private route: ActivatedRoute,private api: ApiService,private fb:
FormBuilder){
    this.id = this.route.snapshot.paramMap.get('id');

    this.notifyForm = this.fb.group({
```

```
    student_id:[''],
    assest_id:[''],
    message:['',[Validators.required]],
    track_id:['',[Validators.required]]
  })
}

ngOnInit(): void {
  console.log(this.id)


  this.api.get_student(this.id).subscribe({next: (res: any) => {
    this.stuData = res.data;
    this.s_number = this.stuData.s_number;
    this.phone = this.stuData.phone;
    this.name = this.stuData.name;
    this.email = this.stuData.email;
    // console.log('data ',this.stuData)
  }})
  this.api.get_student_behaviour(this.id).subscribe({next: (res: any) => {
    this.stuBehaviorData = res.data;
    this.dealing_teach = this.stuBehaviorData.dealing_teach;
    this.dealing_other = this.stuBehaviorData.dealing_other;
    this.attendance = this.stuBehaviorData.attendance;

    //
    this.totalDealing_teach = parseInt(this.dealing_teach)
    this.totalDealing_other = parseInt(this.dealing_other)
    this.totalAttendance = parseInt(this.attendance)
    // console.log(this.stuBehaviorData)
    // this.updateCharts();
  }})

  this.api.get_tracks().subscribe({
    next: (res: any) => {
      this.tracksData = res.data;
      // console.log(this.tracksData);

      // Construct payload with all track IDs
      const trackIds = this.tracksData.map((track: any) => track.id);
      const payload = {
        student_id: this.id,
        track_ids: trackIds
      };
```

```
      // Send payload to get_grads API
      this.api.get_grads(payload).subscribe({
        next: (res: any) => {
          this.stuGradesData = res.data;
          this.final = this.stuGradesData.final;
          this.midterm = this.stuGradesData.midterm;
          this.assignment = this.stuGradesData.assignment;
          this.quiz = this.stuGradesData.quiz;
          for(let i =0; i < this.stuGradesData.length ; i++){
            this.trackNames.push(this.stuGradesData[i].track_name)
            this.totalFinal += this.stuGradesData[i].final;
            this.totalMidterm += this.stuGradesData[i].midterm;
            this.totalAssignment += this.stuGradesData[i].assignment;
            this.totalQuiz += this.stuGradesData[i].quiz;
          }
          // console.log('grades', this.stuGradesData);
          // console.log('grades', this.trackNames);
          // console.log('final = '+this.totalFinal);
          // console.log('mid = '+this.totalMidterm);
          // console.log('ass = '+this.totalAssignment);
          // console.log('quiz = '+this.totalQuiz);
          this.updateCharts();
        }
      });
    }
  });

  // setTimeout(() => {
    // console.log('atten = '+this.totalAttendance);
    // console.log('other = '+this.totalDealing_other);
    // console.log('teach = '+this.totalDealing_teach);
  // }, 500);
  // this.api.get_tracks().subscribe({next: (res: any) => {
  //   this.tracksData = res.data;
  //   console.log(this.tracksData)
  // }})

  // const payload = {
  //   student_id: this.id,
  //   track_id: 1
  // };
  // this.api.get_grads(payload).subscribe({next: (res: any) => {
  //   this.stuGradesData = res.data;
  //   this.final = this.stuGradesData.final;
  //   this.midterm = this.stuGradesData.midterm;
```

```
    //   this.assignment = this.stuGradesData.assignment;
    //   this.quiz = this.stuGradesData.quiz;
    //   console.log('grades ',this.stuGradesData)
    // }})

      this.options = {
        multiple: true,
        closeOnSelect: false,
        width: '100%'
      };
  }

  updateCharts() {
    this.chartOptions.data[0].dataPoints = [
      { name: "Mid-Term Grade", y: isNaN(this.totalMidterm / 4) ? 0 :
this.totalMidterm / 4 },
      { name: "Final Grade", y: isNaN(this.totalFinal / 4) ? 0 : this.totalFinal
/ 4 },
      { name: "Assignment Grade", y: isNaN(this.totalAssignment / 4) ? 0 :
this.totalAssignment / 4 },
      { name: "Quiz Grade", y: isNaN(this.totalQuiz / 4) ? 0 : this.totalQuiz / 4
}
    ];

    this.chartOptions2.data[0].dataPoints = [
      { name: "Attendance and departure", y: isNaN((this.totalAttendance / 3) *
100) ? 0 : (this.totalAttendance / 3) * 100 },
      { name: "Dealing with others", y: isNaN((this.totalDealing_other / 3) *
100) ? 0 : (this.totalDealing_other / 3) * 100 },
      { name: "Dealing with teaching assistants", y:
isNaN((this.totalDealing_teach / 3) * 100) ? 0 : (this.totalDealing_teach / 3) *
100 }
    ];
    // console.log(this.chartOptions.data[0])
    this.isDataLoaded = true;
  }

  chartOptions = {
    animationEnabled: true,
    theme: "light",
    exportEnabled: false,
    title: {
      text: "Student Grades"
    },
    subtitles: [{
```

```
    text: "Grades degree"
  }],
  data: [{
    type: "pie",
    indexLabel: "{name}: {y}%",
    dataPoints: [] as { name: string; y: number; }[]
  }]
}

chartOptions2 = {
  animationEnabled: true,
  theme: "light",
  exportEnabled: false,
  title: {
    text: "Student Behavior"
  },
  subtitles: [{
    text: "Behavior degree"
  }],
  data: [{
    type: "pie",
    indexLabel: "{name}: {y}%",
    dataPoints: [] as { name: string; y: number; }[]
  }]
}

tracksIds: any = [];
changeStudent(e: any){
  this.tracksIds = e;
  console.log(this.tracksIds)
}

insertedSuccessfully: boolean = false;
error: boolean = false;
msg: any;
sendNotify(){
  this.notifyForm.value.student_id = this.id;
  this.notifyForm.value.assest_id = 1;
  // console.log(this.notifyForm)
  if(this.notifyForm){
    for(let i = 0; i < this.tracksIds.length; i++){
      this.notifyForm.value.track_id = this.tracksIds[i]
      this.notifyForm.value.student_id = this.id;
      this.notifyForm.value.assest_id = 1;
```

```
        this.api.insert_notifacations(this.notifyForm.value).subscribe({next:
(res: any) => {
          if(res['message'] == 'Notification inserted successfully.'){
            this.insertedSuccessfully = true;
            this.msg = 'Notification inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
          }else if(res['message'] == 'Notification updated successfully.'){
            this.insertedSuccessfully = true;
            this.msg = 'Notification updated successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
          }
          else if(res['message'] == 'Failed to update notification.'){
            this.error = true;
            this.msg = 'Failed to update notification.';
            setTimeout(() => {
              this.error = false;
            }, 2000);
          }else if(res['message'] == 'Failed to insert notification.'){
            this.error = true;
            this.msg = 'Failed to insert notification.';
            setTimeout(() => {
              this.error = false;
            }, 2000);
          }
          // console.log(res)
        })
      }
    }
  }
}
```

Student evaluation CSS

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
```

Student evaluation HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">Student Evaluation</h1>
    <div class="overlay"></div>
</div>
<div class="row m-3" *ngIf="insertedSuccessfully">
    <div class="alert alert-success">{{msg}}</div>
</div>
<div class="row m-3" *ngIf="error">
    <div class="alert alert-danger">{{msg}}</div>
</div>
<div class="row">
    <form class="col-12" [formGroup]="stuForm">
        <div class="row m-3">
```

```
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Number</label>
                    <ng-select2 class="form-control" [options]="options"
placeholder="Choose student number" (valueChanged)="changeStudent($event)"
formControlName="student_id">
                        <option selected disabled value="0">Choose student
number</option>
                        <option *ngFor="let num of data"
[value]="num.id">{{num.s_number}}</option>
                    </ng-select2>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Student Name</label>
                    <input type="text" class="form-control"
value="{{studentName}}" placeholder="Student Name" readonly disabled>
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Attendance and departure
(%)</label>
                    <input type="number" class="form-control" placeholder="Enter
ratio" name="dealing_teach" formControlName="dealing_teach">
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Dealing with others (%)</label>
                    <input type="number" class="form-control" placeholder="Enter
ratio" name="dealing_other" formControlName="dealing_other">
                </div>
            </div>
            <div class="col-md-4 mb-2">
                <div class="form-group">
                    <label class="form-label">Dealing with teaching assistants
(%)</label>
                    <input type="number" class="form-control" placeholder="Enter
ratio" name="attendance" formControlName="attendance">
                </div>
            </div>
            <div class="col-12 text-center">
```

```
                <button class="btn btn-custom" [disabled]="stuForm.invalid"
(click)="save()">Save</button>
            </div>
        </div>

        <hr class="mb-4 mt-4">

        <div class="row mx-3">
            <div class="col-12">
                <h6>Add students by excel sheet</h6>
                <p>
                    <a style="cursor: pointer;" (click)="downloadExcel()"><i
class="fa fa-cloud-download"></i> Download Sample of sheet</a>
                </p>
            </div>
            <div class="col-4 mt-4">
                <label class="btn btn-custom" for="uploadExcel">Upload Excel
File</label>
                <input type="file" hidden id="uploadExcel" accept=".xlsx,
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
(change)="onFileChange($event)">
            </div>
        </div>
    </form>


    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <div class="table-responsive">
                    <table class="table w-100 bordered" id="example">
                        <thead>
                            <tr>
                                <th>Num</th>
                                <th>Student name</th>
                                <th>Student Number</th>
                                <th>Attendance and departure</th>
                                <th>Dealing with others</th>
                                <th>Dealing with teaching assistants</th>
                                <!-- <th>Action</th> -->
                            </tr>
                        </thead>
                        <tbody>
                            <tr *ngFor="let stu of studentBehaviour;let i =
index">
```

```
                                <td>{{i}}</td>
                                <td>{{stu.student_name}}</td>
                                <td>{{stu.student_number}}</td>
                                <td>{{stu.dealing_teach}}</td>
                                <td>{{stu.dealing_other}}</td>
                                <td>{{stu.attendance}}</td>
                                <!-- <td>
                                    <button class="btn btn-danger"><i class="fa
fa-trash"></i></button>
                                </td> -->
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Student evaluation angular

```
import { Component, OnInit } from '@angular/core';
import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { Select2OptionData } from 'ng-select2';
import { Options } from 'select2';
import { ApiService } from 'src/app/api.service';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import * as saveAs from 'file-saver';
import * as XLSX from 'xlsx';

@Component({
  selector: 'app-student-evaluations',
  templateUrl: './student-evaluations.component.html',
  styleUrls: ['./student-evaluations.component.css']
})
export class StudentEvaluationsComponent implements OnInit{
```

```
  public options!: Options;
  data: any;
  stuForm: FormGroup;
  stuExcelForm: FormGroup;
  studentBehaviour: any;
  gradeType!: string;

  constructor(private api: ApiService,private fb: FormBuilder){
    this.stuForm = this.fb.group({
      student_id:['0',[Validators.required]],
      dealing_teach:['',[Validators.required]],
      dealing_other:['',[Validators.required]],
      attendance:['',[Validators.required]]
    })

    this.stuExcelForm = this.fb.group({
      student_id:[''],
      dealing_teach:[''],
      dealing_other:[''],
      attendance:['']
    })

    this.api.get_student('').subscribe({ next: (res: any) => {
      this.data = res.data;
      // console.log(this.data)
    }})

    this.api.get_all_student_behaviour().subscribe({next: (res: any) => {
      this.studentBehaviour = res.data;
      console.log(this.studentBehaviour)
    }})
  }

  ngOnInit(): void {
    setTimeout(()=>{
      $('#example thead tr').clone(true).addClass('filters').appendTo('#example
thead');

      var table=$('#example').DataTable({
        "orderCellsTop": true,
        "pagingType": "full_numbers",
        scrollCollapse: true,
        "search": true,
        "lengthMenu": [[10, 25, 50,100, -1], [10, 25, 50,100, "All"]],
```

```
        "dom":  "<'row'<'col-md-4'l><'col-md-4'B><'col-md-4'f>><'row'<'col-md-
12't>><'row'<'col-md-6'i><'col-md-6'p>>",
        initComplete: function (api:any) {

       api.aoColumns.forEach(function(aoColumn:any) {

      var cell = $('.filters th').eq(
        aoColumn.idx
    );
    var title = $(cell).text();
    $(cell).html('<input type="text" placeholder="' + title + '"  class="' +
title + '" />');
     if(title == "Phone"){
      $(cell).html('<input type="number" placeholder="Phone" pattern="[0-9]{11}"
maxlength="11" onkeypress="if(this.value.length==11) return false; return /[0-
9]/i.test(event.key)"' + title + '" />');
     }
     if(title == "Action"){
      $(cell).html('<input type="number" style="display:none;"');
     }
    $('input',cell).off('keyup change').on('keyup change', function (e:any) {
                    e.stopPropagation();

            var regexr = '({search})';
$(this).parents('th').find('select').val();
            var cursorPosition = e.selectionStart;

                  var code = e.keyCode || e.which;
                  if (code == 13) {
                    console.log($(this),aoColumn);

                    table .column( aoColumn.idx )
                        .search(
                          $(this).val() != ''
                                      ? regexr.replace('{search}', '((' +
$(this).val() + '))')
                                      : '',
                                      $(this).val() != '',
                                      $(this).val() == ''
                              )
                              .draw();
                          }
                  });
              });
            },
```

```
        });
    },500);

    this.options = {
      multiple: false,
      closeOnSelect: true,
      width: '100%'
    };
  }

  studentName: any;
  changeStudent(e: any){
    this.api.get_student(e).subscribe({next: (res: any) => {
      // console.log(res.data)
      this.studentName = res.data.name
    }})
  }

  changeTrack(e: any){}

  insertedSuccessfully: boolean = false;
  error: boolean = false;
  msg: any;
  save(){
    console.log(this.stuForm)
    if(this.stuForm){
      this.api.insert_student_behavior(this.stuForm.value).subscribe({next: (res:
any) => {

        if(res['message'] == 'student_behavior inserted successfully.'){
          this.insertedSuccessfully = true;
          this.msg = 'Student Behavior inserted successfully.';
          setTimeout(() => {
            this.insertedSuccessfully = false;
            window.location.reload();
          }, 2000);
        }else if (res['message'] == 'student_behavior updated successfully.') {
          this.insertedSuccessfully = true;
          this.msg = 'Student Behavior updated successfully.';
          setTimeout(() => {
            this.insertedSuccessfully = false;
            window.location.reload();
          }, 2000);
          // Handle success if needed
```

```
          // console.log('Grade inserted successfully for student:',
student.student_name);

        }
        else if(res['message'] == 'Failed to insert grade.'){
          this.error = true;
          this.msg = 'Failed to insert grade.';
          setTimeout(() => {
            this.error = false;
          }, 2000);
        }
      }})
    }
  }

  // download excel
  downloadExcel() {
    this.api.get_all_student_behaviour().subscribe((response: any) => {
      if (response.status === 'success') {
        this.exportToExcel(response.data);
      } else {
        console.error('Failed to fetch students data');
      }
    }, error => {
      console.error('Error fetching students data', error);
    });
  }

  exportToExcel(students: any[]) {
    // Ensure only the relevant grade field is included
    const updatedStudents = students.map(student => ({
      ...student
    }));

    const ws: XLSX.WorkSheet = XLSX.utils.json_to_sheet(updatedStudents);
    const wb: XLSX.WorkBook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(wb, ws, 'Students');
    const wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'array' });
    saveAs(new Blob([wbout], { type: 'application/octet-stream' }),
`students_behavior.xlsx`);
  }
  onFileChange(evt: any) {
    const target: DataTransfer = <DataTransfer>(evt.target);
    if (target.files.length !== 1) throw new Error('Cannot use multiple files');
    const reader: FileReader = new FileReader();
```

```
  reader.onload = (e: any) => {
    const bstr: string = e.target.result;
    const wb: XLSX.WorkBook = XLSX.read(bstr, { type: 'binary' });
    const wsname: string = wb.SheetNames[0];
    const ws: XLSX.WorkSheet = wb.Sheets[wsname];
    const data = <any[][]>(XLSX.utils.sheet_to_json(ws, { header: 1 }));

    // Process the data and send it to the backend
    const students = this.processData(data);
    this.uploadStudents(students);
  };
  reader.readAsBinaryString(target.files[0]);
}

processData(data: any[][]): any[] {
  const students = [];

  for (let i = 1; i < data.length; i++) { // Assuming the first row is header
    const row = data[i];
    const student = {
      student_id: row[0],
      student_name: row[1],
      student_number: row[2],
      attendance: row[3],
      dealing_teach: row[4],
      dealing_other: row[5]
    };

    students.push(student);
  }
  return students;
}

uploadStudents(students: any[]) {
  console.log(students)
  for (let i = 0; i < students.length; i++) {
    const student = students[i];
    const student_id = parseInt(student.student_id);
    const formData = {
      student_id: student_id,
      dealing_teach: student.dealing_teach,
      dealing_other: student.dealing_other,
      attendance: student.attendance
    };
```

```
      // Send the individual student data to the API
      this.api.insert_student_behavior(formData).subscribe({
        next: (res: any) => {
          console.log(res)
          if (res['message'] == 'student_behavior inserted successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Student Behavior inserted successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle success if needed
            // console.log('Grade inserted successfully for student:',
student.student_name);

          }else if (res['message'] == 'student_behavior updated successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Student Behavior updated successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle success if needed
            // console.log('Grade inserted successfully for student:',
student.student_name);

          }
          else if (res['message'] == 'student_behavior updated successfully.') {
            this.insertedSuccessfully = true;
            this.msg = 'Grade updated successfully.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 5000);
            // Handle failure if needed
            // console.log('Failed to insert grade for student:',
student.student_name);
          }else if (res['message'] == 'Failed to insert grade.') {
            this.error = true;
            this.msg = 'Failed to insert grade.';
            setTimeout(() => {
              this.insertedSuccessfully = false;
              window.location.reload();
            }, 2000);
            // Handle failure if needed
```

```
            // console.log('Failed to insert grade for student:',
student.student_name);
          }
        },
        error: (err) => {
          // Handle error if needed
          console.error('Error occurred while inserting grade for student:',
student.student_name, err);
        }
      });
    }
  }
}
```

Student CSS

```
.centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    z-index: 99;
}
.container2 {
    position: relative;
    text-align: center;
    color: white;
    width: 100%;
    height: 500px;
    background-position: top;
    background-size: 100%;
    background-image: url('../../../assets/img/female-speaker-giving-
presentation-hall-university-workshop-audience-conference-hall.jpg');
}
.overlay{
    width: 100%;
    height: 100%;
    position: absolute;
    background-color: #00000047;
```

Student HTML

```
<app-header></app-header>
<div class="container2">
    <!-- <img height="500" src="" alt="Snow" style="width:100%;"> -->
    <h1 class="centered">All Students</h1>
    <div class="overlay"></div>
</div>
<div class="row">
    <div class="col-12 text-center">
        <a href="add-students" class="btn btn-custom m-4">Add new Students</a>
    </div>
    <div class="col-12">
        <div class="card m-3">
            <div class="card-body">
                <div class="table-responsive">
                    <table class="table w-100 bordered" id="example">
                        <thead>
                            <tr>
                                <th>Num</th>
                                <th>Student name</th>
                                <th>Email</th>
                                <th>Phone</th>
                                <th>Student Number</th>
                                <th>State</th>
                                <th *ngIf="userType == 'advisor'">Action</th>
                            </tr>
                        </thead>
                        <tbody>
                            <tr *ngFor="let stu of data;let i = index">
                                <td>{{i+1}}</td>
                                <td>{{stu.name}}</td>
                                <td>{{stu.email}}</td>
                                <td>{{stu.phone}}</td>
                                <td>{{stu.s_number}}</td>
                                <td>
                                    <span class="badge badge-success"
style="background: green;" *ngIf="stu.state == 1">Active</span>
                                    <span class="badge badge-danger"
style="background: red;" *ngIf="stu.state == 2">Inactive</span>
                                </td>
                                <td *ngIf="userType == 'advisor'">
                                    <div class="dropdown">
                                        <button class="btn btn-secondary
dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-expanded="false">
```

```
                                        <i class="fa fa-gear"></i>
                                    </button>
                                    <ul class="dropdown-menu">
                                        <!-- <button class="btn btn-
danger"></button> -->
                                        <li><a class="dropdown-item"
href="show-graph/{{stu.id}}"><i class="fa fa-chart-simple"></i> Show Graph and
send notification</a></li>
                                        <!-- <li><a class="dropdown-item"
href="send-notification/{{stu.id}}"><i class="fa fa-bell"></i> Send
Notification</a></li> -->
                                        <!-- <li><a class="dropdown-item"><i
class="fa fa-trash"></i> Delete</a></li> -->
                                    </ul>
                                </div>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<app-footer></app-footer>
```

Student Angular

```
      import { Component, OnInit } from '@angular/core';

import * as $ from 'jquery';
import 'jqueryui';
import 'datatables.net';
import 'datatables.net-buttons/js/dataTables.buttons.min';
import 'datatables.net-buttons/js/buttons.html5.min';
import 'datatables.net-buttons/js/buttons.print.min';
import 'datatables.net-buttons/js/buttons.colVis.min';
import { ApiService } from 'src/app/api.service';

@Component({
  selector: 'app-students',
  templateUrl: './students.component.html',
  styleUrls: ['./students.component.css']
})
```

```
export class StudentsComponent implements OnInit {

  data: any;
  userType: any;

  constructor(private api: ApiService){
    this.api.get_student('').subscribe({ next: (res: any) => {
      this.data = res.data
      console.log(this.data)
    }})
  }

  ngOnInit(): void {
    this.userType = localStorage.getItem('userType')
    setTimeout(()=>{
      $('#example thead tr').clone(true).addClass('filters').appendTo('#example
thead');

      var table=$('#example').DataTable({
        "orderCellsTop": true,
        "pagingType": "full_numbers",
        scrollCollapse: true,
        "search": true,
        "lengthMenu": [[10, 25, 50,100, -1], [10, 25, 50,100, "All"]],
        "dom":  "<'row'<'col-md-4'l><'col-md-4'B><'col-md-4'f>><'row'<'col-md-
12't>><'row'<'col-md-6'i><'col-md-6'p>>",
        initComplete: function (api:any) {

       api.aoColumns.forEach(function(aoColumn:any) {

      var cell = $('.filters th').eq(
        aoColumn.idx
    );
    var title = $(cell).text();
    $(cell).html('<input type="text" placeholder="' + title + '"  class="' +
title + '" />');
     if(title == "Phone"){
      $(cell).html('<input type="number" placeholder="Phone" pattern="[0-9]{11}"
maxlength="11" onkeypress="if(this.value.length==11) return false; return /[0-
9]/i.test(event.key)"' + title + '" />');
     }
     if(title == "Action"){
      $(cell).html('<input type="number" style="display:none;"');
     }
    $('input',cell).off('keyup change').on('keyup change', function (e:any) {
```

```
                    e.stopPropagation();

            var regexr = '({search})';
$(this).parents('th').find('select').val();
            var cursorPosition = e.selectionStart;

                  var code = e.keyCode || e.which;
                  if (code == 13) {
                    console.log($(this),aoColumn);

                    table .column( aoColumn.idx )
                        .search(
                          $(this).val() != ''
                                      ? regexr.replace('{search}', '((' +
$(this).val() + '))')
                                      : '',
                                      $(this).val() != '',
                                      $(this).val() == ''
                              )
                              .draw();
                          }
                  });
              });
            },
        });
    },500);
  }
}
```

API

```
import { TestBed } from '@angular/core/testing';

import { ApiService } from './api.service';

describe('ApiService', () => {
  let service: ApiService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(ApiService);
```

```
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

Back end

Database

```
<?php
header("Access-Control-Allow-Origin: *");
header('Access-Control-Allow-Credentials: true');
header("Access-Control-Allow-Methods: PUT, GET, POST, DELETE");
header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type,
Accept");
header("Content-Type: application/json; charset=UTF-8");
$host='localhost:3306';
$user='root';
$PASS='';
$NAME='ratex';
define('HOST',$host);
define('USER',$user);
define('PASS',$PASS);
define('NAME',$NAME);
$db = new mysqli(HOST ,USER ,PASS ,NAME);
if ($db->connect_errno) {
    die("Database connection error:" . $db->connect_errno);
}
$db -> set_charset("utf8");
?>
```

Delete grade

```php
<?php
include 'database.php';

$input = json_decode(file_get_contents('php://input'), true);
$student_id = $input['student_id'];
$track_id = $input['track_id'];
$type = $input['type'];

$db->begin_transaction();

try {
    if($type == 'final'){
        $stmt = $db->prepare("
        UPDATE student_grads
        SET final = 0
        WHERE s_id = ? AND track_id = ? AND state = 1");
    }else if($type == 'midterm'){
        $stmt = $db->prepare("
        UPDATE student_grads
        SET midterm = 0
        WHERE s_id = ? AND track_id = ? AND state = 1");
    }else if($type == 'assignment'){
        $stmt = $db->prepare("
        UPDATE student_grads
        SET assignment = 0
        WHERE s_id = ? AND track_id = ? AND state = 1");
    }else if($type == 'quiz'){
        $stmt = $db->prepare("
        UPDATE student_grads
        SET quiz = 0
        WHERE s_id = ? AND track_id = ? AND state = 1");
    }
    $stmt->bind_param("ii", $student_id, $track_id);

    if ($stmt->execute()) {
        // Check if any rows were updated
        if ($stmt->affected_rows > 0) {
            // Commit transaction
            $db->commit();
            $response = array("status" => "success", "message" => "Grades updated
successfully.");
        } else {
            // Rollback transaction if no grades were updated
```

```
            $db->rollback();
            $response = array("status" => "error", "message" => "No grades found
for the given user and track.");
        }
    } else {
        // Rollback transaction in case of error
        $db->rollback();
        $response = array("status" => "error", "message" => "Failed to update
grades.");
    }
} catch (Exception $e) {
    // Rollback transaction in case of error
    $db->rollback();
    $response = array("status" => "error", "message" => "Failed to update
grades.");
}

$stmt->close();
$db->close();

// Return JSON response
header('Content-Type: application/json');
echo json_encode($response);
?>
```

Delete student

```
<?php
include 'database.php';

$student_id = $_GET['id'];
$db->begin_transaction();

try {
    $stmt = $db->prepare("UPDATE students SET state = 0 WHERE id = ?");
    $stmt->bind_param("i", $student_id);
    $stmt->execute();

    if ($stmt->affected_rows > 0) {
        $stmt = $db->prepare("UPDATE users SET state = 0 WHERE s_id = ?");
        $stmt->bind_param("i", $student_id);
        $stmt->execute();
```

```
        $stmt = $db->prepare("UPDATE student_grads SET state = 0 WHERE s_id =
?");
        $stmt->bind_param("i", $student_id);
        $stmt->execute();
        $stmt = $db->prepare("UPDATE student_behavior SET state = 0 WHERE s_id =
?");
        $stmt->bind_param("i", $student_id);
        $stmt->execute();
        $db->commit();
        $response = array("status" => "success", "message" => "State updated
successfully.");
    } else {
        $db->rollback();
        $response = array("status" => "error", "message" => "Student not
found.");
    }
} catch (Exception $e) {
    $db->rollback();
    $response = array("status" => "error", "message" => "Failed to update
state.");
}

$stmt->close();
$db->close();

echo json_encode($response);
?>
```

Get all student behavior

```
<?php
include 'database.php';

// Set the correct content type header
header('Content-Type: application/json');

$response = array();

if (isset($_GET['id'])) {
    // Fetch specific student
    $student_id = $_GET['id'];
```

```
    $sql = "
        SELECT
            s.id AS student_id,
            s.name AS student_name,       -- Replace 'name' with the actual
column name for student name
            s.s_number AS student_number, -- Replace 's_number' with the actual
column name for student number
            sb.attendance,
            sb.dealing_teach,
            sb.dealing_other
        FROM
            students s
        LEFT JOIN
            student_behavior sb ON s.id = sb.s_id
        WHERE
            s.id = ? AND sb.state IS NOT NULL
    ";

    $stmt = $db->prepare($sql);
    $stmt->bind_param("i", $student_id);

    if ($stmt->execute()) {
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            // Fetch the student data
            $student_data = $result->fetch_assoc();
            $response = array("status" => "success", "data" => $student_data);
        } else {
            $response = array("status" => "error", "message" => "Student not
found or no behavior record with non-null state.");
        }
    } else {
        $response = array("status" => "error", "message" => "Failed to retrieve
student data.");
    }

    $stmt->close();
} else {
    // Fetch all students with non-null state in student_behavior
    $sql = "
        SELECT
            s.id AS student_id,
            s.name AS student_name,       -- Replace 'name' with the actual
column name for student name
```

```
            s.s_number AS student_number, -- Replace 's_number' with the actual
column name for student number
            sb.attendance,
            sb.dealing_teach,
            sb.dealing_other
        FROM
            students s
        LEFT JOIN
            student_behavior sb ON s.id = sb.s_id
        WHERE
            sb.state IS NOT NULL
    ";

    $stmt = $db->prepare($sql);

    if ($stmt->execute()) {
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            $students = array();
            while ($student_data = $result->fetch_assoc()) {
                $students[] = $student_data;
            }
            $response = array("status" => "success", "data" => $students);
        } else {
            $response = array("status" => "error", "message" => "No students
found.");
        }
    } else {
        $response = array("status" => "error", "message" => "Failed to retrieve
students data.");
    }

    $stmt->close();
}

$db->close();

// Return JSON response
echo json_encode($response);
?>
```

Get all students with grades

```php
<?php
include 'database.php';

// Prepare the SQL statement
$stmt = $db->prepare("
    SELECT
        student_grads.id,
        student_grads.s_id,
        student_grads.final,
        student_grads.midterm,
        student_grads.quiz,
        student_grads.assignment,
        student_grads.track_id,
        students.name AS student_name,
        students.s_number AS student_number,
        tracks.name AS track_name
    FROM
        student_grads
    JOIN
        students ON student_grads.s_id = students.id
    JOIN
        tracks ON student_grads.track_id = tracks.id
    WHERE
        student_grads.state = 1
    AND
        students.state = 1");

if ($stmt->execute()) {
    $result = $stmt->get_result();
    $grades = array();

    while ($row = $result->fetch_assoc()) {
        $grades[] = $row;
    }

    if (count($grades) > 0) {
        $response = array("status" => "success", "data" => $grades);
    } else {
        $response = array("status" => "error", "message" => "No grades found.");
    }
} else {
    $response = array("status" => "error", "message" => "Failed to retrieve
grades.");
```

```
}

$stmt->close();
$db->close();

// Return JSON response
header('Content-Type: application/json');
echo json_encode($response);
?>
```

Get grades

```
<?php
include 'database.php';

// Set the content type to JSON
header('Content-Type: application/json');

// Get the input data
$input = json_decode(file_get_contents('php://input'), true);

if ($input === null) {
    echo json_encode(array("status" => "error", "message" => "Invalid JSON
input."));
    exit;
}

$student_id = isset($input['student_id']) ? intval($input['student_id']) : null;
$track_id = isset($input['track_id']) ? intval($input['track_id']) : null;

if ($student_id === null) {
    echo json_encode(array("status" => "error", "message" => "Student ID is
required."));
    exit;
}

$stmt = null;

// Prepare the SQL statement
if ($track_id !== null) {
    $stmt = $db->prepare("
```

```
        SELECT student_grads.id, student_grads.s_id, student_grads.final,
student_grads.midterm, student_grads.quiz,
        student_grads.assignment, tracks.name AS track_name, tracks.id AS
track_id
        FROM student_grads
        JOIN tracks ON student_grads.track_id = tracks.id
        JOIN users ON student_grads.s_id = users.s_id
        WHERE student_grads.s_id = ? AND student_grads.track_id = ? AND
student_grads.state = 1 AND users.state = 1");
    $stmt->bind_param("ii", $student_id, $track_id);
} else {
    $stmt = $db->prepare("
        SELECT student_grads.id, student_grads.s_id, student_grads.final,
student_grads.midterm, student_grads.quiz,
        student_grads.assignment, tracks.name AS track_name, tracks.id AS
track_id
        FROM student_grads
        JOIN tracks ON student_grads.track_id = tracks.id
        JOIN users ON student_grads.s_id = users.s_id
        WHERE student_grads.s_id = ? AND student_grads.state = 1 AND users.state
= 1");
    $stmt->bind_param("i", $student_id);
}

if ($stmt->execute()) {
    $result = $stmt->get_result();
    $grades = array();

    while ($row = $result->fetch_assoc()) {
        $grades[] = $row;
    }

    if (count($grades) > 0) {
        $response = array("status" => "success", "data" => $grades);
    } else {
        $response = array("status" => "error", "message" => "No grades found for
the given user.");
    }
} else {
    $response = array("status" => "error", "message" => "Failed to retrieve
grades.");
}

$stmt->close();
$db->close();
```

```
echo json_encode($response);
?>
```

Get notification

```
<?php
include 'database.php';

header('Content-Type: application/json');

$input = json_decode(file_get_contents('php://input'), true);

if (isset($input['student_id'])) {
    $student_id = $db->real_escape_string($input['student_id']);

    $sql = "SELECT notifications.message, notifications.track_id, tracks.name
            FROM notifications
            JOIN tracks ON notifications.track_id = tracks.id
            WHERE notifications.student_id = ?";
    $stmt = $db->prepare($sql);
    $stmt->bind_param("i", $student_id);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        $notifications = array();
        while ($row = $result->fetch_assoc()) {
            $notifications[] = array(
                "message" => $row['message'],
                "track_id" => $row['track_id'],
                "track_name" => $row['name']
            );
        }
        echo json_encode(array("success" => true, "notifications" =>
$notifications));
    } else {
        echo json_encode(array("success" => false, "message" => "No notifications
found for the given student_id"));
    }

    $stmt->close();
} else {
```

```
    echo json_encode(array("success" => false, "message" => "student_id is
required"));
}

$db->close();
?>
```

Get students

```
?>


<?php
include 'database.php';

// Set the correct content type header
header('Content-Type: application/json');

$response = array();

if (isset($_GET['id'])) {
    // Fetch specific student
    $student_id = $_GET['id'];

    $stmt = $db->prepare("SELECT * FROM students WHERE id = ? AND state = 1");
    $stmt->bind_param("i", $student_id);

    if ($stmt->execute()) {
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            // Fetch the student data
            $student_data = $result->fetch_assoc();
            $response = array("status" => "success", "data" => $student_data);
        } else {
            $response = array("status" => "error", "message" => "Student not
found.");
        }
    } else {
        $response = array("status" => "error", "message" => "Failed to retrieve
student data.");
    }
```

```
    $stmt->close();
} else {
    // Fetch all students
    $stmt = $db->prepare("SELECT * FROM students WHERE state = 1");

    if ($stmt->execute()) {
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            $students = array();
            while ($student_data = $result->fetch_assoc()) {
                $students[] = $student_data;
            }
            $response = array("status" => "success", "data" => $students);
        } else {
            $response = array("status" => "error", "message" => "No students
found.");
        }
    } else {
        $response = array("status" => "error", "message" => "Failed to retrieve
students data.");
    }

    $stmt->close();
}

$db->close();

// Return JSON response
echo json_encode($response);
?>
```

Get student behvior

```php
<?php
include 'database.php';

// Check if 'id' parameter is set
if (isset($_GET['id'])) {
    $student_id = intval($_GET['id']); // Convert the id to an integer

    $stmt = $db->prepare("SELECT * FROM student_behavior WHERE s_id = ? AND
state=1");
    $stmt->bind_param("i", $student_id);

    if ($stmt->execute()) {
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            // Fetch the student data
            $student_data = $result->fetch_assoc();
            $response = array("status" => "success", "data" => $student_data);
        } else {
            $response = array("status" => "error", "message" => "Student not
found.");
        }
    } else {
        $response = array("status" => "error", "message" => "Failed to retrieve
student data.");
    }

    $stmt->close();
    $db->close();
} else {
    $response = array("status" => "error", "message" => "No student ID
provided.");
}

// Return JSON response
echo json_encode($response);
?>
```

Get student number

```php
<?php
include 'database.php';

$student_id = $_GET['id'];

$stmt = $db->prepare("SELECT * FROM students WHERE s_number = ? AND state=1");
$stmt->bind_param("i", $student_id);

if ($stmt->execute()) {
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        // Fetch the student data
        $student_data = $result->fetch_assoc();
        $response = array("status" => "success", "data" => $student_data);
    } else {
        $response = array("status" => "error", "message" => "Student not
found.");
    }
} else {
    $response = array("status" => "error", "message" => "Failed to retrieve
student data.");
}

$stmt->close();
$db->close();

// Return JSON response
echo json_encode($response);
?>
```

Get student with specific grade

```php
<?php
include 'database.php';

// Get the requested grade type from the query parameter
$gradeType = isset($_GET['grade_type']) ? $_GET['grade_type'] : '';

// Validate the grade type
```

```php
$validGrades = ['midterm', 'final', 'assignment', 'quiz'];
if (!in_array($gradeType, $validGrades)) {
    echo json_encode(["status" => "error", "message" => "Invalid grade type."]);
    exit();
}

// Prepare the SQL statement to include all students and join with tracks table
$stmt = $db->prepare("
    SELECT
        students.id AS student_id,
        students.name AS student_name,
        students.s_number AS student_number,
        students.email AS student_email,
        tracks.id AS subject_id,
        tracks.name AS subject_name,
        COALESCE(student_grads.$gradeType, 'No Grade') AS grade
    FROM
        students
    LEFT JOIN
        student_grads ON student_grads.s_id = students.id AND student_grads.state
= 1
    LEFT JOIN
        tracks ON student_grads.track_id = tracks.id
    WHERE
        students.state = 1");

if ($stmt->execute()) {
    $result = $stmt->get_result();
    $grades = array();

    while ($row = $result->fetch_assoc()) {
        // Rename the grade column to the specified grade type
        $row[$gradeType] = $row['grade'];
        unset($row['grade']);
        $grades[] = $row;
    }

    if (count($grades) > 0) {
        $response = array("status" => "success", "data" => $grades);
    } else {
        $response = array("status" => "error", "message" => "No grades found.");
    }
} else {
    $response = array("status" => "error", "message" => "Failed to retrieve
grades.");
```

```
}

$stmt->close();
$db->close();

// Return JSON response
header('Content-Type: application/json');
echo json_encode($response);
?>
```

Get tracks

```
include 'database.php';

$stmt = $db->prepare("SELECT * FROM tracks");

if ($stmt->execute()) {
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        // Fetch all tracks data
        $data = array();
        while ($row = $result->fetch_assoc()) {
            $data[] = $row;
        }
        $response = array("status" => "success", "data" => $data);
    } else {
        $response = array("status" => "error", "message" => "No tracks.");
    }
} else {
    $response = array("status" => "error", "message" => "Failed to retrieve
tracks data.");
}

$stmt->close();
$db->close();

// Return JSON response
echo json_encode($response);
?>
```

Import student

```php
<?php
include 'database.php';

$input = json_decode(file_get_contents('php://input'), true);
$students = $input['students'];

foreach ($students as $student) {
    $name = $student['name'];
    $phone = $student['phone'];
    $email = $student['email'];
    $password = $student['password'];
    $s_number = $student['s_number'];

    // Check if student exists
    $stmt = $db->prepare("SELECT id, phone, s_number FROM students WHERE email =
?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        // Student exists, fetch current data
        $stmt->bind_result($id, $current_phone, $current_s_number);
        $stmt->fetch();

        // Check if data has changed
        if ($phone !== $current_phone || $s_number !== $current_s_number) {
            // Update student data
            $stmt = $db->prepare("UPDATE students SET name = ?, phone = ?,
s_number = ? WHERE id = ?");
            $stmt->bind_param("sssi", $name, $phone, $s_number, $id);
            $stmt->execute();
        }
    } else {
        // Insert new student
        $stmt = $db->prepare("INSERT INTO students (name, phone, email, s_number)
VALUES (?, ?, ?, ?)");
        $stmt->bind_param("ssss", $name, $phone, $email, $s_number);
        $stmt->execute();

        // Get the new student's ID
        $student_id = $stmt->insert_id;
```

```
        // Hash the password and insert into users table
        $hashed_password = password_hash($password, PASSWORD_DEFAULT);
        $type = 'student';
        $stmt = $db->prepare("INSERT INTO users (s_id, name, email, password,
type) VALUES (?, ?, ?, ?, ?)");
        $stmt->bind_param("issss", $student_id, $name, $email, $hashed_password,
$type);
        $stmt->execute();
    }
}

$stmt->close();
$db->close();

$response = array("status" => "success", "message" => "Students data processed
successfully.");
echo json_encode($response);
?>
```

Insert behavior

```
<?php
include 'database.php';

$input = json_decode(file_get_contents('php://input'), true);

$dealing_teach = $input['dealing_teach'];
$student_id = $input['student_id'];
$dealing_other = $input['dealing_other'];
$attendance = $input['attendance'];

// Check if a record for the student already exists
$stmt = $db->prepare("SELECT id FROM student_behavior WHERE s_id = ?");
$stmt->bind_param("i", $student_id);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
    // Record exists, update it
    $stmt->bind_result($behavior_id);
    $stmt->fetch();
    $stmt->close();
```

```
    $stmt = $db->prepare("UPDATE student_behavior SET dealing_teach = ?, dealing_other = ?, attendance = ? WHERE id = ?");
    $stmt->bind_param("iiii", $dealing_teach, $dealing_other, $attendance, $behavior_id);

    if ($stmt->execute()) {
        $response = array("status" => "success", "message" => "Student behavior updated successfully.");
    } else {
        $response = array("status" => "error", "message" => "Failed to update student behavior.");
    }
} else {
    // Record does not exist, insert it
    $stmt->close();

    $stmt = $db->prepare("INSERT INTO student_behavior (s_id, dealing_teach, dealing_other, attendance) VALUES (?, ?, ?, ?)");
    $stmt->bind_param("iiii", $student_id, $dealing_teach, $dealing_other, $attendance);

    if ($stmt->execute()) {
        $response = array("status" => "success", "message" => "Student behavior inserted successfully.");
    } else {
        $response = array("status" => "error", "message" => "Failed to insert student behavior.");
    }
}

$stmt->close();
$db->close();

header('Content-Type: application/json');
echo json_encode($response);
?>
```

Insert grade

```php
<?php
include 'database.php';

$input = json_decode(file_get_contents('php://input'), true);

$s_id = $input['s_id'];
$track_id = $input['track_id'];
if(isset($input['final'])){
    $grade = $input['final'];
}else if(isset($input['quiz'])){
    $grade = $input['quiz'];
}else if(isset($input['midterm'])){
    $grade = $input['midterm'];
}else if(isset($input['assignment'])){
    $grade = $input['assignment'];
}
$stmt = $db->prepare("SELECT id FROM student_grads WHERE s_id = ? AND track_id =
?");
$stmt->bind_param("ii", $s_id, $track_id);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
    $stmt->bind_result($grade_id);
    $stmt->fetch();
    if(isset($input['final'])){
        $stmt = $db->prepare("UPDATE student_grads SET final = ? WHERE id = ?");
    }else if(isset($input['quiz'])){
        $stmt = $db->prepare("UPDATE student_grads SET quiz = ? WHERE id = ?");
    }else if(isset($input['midterm'])){
        $stmt = $db->prepare("UPDATE student_grads SET midterm = ? WHERE id =
?");
    }else if(isset($input['assignment'])){
        $stmt = $db->prepare("UPDATE student_grads SET assignment = ? WHERE id =
?");
    }

    $stmt->bind_param("di", $grade, $grade_id);

    if ($stmt->execute()) {
        $response = array("status" => "success", "message" => "Grade updated
successfully.");
    } else {
```

```
        $response = array("status" => "error", "message" => "Failed to update
grade.");
    }
} else {
    // Insert new grade
    if(isset($input['final'])){
        $stmt = $db->prepare("INSERT INTO student_grads (s_id, track_id, final)
VALUES (?, ?, ?)");
    }else if(isset($input['quiz'])){
        $stmt = $db->prepare("INSERT INTO student_grads (s_id, track_id, quiz)
VALUES (?, ?, ?)");
    }else if(isset($input['midterm'])){
        $stmt = $db->prepare("INSERT INTO student_grads (s_id, track_id, midterm)
VALUES (?, ?, ?)");
    }else if(isset($input['assignment'])){
        $stmt = $db->prepare("INSERT INTO student_grads (s_id, track_id,
assignment) VALUES (?, ?, ?)");
    }
    $stmt->bind_param("iid", $s_id, $track_id, $grade);

    if ($stmt->execute()) {
        $response = array("status" => "success", "message" => "Grade inserted
successfully.");
    } else {
        $response = array("status" => "error", "message" => "Failed to insert
grade.");
    }
}

$stmt->close();
$db->close();

header('Content-Type: application/json');
echo json_encode($response);
?>
```

Insert notification

```php
<?php
include 'database.php';

$input = json_decode(file_get_contents('php://input'), true);

$assest_id = $input['assest_id'];
$student_id = $input['student_id'];
$message = $input['message'];
$track_id = $input['track_id'];

// Check if the track_id already exists in the database for the given student_id
$stmt_check = $db->prepare("SELECT * FROM notifications WHERE student_id = ? AND
track_id = ?");
$stmt_check->bind_param("ii", $student_id, $track_id);
$stmt_check->execute();
$result = $stmt_check->get_result();

if ($result->num_rows > 0) {
    // If the track_id exists for the given student_id, update the message
    $stmt_update = $db->prepare("UPDATE notifications SET message = ? WHERE
student_id = ? AND track_id = ?");
    $stmt_update->bind_param("sii", $message, $student_id, $track_id);

    if ($stmt_update->execute()) {
        $response = array("status" => "success", "message" => "Notification
updated successfully.");
    } else {
        $response = array("status" => "error", "message" => "Failed to update
notification.");
    }

    $stmt_update->close();
} else {
    // If the track_id doesn't exist for the given student_id, insert a new
record
    $stmt_insert = $db->prepare("INSERT INTO notifications (assest_id,
student_id, message, track_id) VALUES (?, ?, ?, ?)");
    $stmt_insert->bind_param("iisi", $assest_id, $student_id, $message,
$track_id);

    if ($stmt_insert->execute()) {
        $response = array("status" => "success", "message" => "Notification
inserted successfully.");
```

```
    } else {
        $response = array("status" => "error", "message" => "Failed to insert
notification.");
    }

    $stmt_insert->close();
}

$stmt_check->close();
$db->close();
echo json_encode($response);
?>
```

Insert student

```
<?php
include 'database.php';

// Retrieve input data and decode JSON
$input = json_decode(file_get_contents('php://input'), true);

// Check if input data is valid
if (is_null($input) || !is_array($input)) {
    $response = array("status" => "error", "message" => "Invalid input data.");
    echo json_encode($response);
    exit;
}

// Validate required fields
$required_fields = ['name', 'phone', 'email', 'password', 's_number'];
foreach ($required_fields as $field) {
    if (!isset($input[$field]) || empty($input[$field])) {
        $response = array("status" => "error", "message" => "Missing required
field: $field.");
        echo json_encode($response);
        exit;
    }
}

// Retrieve data from input array
$name = $input['name'];
$phone = $input['phone'];
```

```
$email = $input['email'];
$password = $input['password'];
$s_number = $input['s_number'];

// Check if email already exists
$stmt = $db->prepare("SELECT id FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
    $response = array("status" => "error", "message" => "Email already exists.");
} else {
    // Insert student data
    $stmt = $db->prepare("INSERT INTO students (name, phone, email, s_number)
VALUES (?, ?, ?, ?)");
    $stmt->bind_param("ssss", $name, $phone, $email, $s_number);

    if ($stmt->execute()) {
        $student_id = $stmt->insert_id;
        $hashed_password = password_hash($password, PASSWORD_DEFAULT);
        $type='student';

        // Insert user data
        $stmt = $db->prepare("INSERT INTO users (s_id, name, email, password,
type) VALUES (?, ?, ?, ?, ?)");
        $stmt->bind_param("issss", $student_id, $name, $email, $hashed_password,
$type);

        if ($stmt->execute()) {
            $response = array("status" => "success", "message" => "Student and
user data inserted successfully.");
        } else {
            $response = array("status" => "error", "message" => "Failed to insert
user data.");
        }
    } else {
        $response = array("status" => "error", "message" => "Failed to insert
student data.");
    }
}

$stmt->close();
$db->close();
echo json_encode($response);
```

```
?>
```

Insert student behavior

```
<?php
include 'database.php';

$input = json_decode(file_get_contents('php://input'), true);

$dealing_teach = $input['dealing_teach'];
$student_id = intval($input['student_id']); // Convert to integer
$dealing_other = $input['dealing_other'];
$attendance = $input['attendance'];

// Check if the student_id already exists in the student_behavior table
$check_stmt = $db->prepare("SELECT s_id FROM student_behavior WHERE s_id = ?");
$check_stmt->bind_param("i", $student_id);
$check_stmt->execute();
$check_stmt->store_result();

if ($check_stmt->num_rows > 0) {
    // Student exists, update the record
    $update_stmt = $db->prepare("UPDATE student_behavior SET dealing_teach = ?,
dealing_other = ?, attendance = ? WHERE s_id = ?");
    $update_stmt->bind_param("iiii", $dealing_teach, $dealing_other, $attendance,
$student_id);

    if ($update_stmt->execute()) {
        $response = array("status" => "success", "message" => "student_behavior
updated successfully.");
    } else {
        $response = array("status" => "error", "message" => "Failed to update
student_behavior data.");
    }

    $update_stmt->close();
} else {
    // Student does not exist, insert a new record
    $insert_stmt = $db->prepare("INSERT INTO student_behavior (s_id,
dealing_teach, dealing_other, attendance) VALUES (?, ?, ?, ?)");
    $insert_stmt->bind_param("iiii", $student_id, $dealing_teach, $dealing_other,
$attendance);
```

```
    if ($insert_stmt->execute()) {
        $response = array("status" => "success", "message" => "student_behavior
inserted successfully.");
    } else {
        $response = array("status" => "error", "message" => "Failed to insert
student_behavior data.");
    }

    $insert_stmt->close();
}

$check_stmt->close();
$db->close();
echo json_encode($response);
?>
```

Login

```
<?php
include 'database.php';
$input = json_decode(file_get_contents('php://input'), true);

if (isset($input['email']) && isset($input['password'])) {
    $email = $input['email'];
    $password = $input['password'];

    // Prepare the SQL statement to prevent SQL injection
    $stmt = $db->prepare("
        SELECT u.id AS user_id, u.password, u.type, u.name, s.id AS student_id
        FROM users u
        LEFT JOIN students s ON u.email = s.email  -- Adjust this line based on
the actual relationship between users and students
        WHERE u.email = ? AND u.state = 1
    ");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        if (password_verify($password, $row['password'])) {
```

```
            echo json_encode(array(
                "success" => true,
                "message" => "Login successful",
                "user_id" => $row['user_id'],
                "student_id" => $row['student_id'],  // Return the student_id
                "type" => $row['type'],
                "name" => $row['name']
            ));
        } else {
            echo json_encode(array("success" => false, "message" => "Invalid
password"));
        }
    } else {
        echo json_encode(array("success" => false, "message" => "No user found
with that email"));
    }

    $stmt->close();
} else {
    echo json_encode(array("success" => false, "message" => "Email and password
required"));
}

$db->close();
?>
```

Student grade

```
<?php
include 'database.php';

$input = json_decode(file_get_contents('php://input'), true);
$students = $input['students'];
$gradeType = $input['gradeType'];

foreach ($students as $student) {
    $studentName = $student['student_name'];
    $studentNumber = $student['student_number'];
    $grade = $student[$gradeType];

    // Fetch student ID based on student number
    $stmt = $db->prepare("SELECT id FROM students WHERE s_number = ?");
```

```
    $stmt->bind_param("s", $studentNumber);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        $stmt->bind_result($student_id);
        $stmt->fetch();

        // Check if the grade already exists for the student in the specified
track
        $stmt2 = $db->prepare("SELECT id FROM student_grads WHERE s_id = ? AND
track_id = ? AND state = 1");
        $stmt2->bind_param("ii", $student_id, $track_id);
        $stmt2->execute();
        $stmt2->store_result();

        if ($stmt2->num_rows > 0) {
            // Update existing grade
            $stmt2->bind_result($grade_id);
            $stmt2->fetch();

            $stmt3 = $db->prepare("UPDATE student_grads SET $gradeType = ? WHERE
id = ?");
            $stmt3->bind_param("si", $grade, $grade_id);
            $stmt3->execute();
        } else {
            // Insert new grade record
            $stmt3 = $db->prepare("INSERT INTO student_grads (s_id, track_id,
$gradeType, state) VALUES (?, ?, ?, 1)");
            $stmt3->bind_param("iis", $student_id, $track_id, $grade);
            $stmt3->execute();
        }

        $stmt2->close();
    }
}

$stmt->close();
$db->close();

$response = array("status" => "success", "message" => "Grades data processed
successfully.");
echo json_encode($response);
?>
```

Conclusion

In conclusion, the development and implementation of the Student Evaluation System marks a significant advancement in how educational institutions can manage and analyze student performance data. This system offers a streamlined, efficient, and user-friendly approach to tracking academic progress, identifying areas of improvement, and making informed decisions to enhance the overall learning experience.

Key benefits of the Student Evaluation System include:

- **Enhanced Data Management**: Centralized storage of student records, grades, and evaluations reduces administrative burden and minimizes errors.
- **Real-time Insights**: Immediate access to performance metrics allows educators to quickly identify and address learning gaps.
- **Improved Communication**: Facilitates better communication between students, teachers, and parents through timely updates and transparent reporting.
- **Customized Feedback**: Enables personalized feedback tailored to individual student needs, promoting targeted interventions and support.
- **Data-Driven Decisions**: Empowers educators with robust analytics tools to make evidence-based decisions aimed at improving educational outcomes.

By integrating modern technology into the evaluation process, the Student Evaluation System not only enhances operational efficiency but also contributes to a more personalized and effective educational environment. As a result, students receive the support they need to succeed, teachers can focus more on teaching, and administrators gain valuable insights to shape future educational strategies.

Moving forward, continuous updates and enhancements to the system, based on user feedback and evolving educational needs, will ensure that it remains a vital tool in fostering academic excellence and innovation in education.