بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

An-Najah National University
Faculty of Engineering and Information Technology
Computer Engineering Department

Software Graduation Project

An Najah Rank

Prepared by:
-Momen Odeh *-Noor Aldeen Abu Shehadeh*
Supervised by:
Dr. Samer Arandi

Submitted on:
22th, January 2024

Presented in partial fulfillment of the requirements for Bachelor degree in Computer Engineering.

Dedication

Dedication to loving memory of our grandmother, our loving parents, family, friends and for everyone who believed and loved us.

Acknowledgment

We extend our deepest gratitude and appreciation to the individuals who have played a significant role in our graduation project. Their guidance, support, and unwavering belief in our abilities have been invaluable throughout this journey.

*We would like to **thank our supervisor Dr. Samer Arandi** a lot for his helpful, kind, patience and taking care of us, and for making everything simple. He was always inspiring and encouraging us to move.*

*We would also like to **thank all the teachers and teacher's assistant in the Department of Computer Engineering**, and we feel proud to be students in it, as this helps us to improve our educational level as well as improve our skills.*

Disclaimer

This report was written by students at the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the students. An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.

Table of content

List of Figures

List of tables

Abstract

One of the most important skills for any programmer is problem-solving skills, and there are many websites that can be used to train these skills, such as HackerRank, Codeforces, LeetCode, etc.

At An-Najah National University, professor always strive to improve students' problem- solving skills in many subjects such as computer programming, data structures, algorithms, and object-oriented programming by assigning problem-solving assignments and quizzes using problem-solving websites. However, they face several challenges in using these websites, such as difficulty tracking student submissions, an inability to directly identify code similarities among students' submissions, and the inability to manually mark incorrect answers.

We built this project by creating a web application with React JS as the frontend and Flask Python as the backend. We used Docker to containerize the application, allowing easy deployment on the cloud or any local server. Additionally, we leveraged several services from Amazon Web Services (AWS), including S3 for storage, RDS for the MySQL database engine, and EC2 for deploying the web application.

Chapter 1: Introduction

- **Statement of the problem**

The problem-solving skills are one of the most important skills in the workplace, so An-Najah National University strives to improve these skills in our students by incorporating problem-solving tasks into many courses using external problem-solving websites. However, these websites lack essential features that would simplify the problem-solving process and make solution grading more efficient. This emphasizes the increasing importance of a web application to address all these challenges.

One of the primary challenges lies in the difficulty of efficiently tracking and managing student submissions. This hinders the seamless monitoring of individual progress and the timely assessment of assignments. Additionally, there is a limitation in directly identifying code similarities among the submissions, making it challenging to address potential collaboration or plagiarism issues effectively.

Another notable challenge is the absence of a streamlined mechanism for manually marking incorrect answers. This deficiency impedes the ability of professors to provide targeted feedback, hindering the learning process for students.

▪ Objective

The purpose of our work is to create a web application for problem-solving that is easy to use for both students and professors. We aim to achieve this by incorporating new features not available in other problem-solving web applications. The objectives of our work are as follow:

- Registration and login for both students, professors and admin on the web application.

- Professors can create new courses and enroll students in them by simply uploading the excel file exported from any zajel course.

- Professors can add contests to their courses. For each contest, the professor provides challenges, and each challenge should have a set of input test cases along with the expected correct output. The system will automatically evaluate the challenges based on the provided output test cases. Additionally, each contest has a designated starting and ending date, during which it will be available to the students.

- Professors can view a list of students who have submitted challenge, their grades, and the similarity of their submissions. They can also review the submissions and optionally manually mark last submission that was found incorrect by the system.

- The professor can also track the progress of the student submission, i.e. they can see the changes from the first version the students submit to the last (hopefully) correct answer.

- Students can access their homepage on the system which shows information about the assignments and quizzes in current or previous courses. The student can start solving the assignments/quizzes assigned to them by writing code in their preferred programming language, such as C, C++, Java, Python, JavaScript, or REGX directly in the browser. They can then run the code to check if it passes or fails test cases.

- The student can also view the status for each assignment/quiz, their score and general performance.

- User receive notification when a related event occurs.

- Any user can make chatting with other user.

Scope of the work

- **Frontend using React JS:** We developed the frontend using React JS, building the user interface with the React JS library, utilizing React Bootstrap as the UI kit, React Router, and incorporating React-JSS for styled components.

- **Backend using Flask python:** We developed the backend using Flask python microservices framework.

- **Database using MySQL:** We chose the MySQL database because our data is relational. Subsequently, we generated the tables using MySQL Workbench.

- **Amazon Web Services (AWS):** We utilize various services from AWS, using the S3 service for storing files and images, the RDS service for the MySQL database engine, and the EC2 service for deploying the web application.

- **DevOps:** We generated a portable copy of our project that can be easily deployed on any device using Docker and Docker Compose technologies.

- **Testing:** After building our project, we conducted manual tests to ensure that everything worked correctly.

Importance of the work

The An-Najah Rank web application has many features that enhance usability and includes new functionalities. Here are the reasons that explain why this web application is important:

- **Easy to use:** The web application is user-friendly for all users, including admin, professors, and students. And that appear in simplicity of user interface.

- **Check plagiarism:** We have added a 'calculate similarity' feature that can check the similarity between student code submissions.

- **Show all submissions:** We can easily to traversing student submissions by viewing all last submissions of students in one place and can traverse all submissions on any student easily.

- **Manual Marking:** We have added a manual marking feature that allows professor to remarking the last submission of any student.

- **Flexibility of test cases:** The professor can adjust the final grade of challenge by adding new test case that will automatically run the new test case on the last submission code and adjust the final grade based on all results.

- **This web application is implemented specifically for educational use:** We have customized many features for this purpose, such as limiting the programming languages that can be used to solve the challenge. Professors can easily add students by uploading an Excel sheet.

▪ Organization of the report

The report is structured in a logical and systematic manner to effectively present the information related to the project. The organization of the report is as follows:

- **Introduction:** This section provides an overview of the project, highlighting the problem statement, objectives, and the importance of the work. It sets the context for the rest of the report.

- **Theoretical Background and Previous Work:** In Chapter 2, It presents a comprehensive review of existing research, studies, and relevant literature related to An Najah Rank, automation techniques, and similar projects. This section helps establish the project's context and highlights any gaps in the existing knowledge.

- **Methodology:** Chapter 3 explains the materials and methods used throughout the project. It provides a detailed description of the experimental setup, the Web application development process. The chapter outlines the steps taken to achieve the project objectives.

- **Results and Analysis:** Chapter 4 presents the results obtained from the project. It includes the outcomes of the process using the An Najah Rank web application, as well as any relevant data or measurements. The results are analyzed and interpreted to draw meaningful conclusions.

- **Discussion:** Chapter 5 focuses on the discussion of the results. It provides a comprehensive analysis of the findings, highlighting the features, benefits, and limitations of the An Najah Rank web application. The chapter also addresses any challenges faced during the project and offers recommendations for future improvements.

- **Conclusion and Recommendation:** chapter6 concludes report by summarizing the key findings, reiterating the significance of the work, and highlighting its potential impact. It may also include a reflection on the overall project experience and suggestions for further research.

- **References:** A list of all the references cited throughout the report is provided in the References section, following the conclusion.

Chapter 2: Theoretical Background and Previous Work

These days, there are many problem-solving web applications, such as LeetCode, CodeForces, and HackerRank. However, these web applications are not completely suitable for educational purposes. Therefore, we built a problem-solving web application that combines solving problems for students and adds the educational features needed for professors, making the process more straightforward.

Chapter 3: Methodology

In our project, we diligently adhere to the Software Development Life Cycle (SDLC), a systematic approach that enables cost-effective and time-efficient software development. SDLC guides our development teams through essential stages such as planning, design, development, testing, deployment, and maintenance. This structured process not only aims to design and build high-quality software but also minimizes project risks through forward planning. By following SDLC, we ensure that the software meets customer expectations during production and beyond, contributing to the overall success and reliability of our projects.

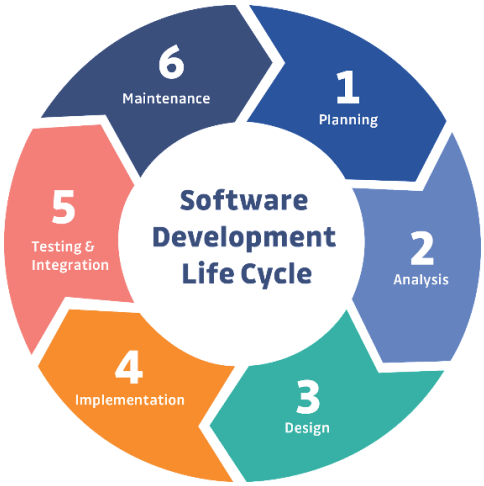


Figure 1: Software Development Life Cycle

In our project, we have embraced the Agile methodology as the guiding framework for our software development process. Agile is a dynamic and iterative approach that prioritizes flexibility, collaboration, and customer satisfaction. Unlike traditional linear models, Agile promotes adaptability to changing requirements and a continuous feedback loop, allowing us to respond promptly to evolving project needs.

Figure 2: Agile Methodology

3.1 Planning:

We met with our client, Dr. Samer Arandi, to discuss the project features and decide which ones will be implemented. During our meeting, we explored various problem-solving websites to gain insights and ideas for the project.

Our collaboration extended beyond the existing features as we explored new functionalities to enhance the project. This discussion not only provided a clearer vision for the project but also facilitated the identification of potential innovative features to meet both current and future user needs.

3.2 Analysis:

In the initial phase of our software project, thorough analysis was conducted to gather and document project requirements through stakeholder engagement and user feedback sessions. This process involved crafting user stories to delineate specific functionalities and envisioning the system's architecture through Unified Modeling Language (UML) diagrams.

3.2.1 UML Diagram:

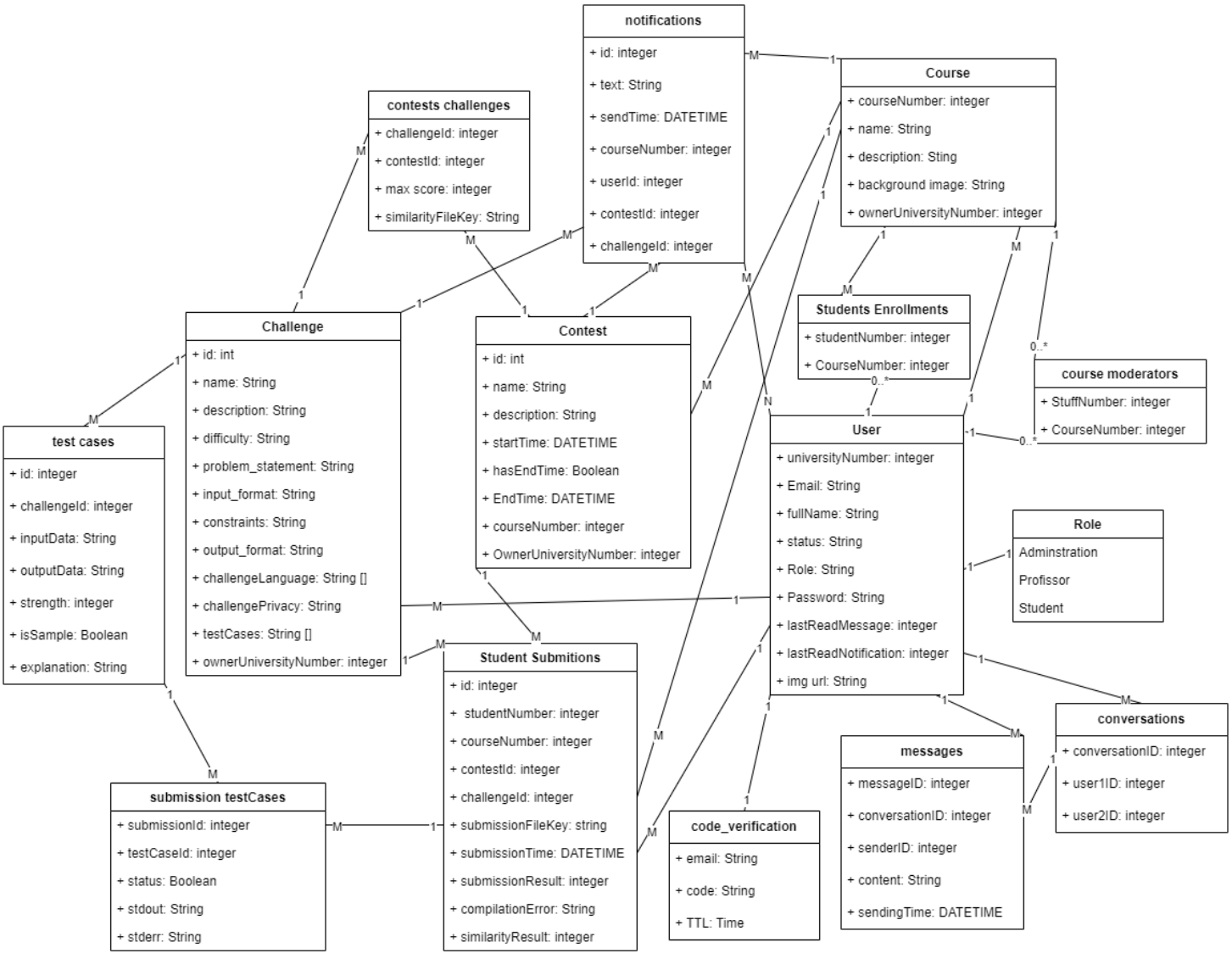


Figure 3: UML Diagram

3.2.2 User Stories:

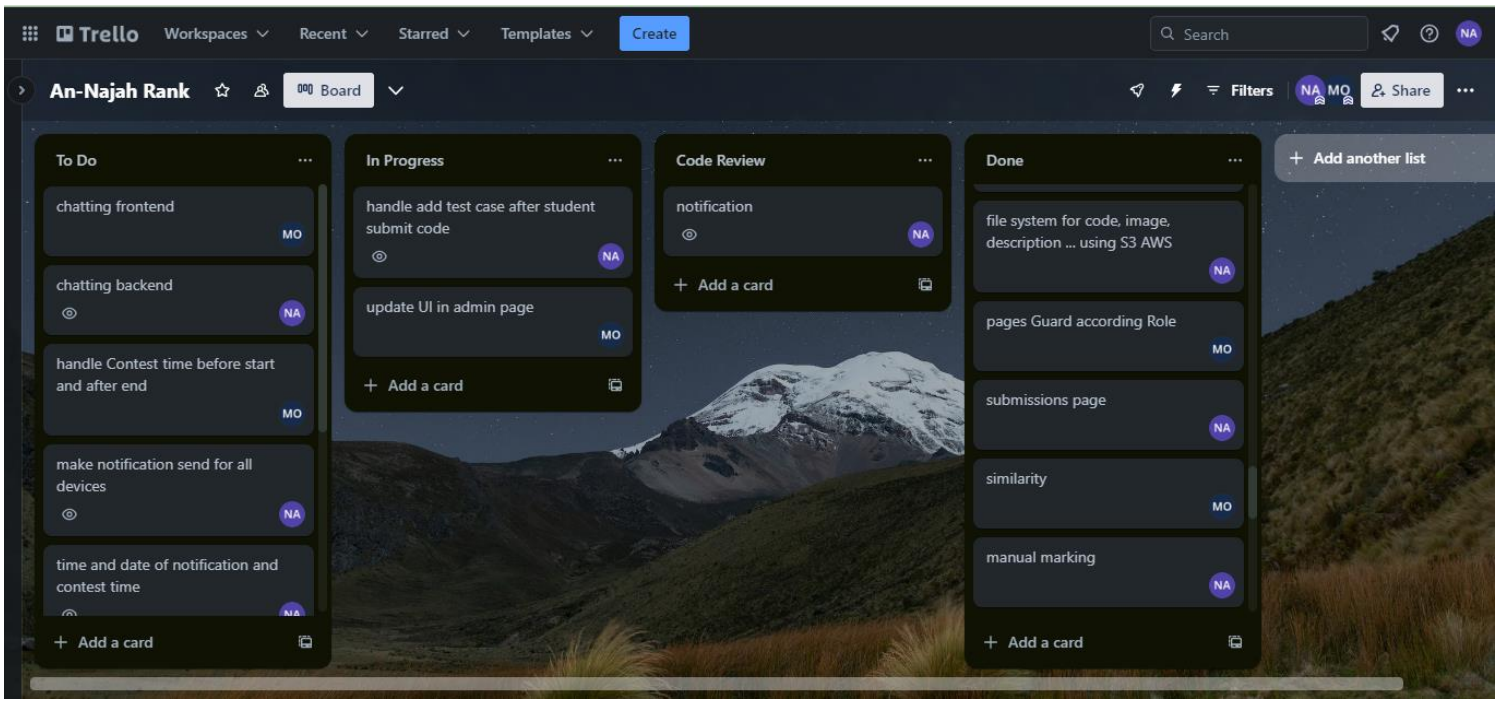


Figure 4: User Stories

3.3 Design:

3.3.1 Tools:

3.3.1.1 Frontend tools:

3.3.1.1.1 React JS:

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It makes it easy to compose complex UIs from small and isolated pieces of code called components.

In our project we used ReactJS as the front-end technology due to the easiness of learning, rich set of user-interface, community support, and the fast development of software. In addition, it offers the capability to reuse already built components.

Figure 5: React

3.3.1.1.2 React Bootstrap:

This UI kit contains many ready components that can be used directly with some customization for style. Additionally, this UI kit provides components that can make the design responsive easily.

Figure 6: React Bootstrap

3.3.1.1.3 React JSS:

Is a library that enables styling React components using JavaScript. Providing powerful features such as:

- Dynamic Theming - allows context-based theme propagation and runtime updates.

- Function values and rules are updated automatically with any data that passed as probs.

Figure 7: React JSS

3.3.1.1.4 React Router:

React Router enables "client side routing".

Client side routing allows your app to update the URL from a link click without making another request for another document from the server. Instead, your app can immediately render some new UI and make data requests with fetch to update the page with new information.

This enables faster user experiences because the browser doesn't need to request an entirely new document or re-evaluate CSS and JavaScript assets for the next page. It also enables more dynamic user experiences with things like animation.

Figure 8: React Router

3.3.1.2 Backend tools:

3.3.1.2.1 Flask python:

Flask is a lightweight and user-friendly Python web framework that streamlines backend development. While originally designed for simplicity, Flask proves versatile for building microservices. It provides a simple way to create and deploy dynamic web applications; it enables developers to focus on the application logic rather than worrying about the underlying infrastructure. Moreover, it offers a great deal of freedom and control over application development. Its integration with Python libraries and technologies makes it easy to integrate with a wide variety of software development tools and solutions.

Figure 9: Flask python

3.3.1.2.2 Pandas:

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Figure 10: Pandas

3.3.1.2.3 SocketIO:

Flask-SocketIO is an extension for Flask that facilitates low-latency, bidirectional communication between the server and clients using WebSockets. It allows real-time, interactive features to be implemented in Flask applications by enabling seamless communication between the server and connected clients.

Figure 11: SocketIO

3.3.1.3 DevOps tools:

3.3.1.3.1 GitHub:

Git is open-source version control software, used for managing and tracking file revisions. You can use Git with any file type, but it's most often used for tracking code files.

GitHub is an online software development platform. It's used for storing, tracking, and collaborating on software projects.

Figure 12: GitHub

3.3.1.3.2 Trello:

Trello is the visual tool that empowers your team to manage any type of project, workflow, or task tracking. Add files, checklists, or even automation: Customize it all for how your team works best.

Figure 13: Trello

3.3.1.3.3 Docker:

Docker is a software platform that uses OS-level virtualization to deliver software in packages called containers. It allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that contain everything the software needs to run, including libraries, system tools, code, and runtime. By using Docker, you can quickly deploy and scale applications into any environment and be confident that your code will run.

Figure 14: Docker

3.3.1.3.4 Docker Compose:

Compose is a tool for defining and running multi-container Docker applications. With Compose, we use a YAML file to configure the application's services. Then, with a single command, you can create and start all the services from your configuration.

Figure 15: Docker Compose

3.3.1.3 AWS CloudFormation:

AWS CloudFormation is Amazon Web Services' (AWS) native IaC tool. It enables you to define infrastructure resources using YAML or JSON templates, ensuring automation and consistent deployments in the AWS environment.

Figure 16: AWS CloudFormation

3.3.1.4.1 AWS EC2:

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster.

Figure 17: AWS EC2

3.3.1.4.2 AWS S3:

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Figure 18: AWS S3

3.3.1.4.3 AWS RDS:

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.

Figure 19: AWS RDS

3.3.1.5 Development tools:

3.3.1.5.1 VS Code:

Used for React development.

Figure 20: VS Code

3.3.1.5.2 pycharm:

Used for Flask development.

Figure 21: pycharm

3.3.1.5.3 MySQL Workbench:

Used for building and monitoring database.

Figure 22: MySQL Workbench

3.3.1.5.4 Postman:

Used for test backend APIs.

Figure 23: Postman

3.3.1.5.5 Docker Desktop:

Used for managing images and containers.

Figure 24: Docker Desktop

3.3.1.5.6 Draw io:

Used for design UML diagram.

Figure 25: Draw io

3.3.2 Architecture:

3.3.2.1 Architectural Style:

We used **RESTful** architectural style, which is a design approach for networked applications prioritizing simplicity, scalability, and loose coupling. It utilizes a stateless client-server model with principles such as statelessness, a uniform interface, and resource-based interactions. Key advantages encompass simplicity, scalability, and a consistent interface.

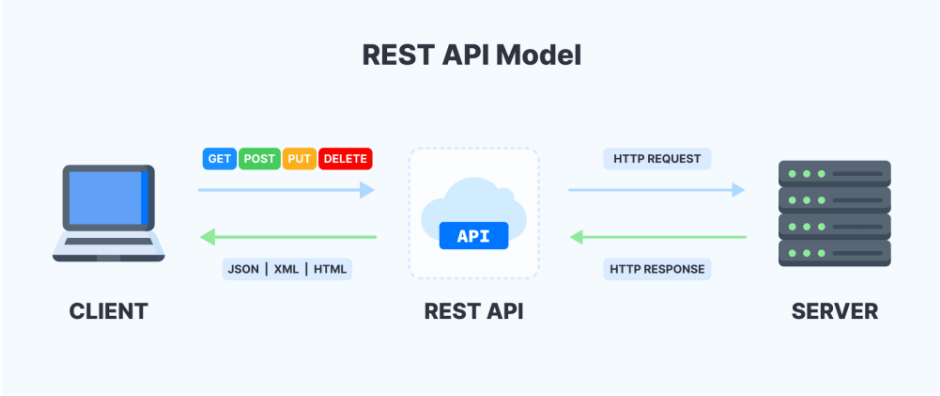


Figure 26: Restful Architectural Style

To send requests from the frontend to the backend, **Axios**, a popular JavaScript library, is commonly used in React applications. Axios simplifies the process of making asynchronous HTTP requests to external resources, particularly APIs. It is favored for its simplicity, flexibility, and notable features, including automatic JSON data transformation in responses.

Figure 27: AXIOS library

3.3.2.2 Architectural Pattern:

We used Microservices architectural pattern, which is particularly beneficial for large and complex applications where different functionalities can be developed and maintained independently.

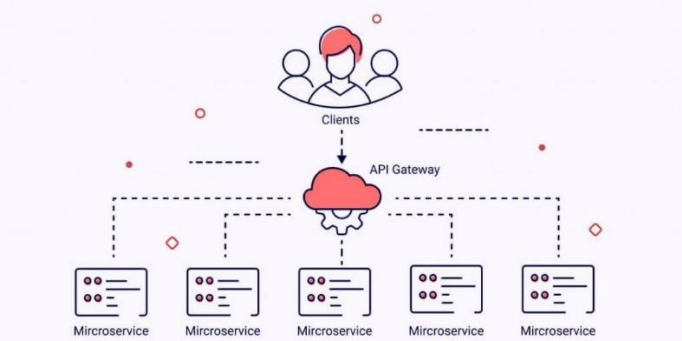


Figure 28: Microservice Architectural pattern

3.3.2.3 Project Structure:

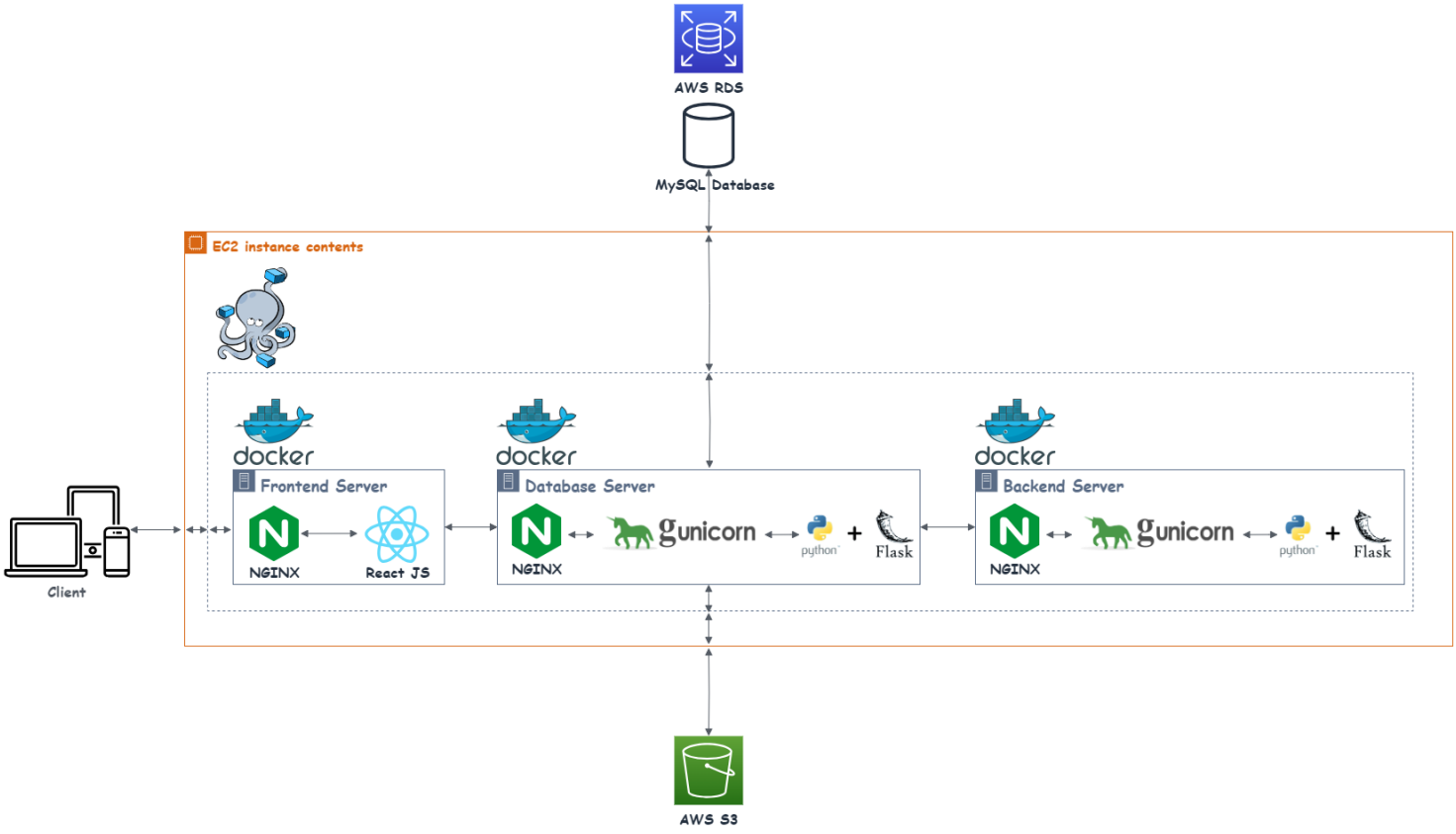


Figure 29: Project Structure

We divide the project into 3 containers:

1- Frontend container: Handles client requests and returns the UI to the client.
2- Database container: Manages requests from the Frontend container. If the request is related to code operations, it passes the request to the Backend container and returns the response to the Frontend container.
3- Backend container: Handles code operation requests, such as compiling and running code.

3.3.3 Security:

3.3.3.1 Authentication:

To use the web application, you must have an account. To obtain one, you need to register on the system and confirm your registration by entering the valid verification code received via email. When a user log in into the system, we authenticate their information. If the authentication is successful, we generate a token and return it to the frontend.

3.3.3.2 Autherization:

After logging in, each request to the backend should include a token. In the backend, the system first checks the validity of the token. If the token is valid, it is passed to the API; otherwise, an unauthorized response is returned. Upon receiving a request, the API checks the user's role, which is extracted from the token. If the user has the necessary access rights to the API, the request is processed; otherwise, an unauthorized response is returned.

3.3.3.3 CORS policies:

In the backend, we enable the CORS policy for the frontend address, so any received request from another address will be rejected.

3.3.3.4 : Library used:

Figure 30: Flask Mai

Figure 31: pyJWT

Figure 32: Flask-Cors

3.4 Implementation:

3.4.1 User Features:

3.4.1.1 Registration:

After entering their information, the user can choose to sign up as a professor. Subsequently, upon email verification, their request will appear on the admin page for acceptance or rejection. If the request is accepted, the user can log into the system; otherwise, they will not be allowed to access the system. For non-professor accounts, after email verification, users can log into the system.

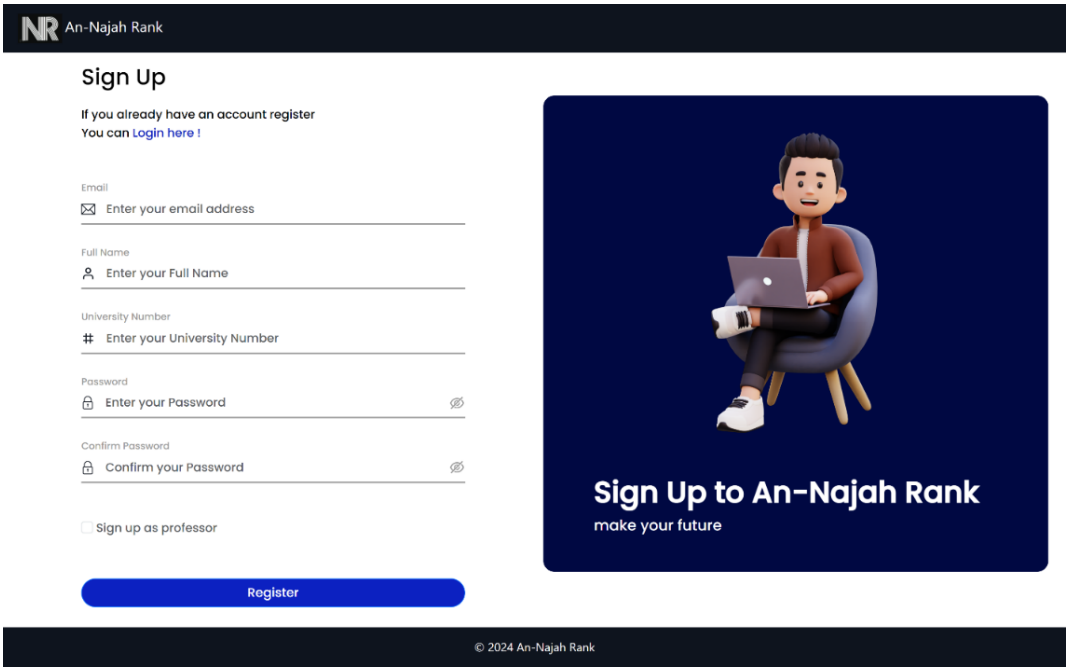


Figure 33: sign up

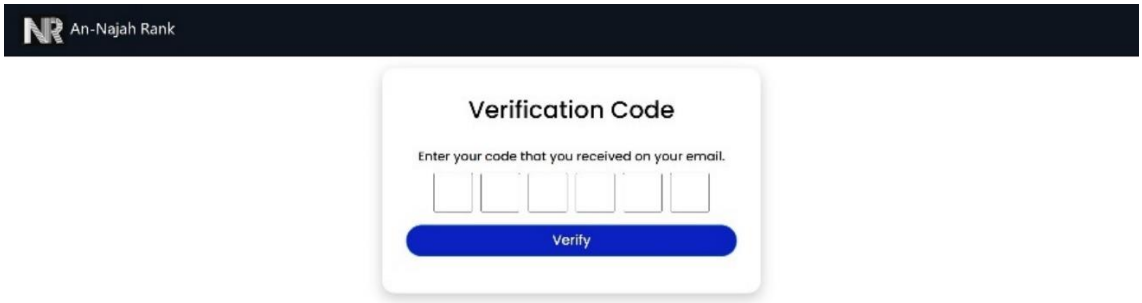


Figure 34: verification code

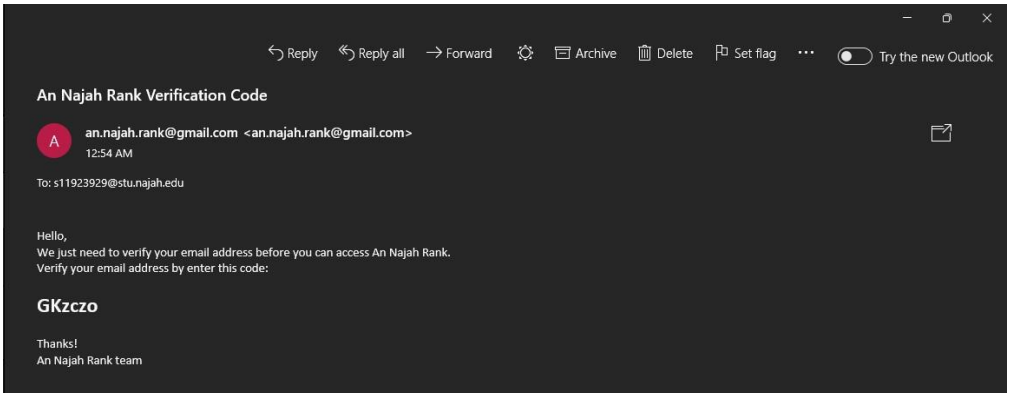


Figure 35: email verification message

3.4.1.2 Sign in:

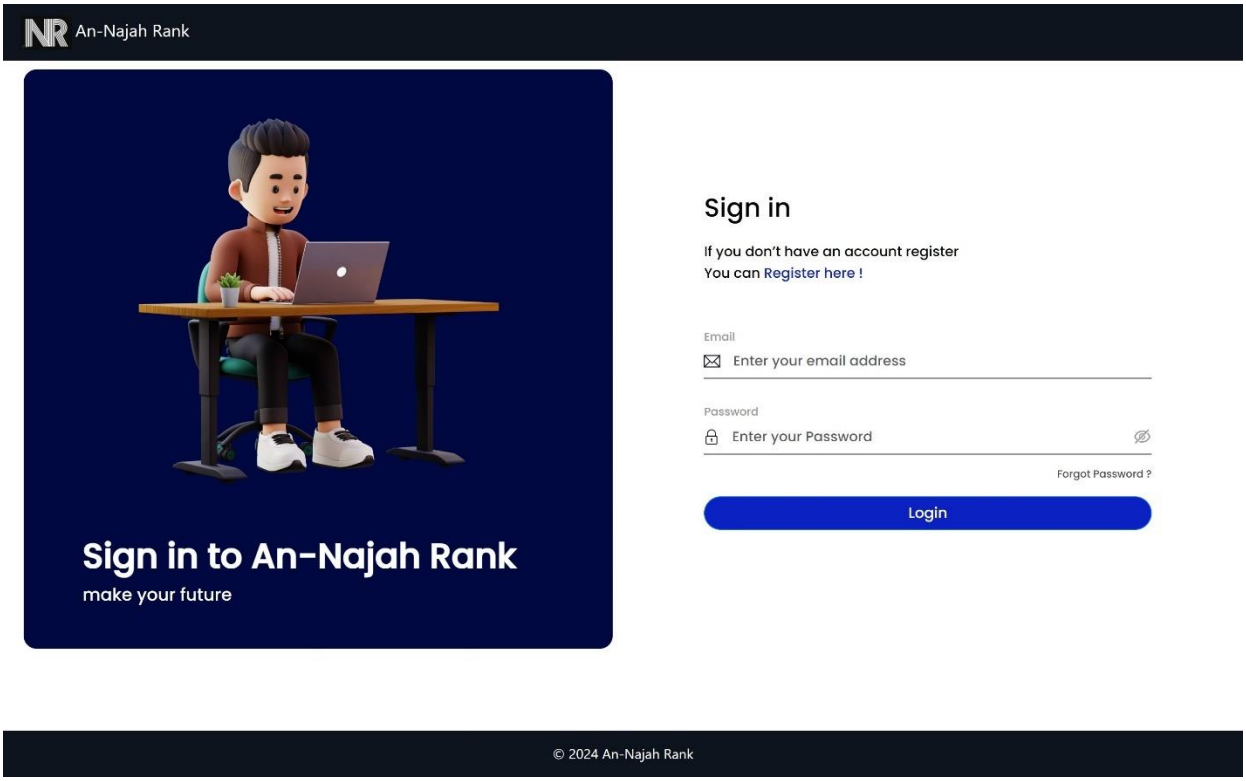


Figure 36: log in

3.4.1.3 Forget password:

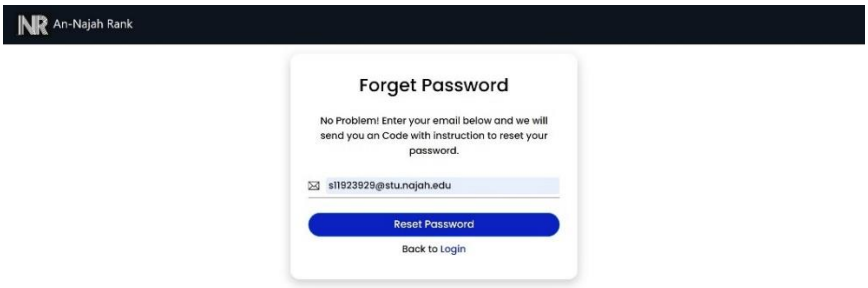


Figure 37: forget password

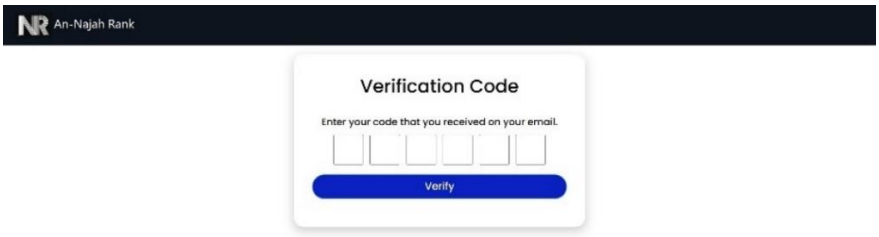


Figure 38: verification code for reset password

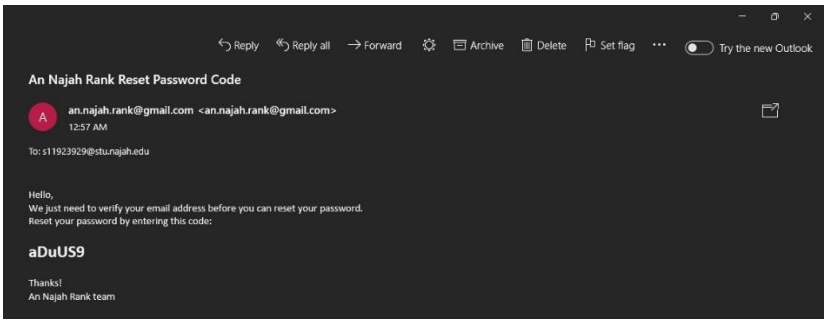


Figure 39: verification code message for reset password

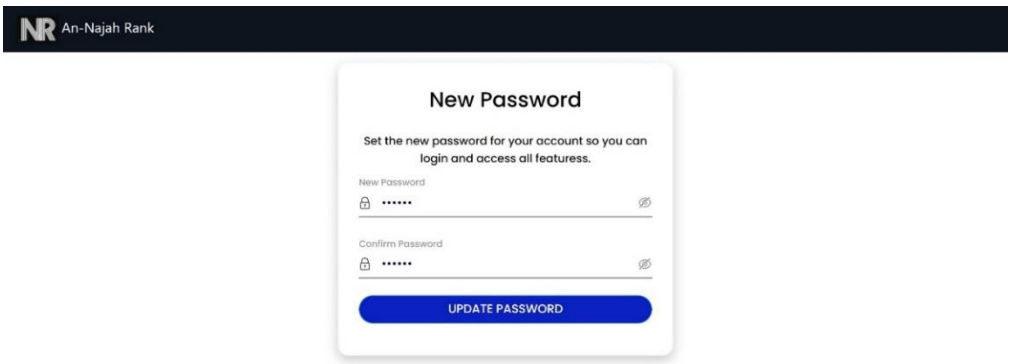


Figure 40: set new password

3.4.1.4 Account Settings:

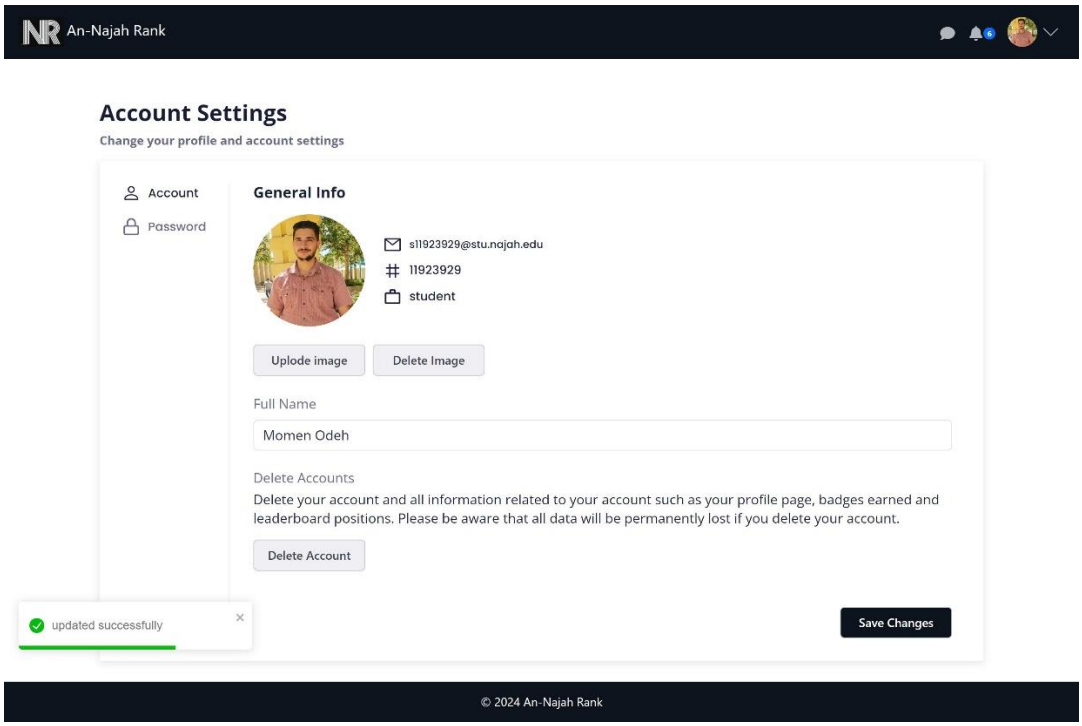


Figure 41: User account settings

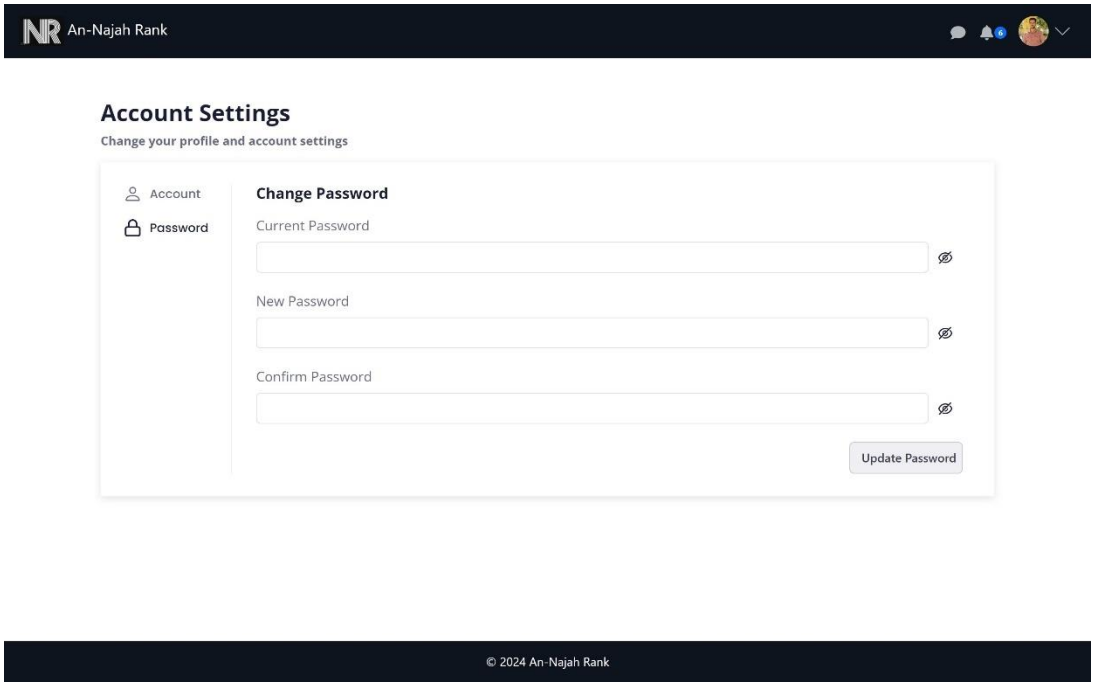


Figure 42: Password settings

3.4.1.5 Chatting:

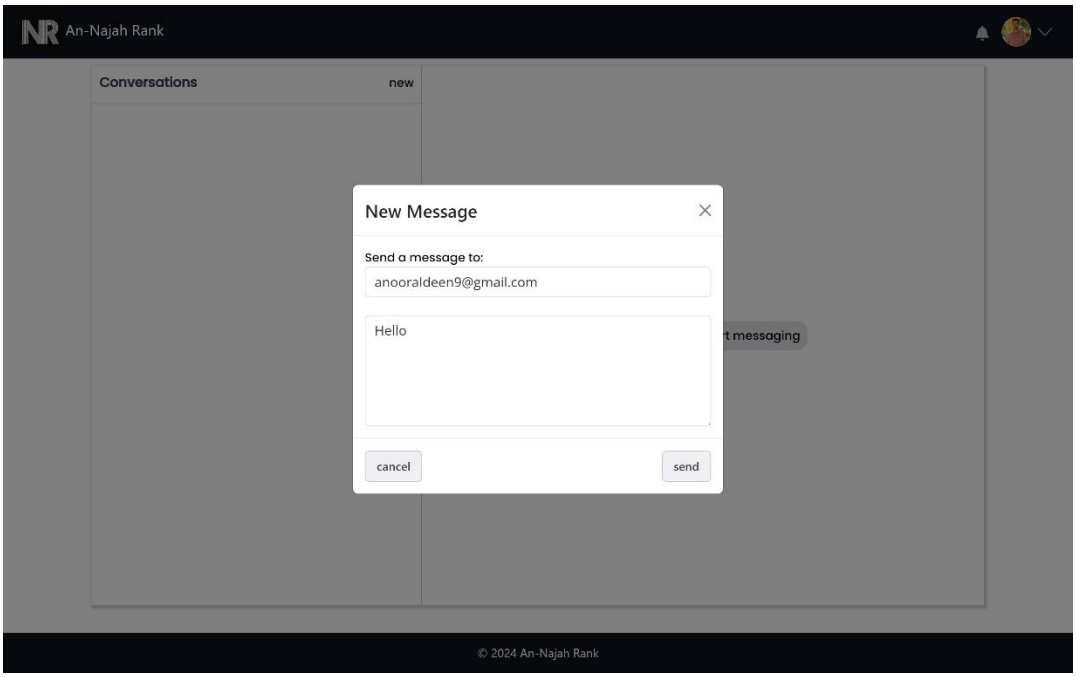


Figure 43: Create new Chat

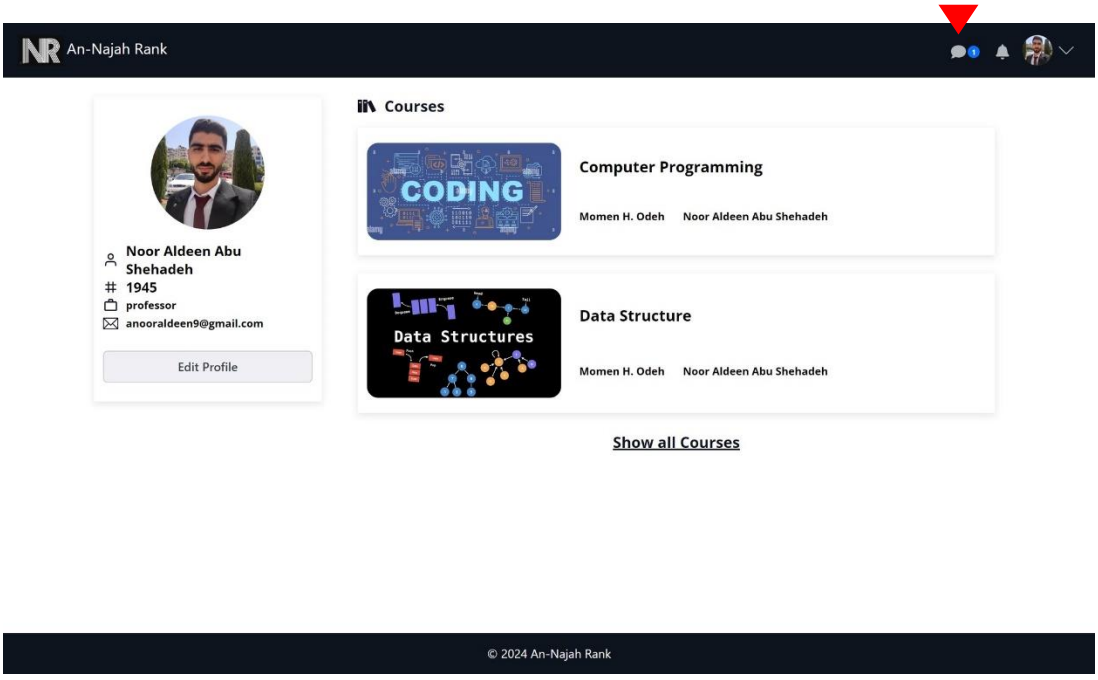


Figure 44: Chatting notification

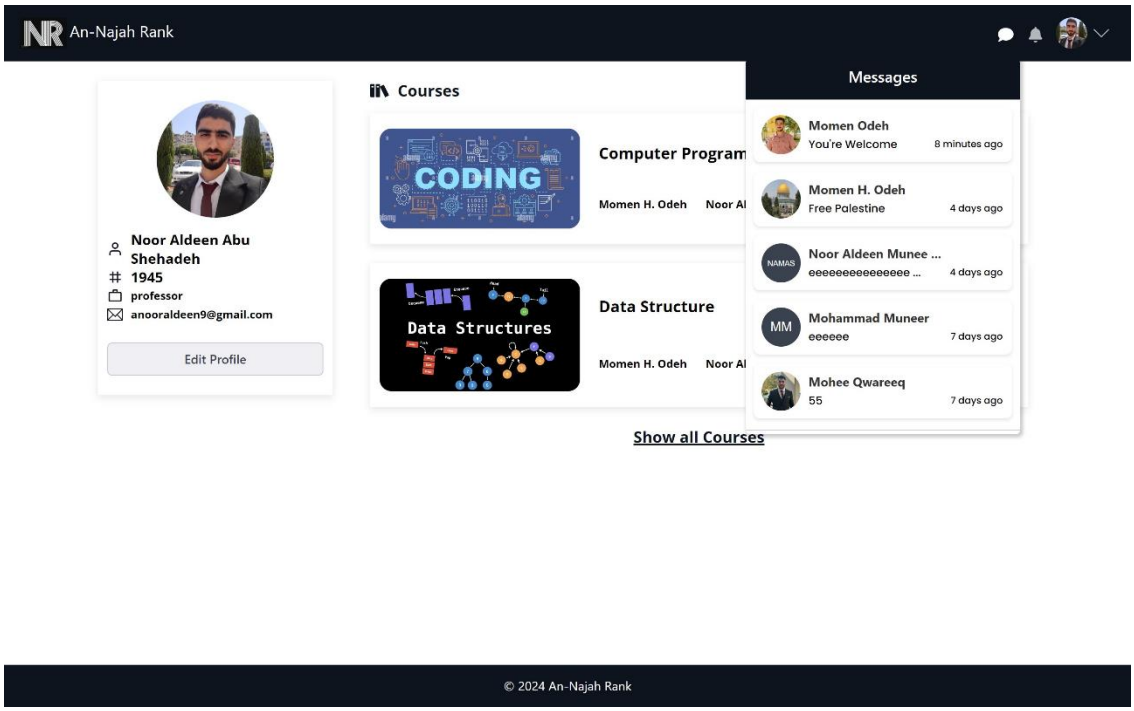


Figure 45: See chatting notification pop-up

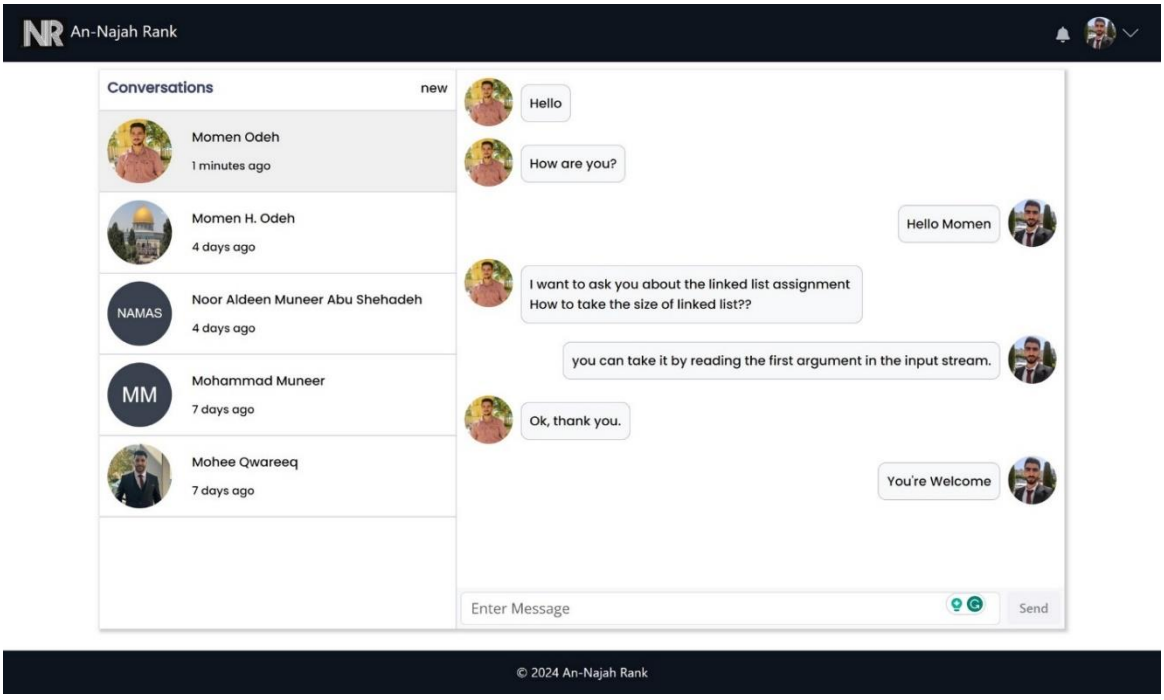


Figure 46: Chatting conversations

3.4.2 Admin features:

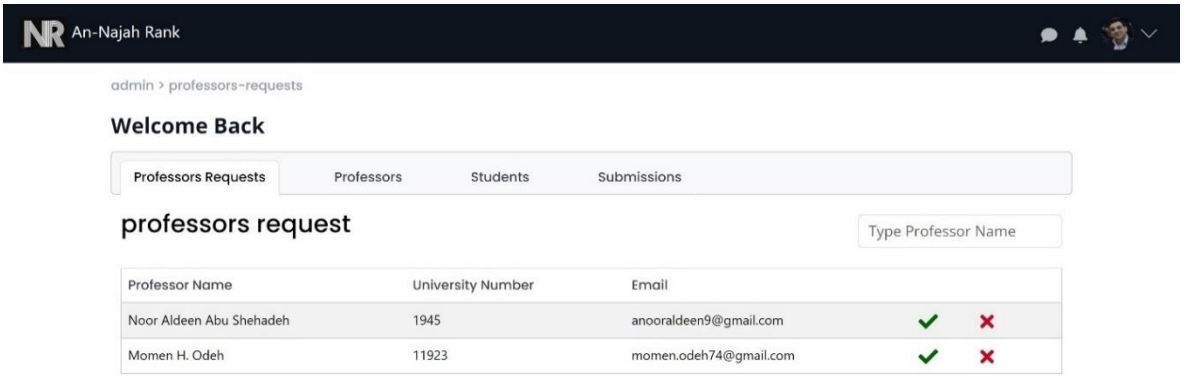


Figure 47: pending professors in admin page

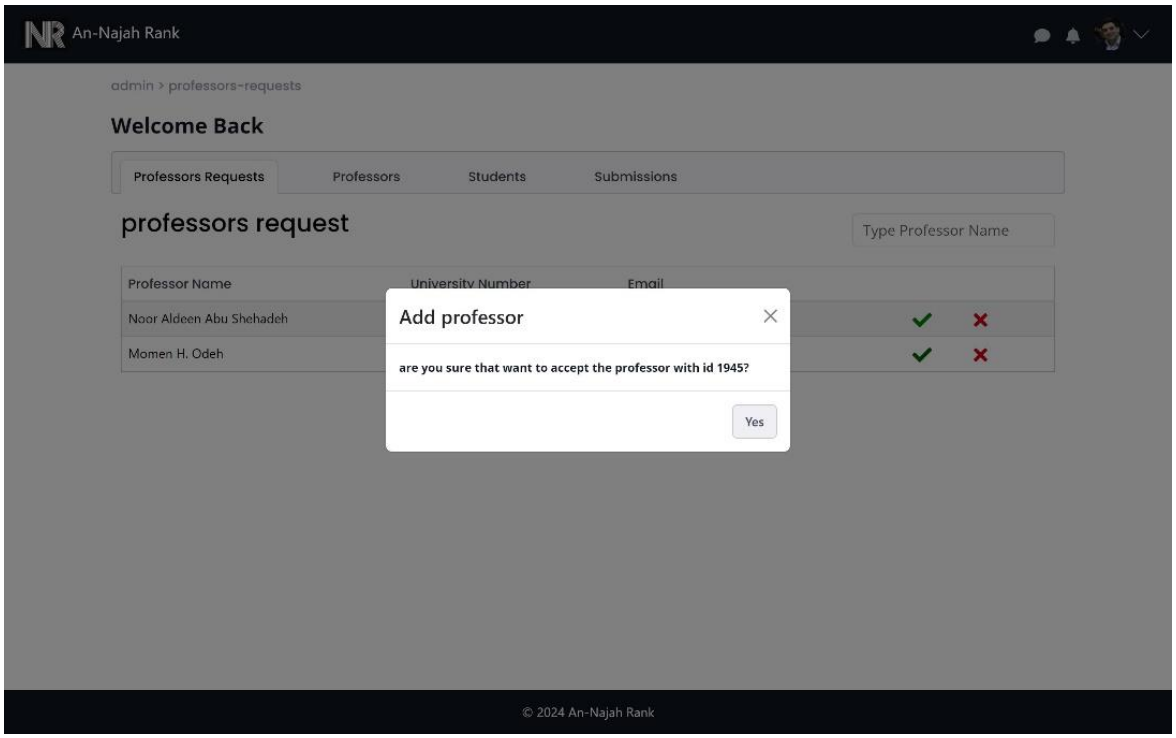


Figure 48: approve professor

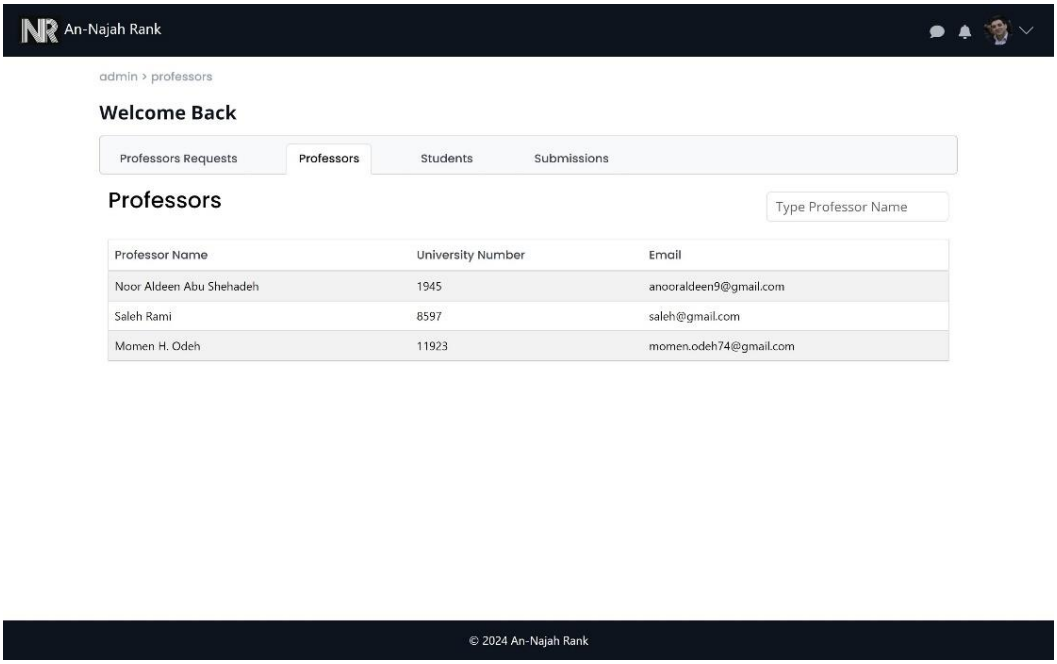


Figure 49: all professors in the system

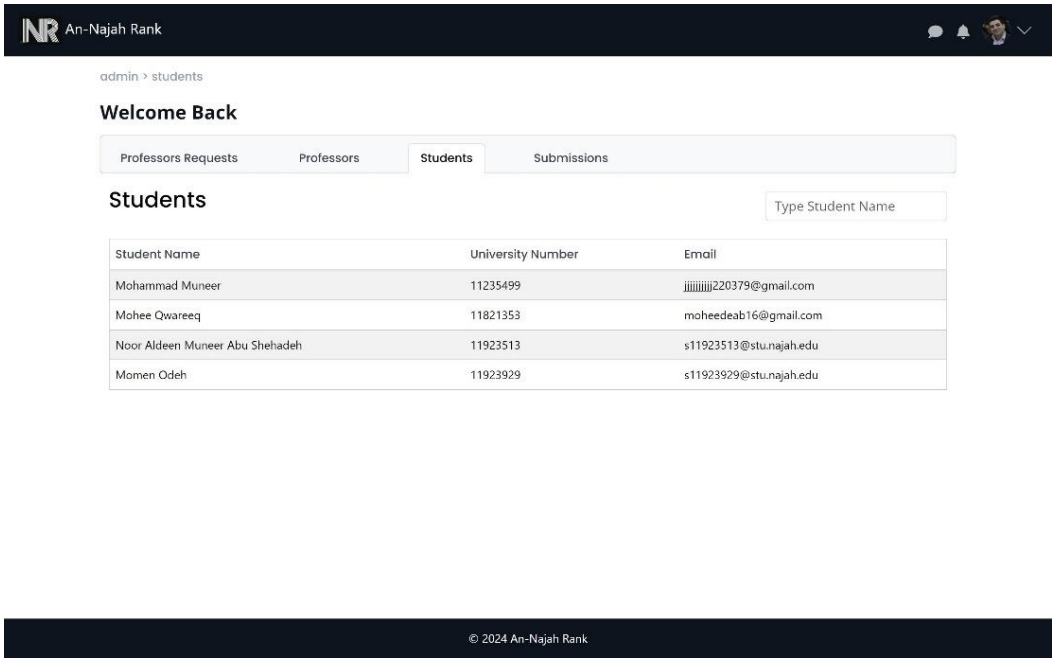


Figure 50: all students in the system

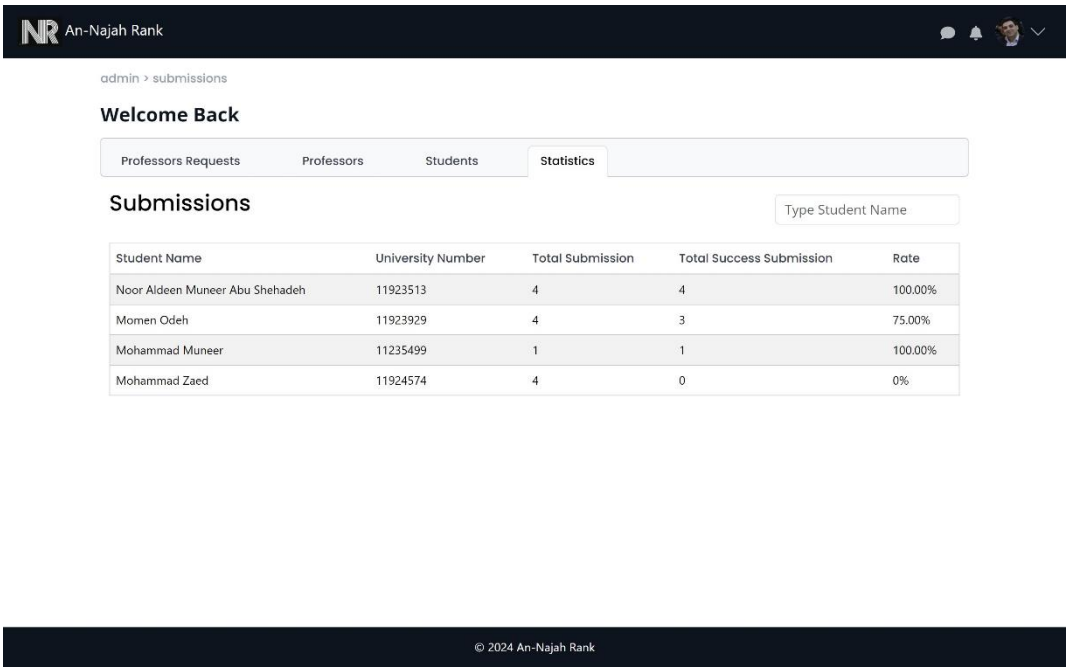


Figure 51: Students statistics

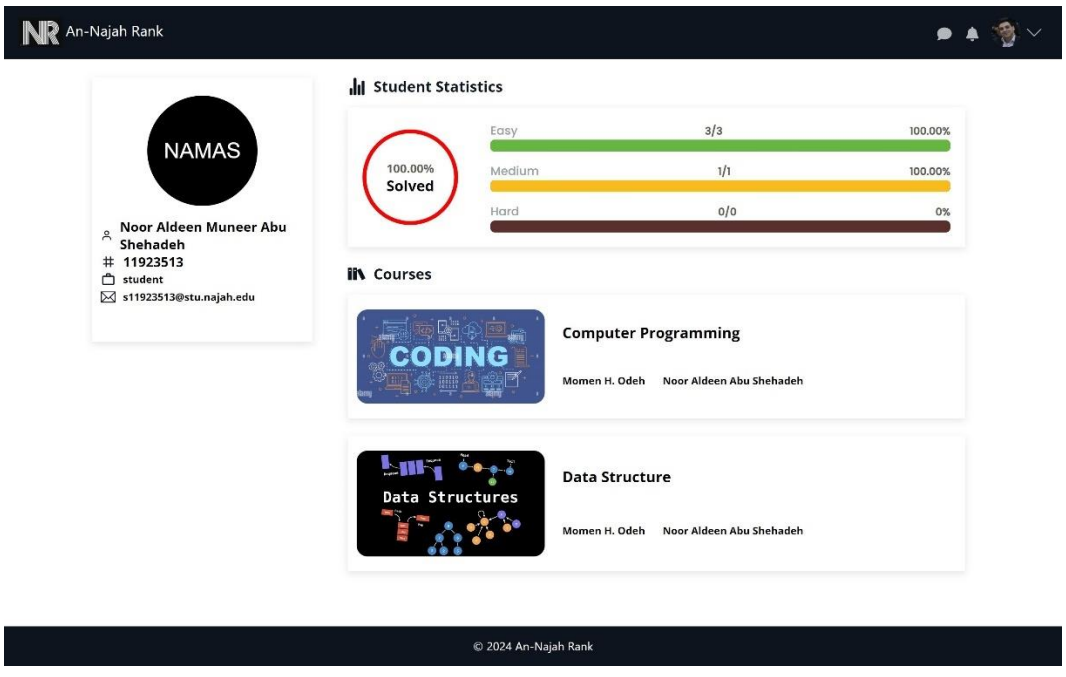


Figure 52: Viewing student profiles from the admin side.

3.4.3 Professor Features:

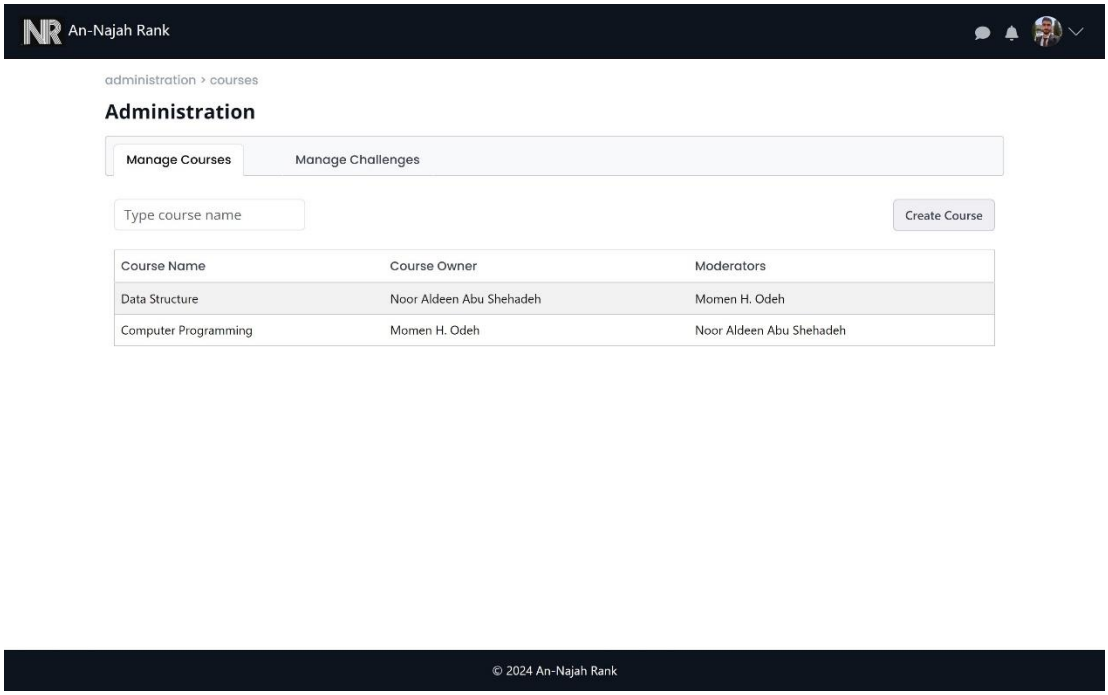


Figure 53: Manage Courses in administration page

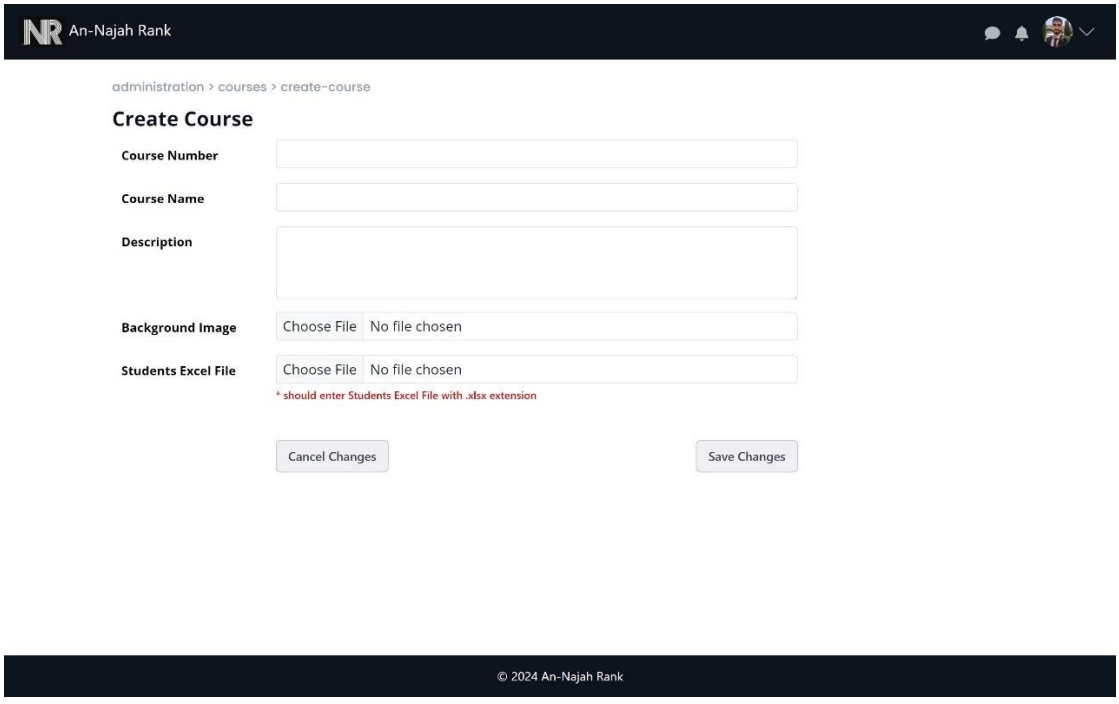


Figure 54: Create course

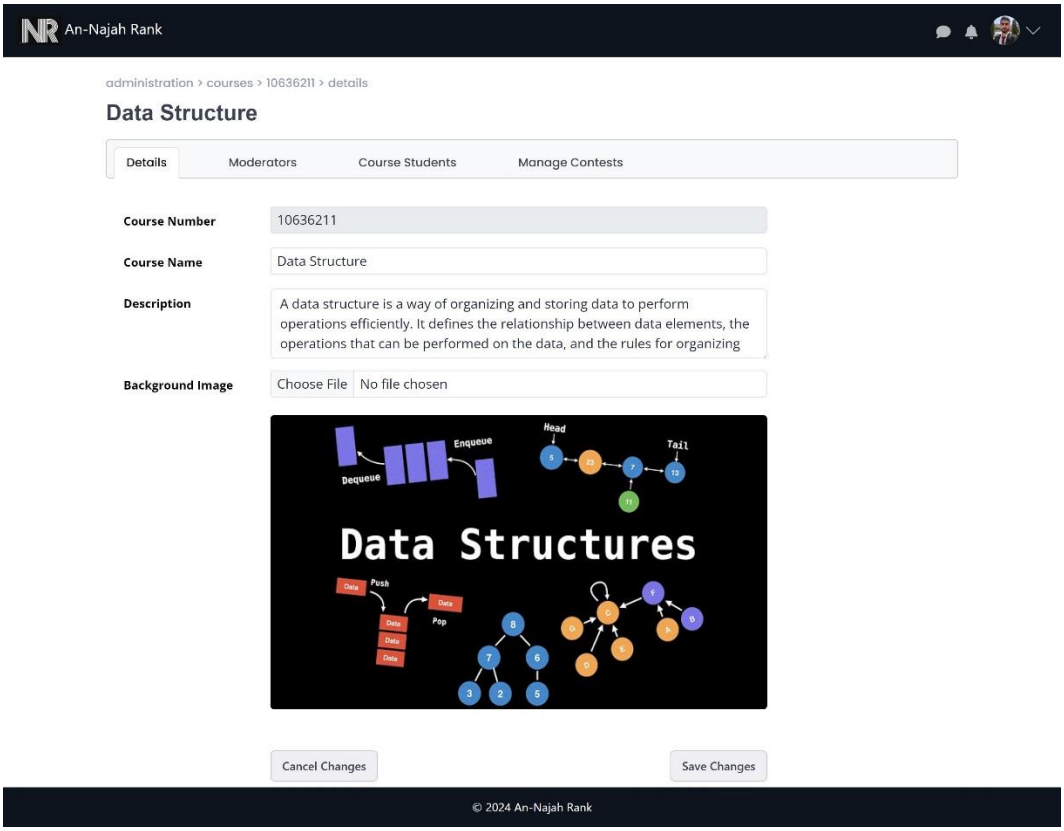


Figure 55: Manage course details

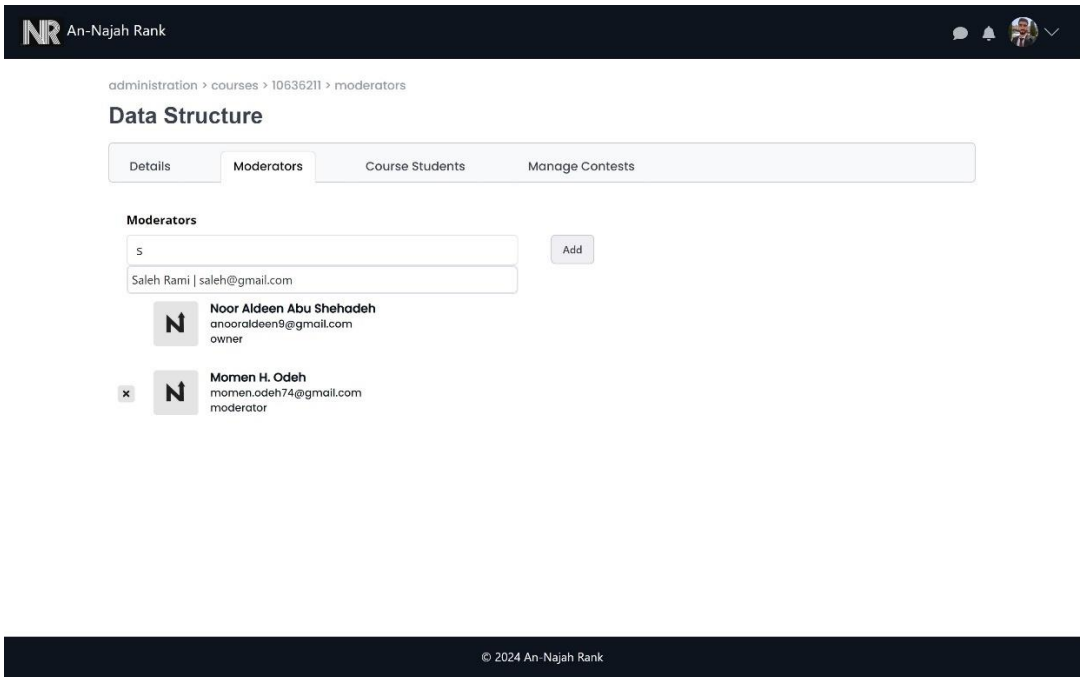


Figure 56: Manage course moderators

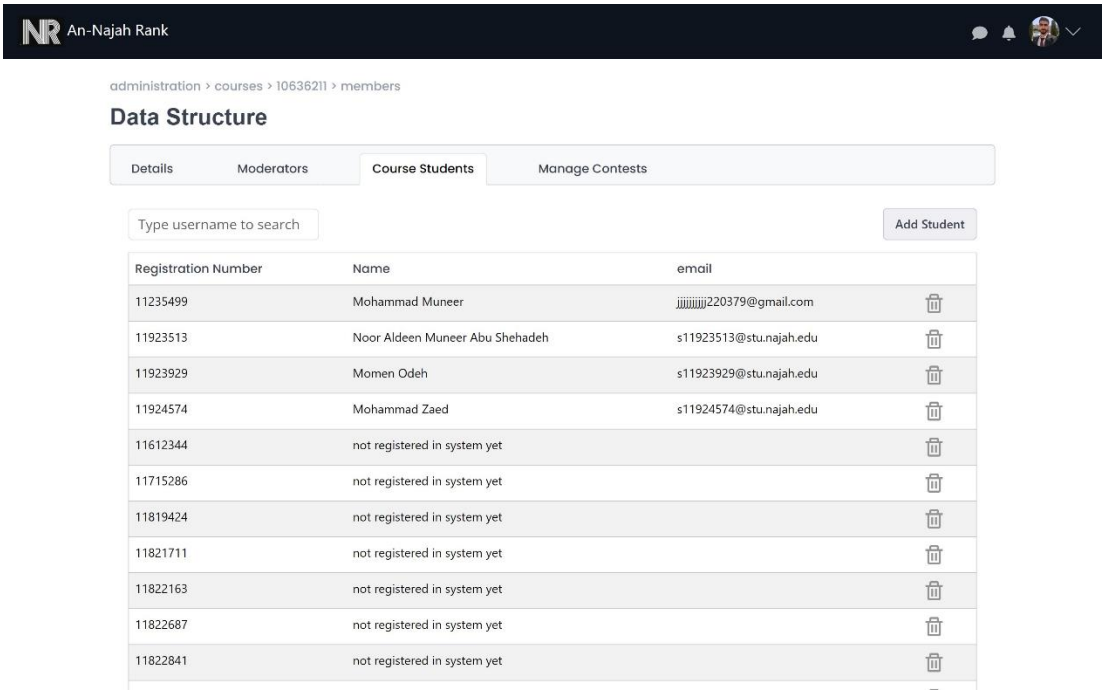


Figure 57: Manage students in course

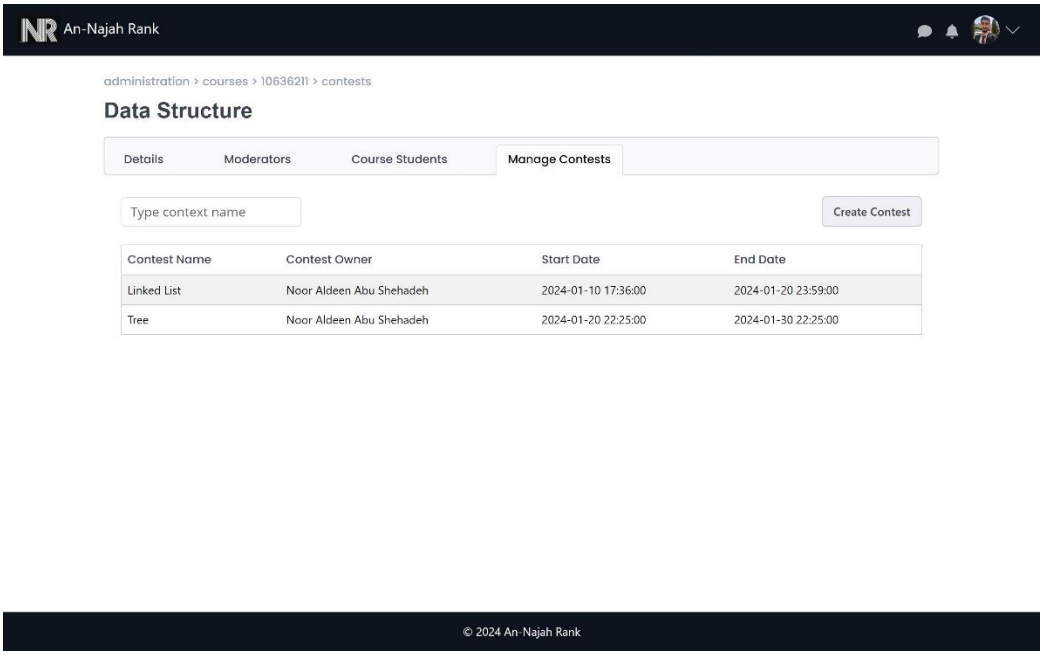


Figure 58: Manage contests

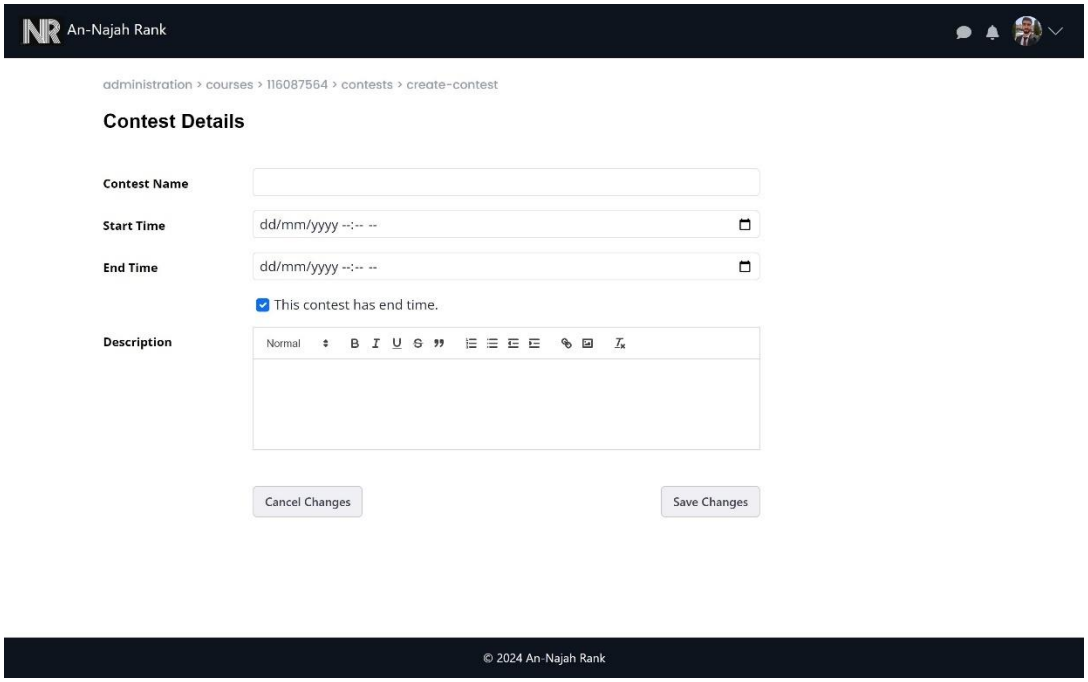


Figure 59: Create contest

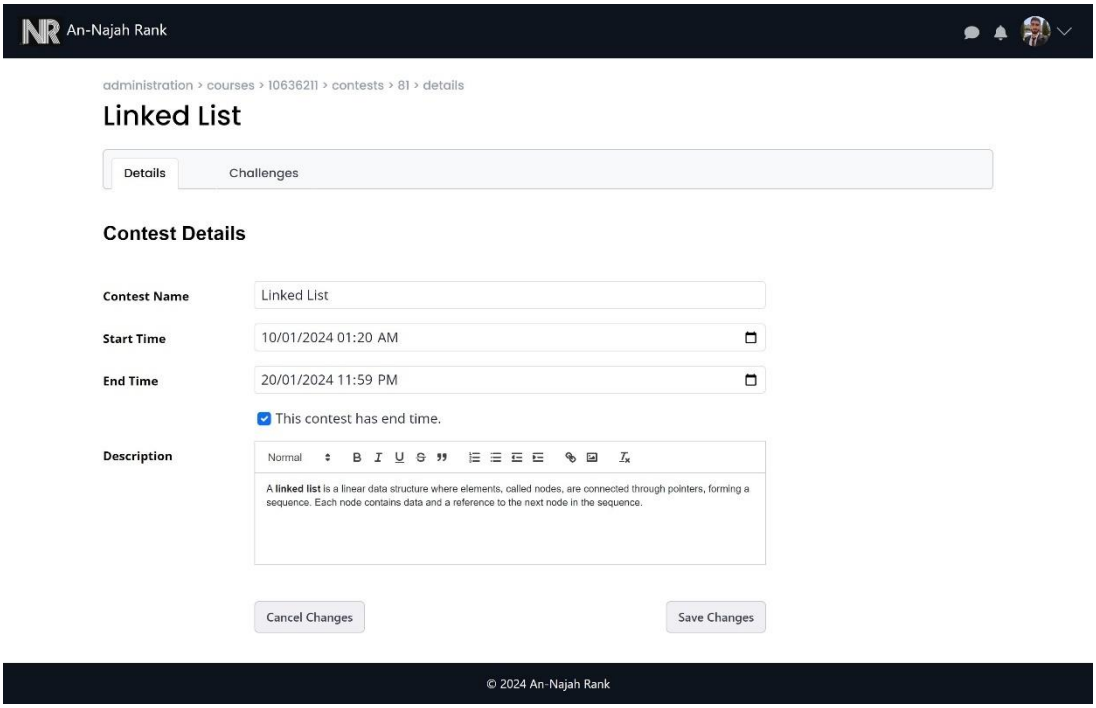


Figure 60: Manage contest details

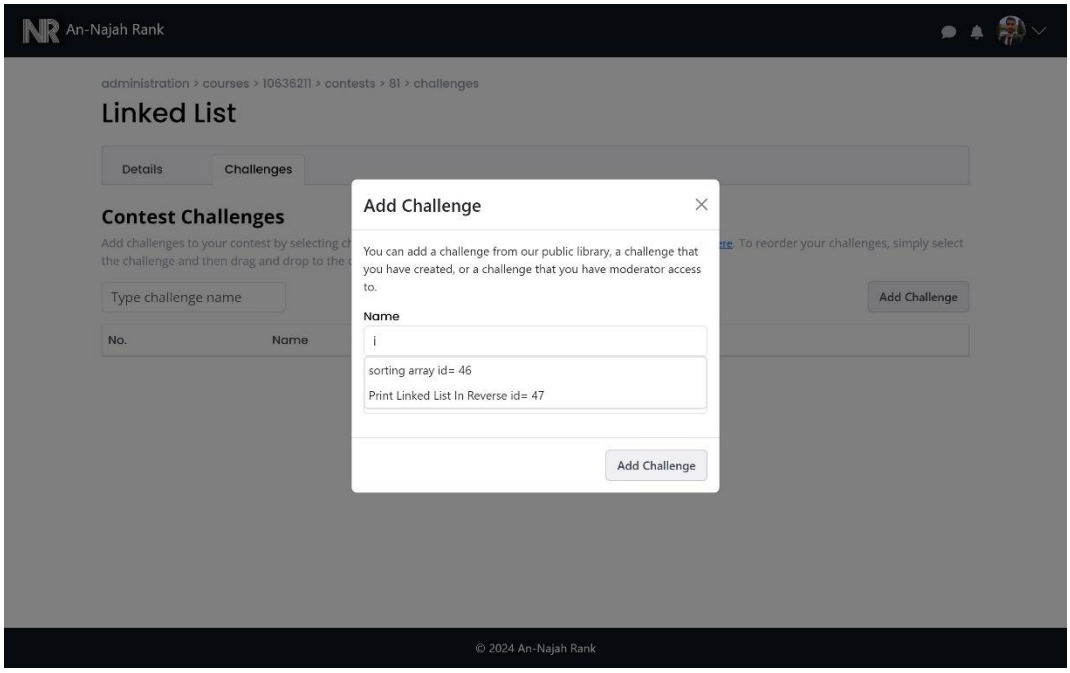


Figure 61: Add challenge to contest

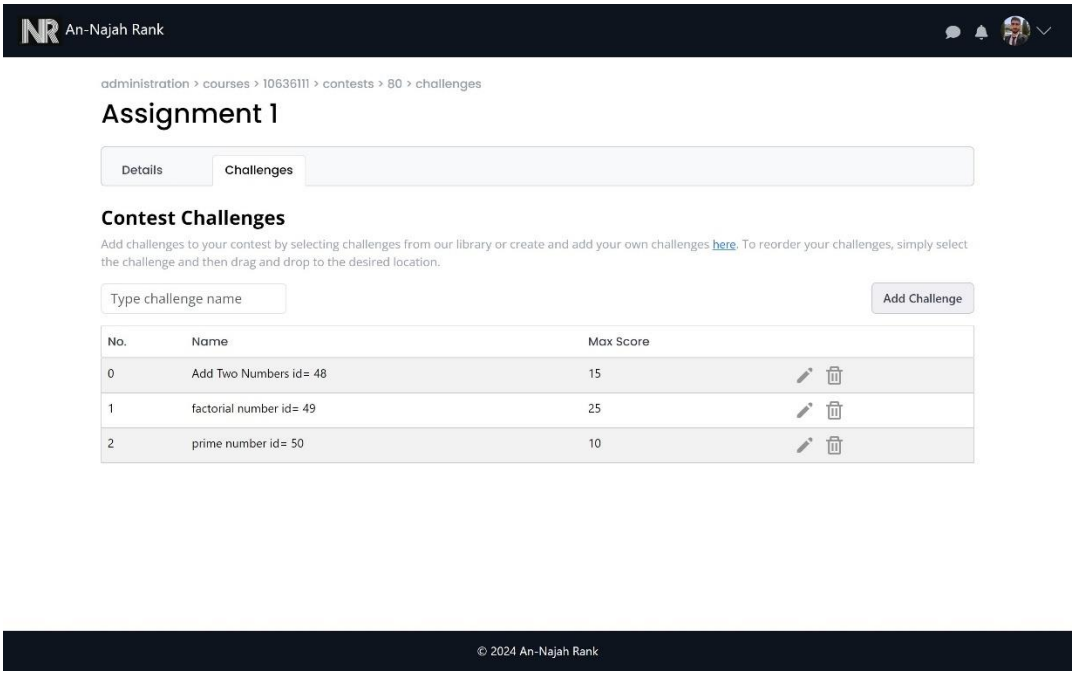


Figure 62: Manage challenges in contest

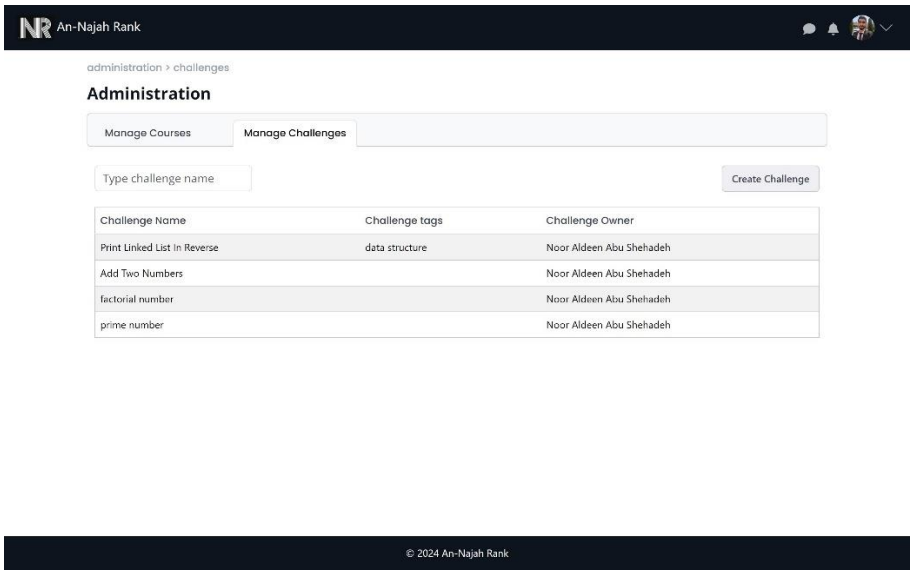


Figure 63: Manage challenges

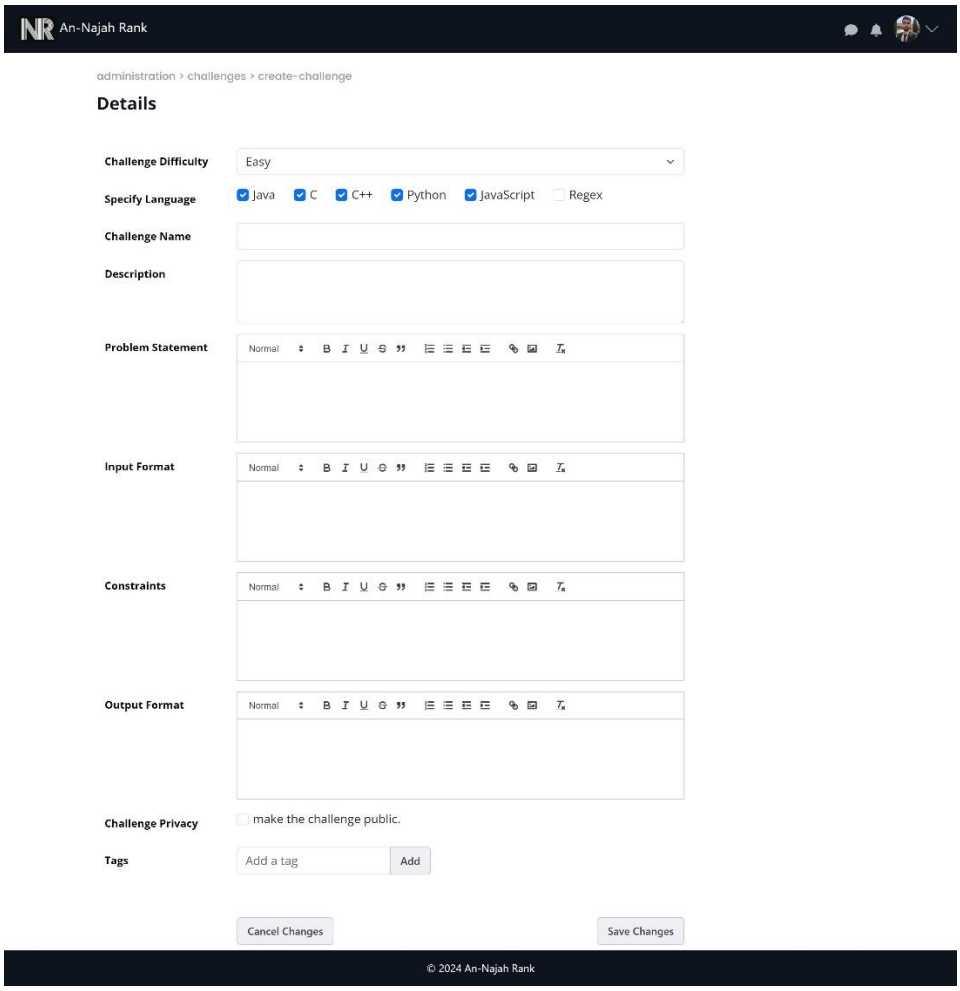


Figure 64: Create challenge

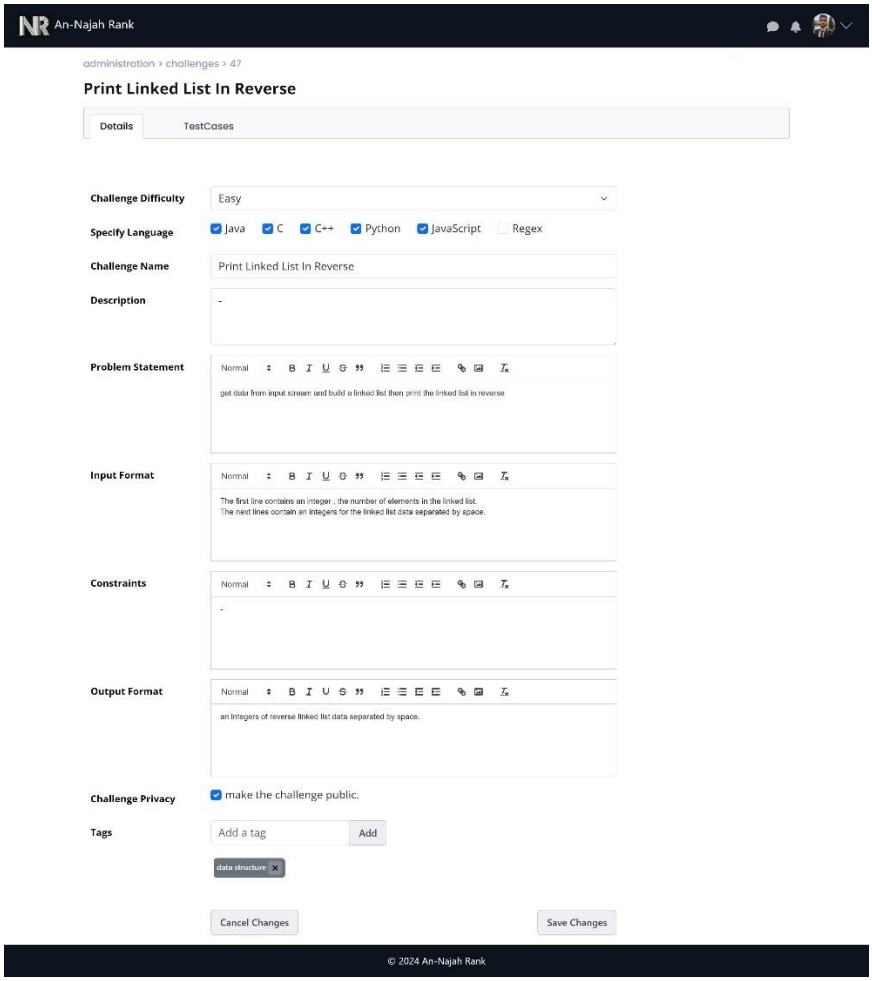


Figure 65: Manage challenge details

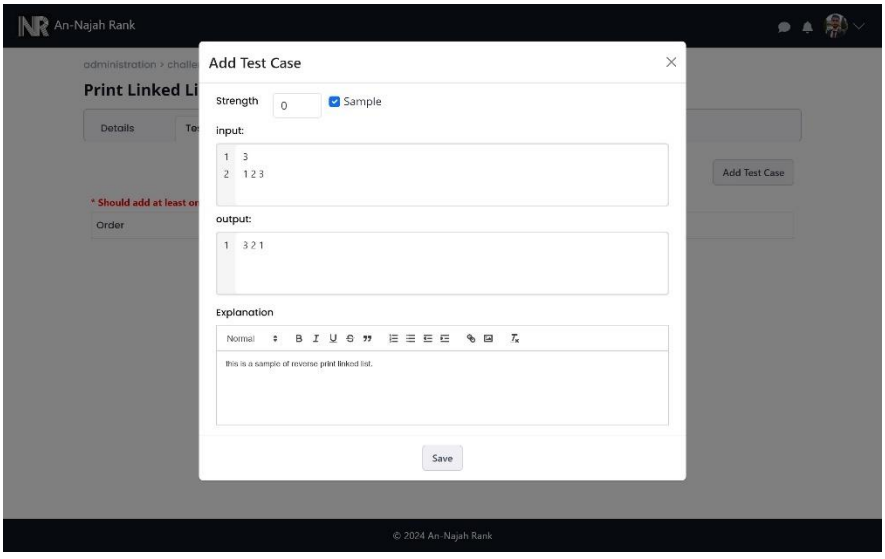


Figure 66: Add test case to challenge

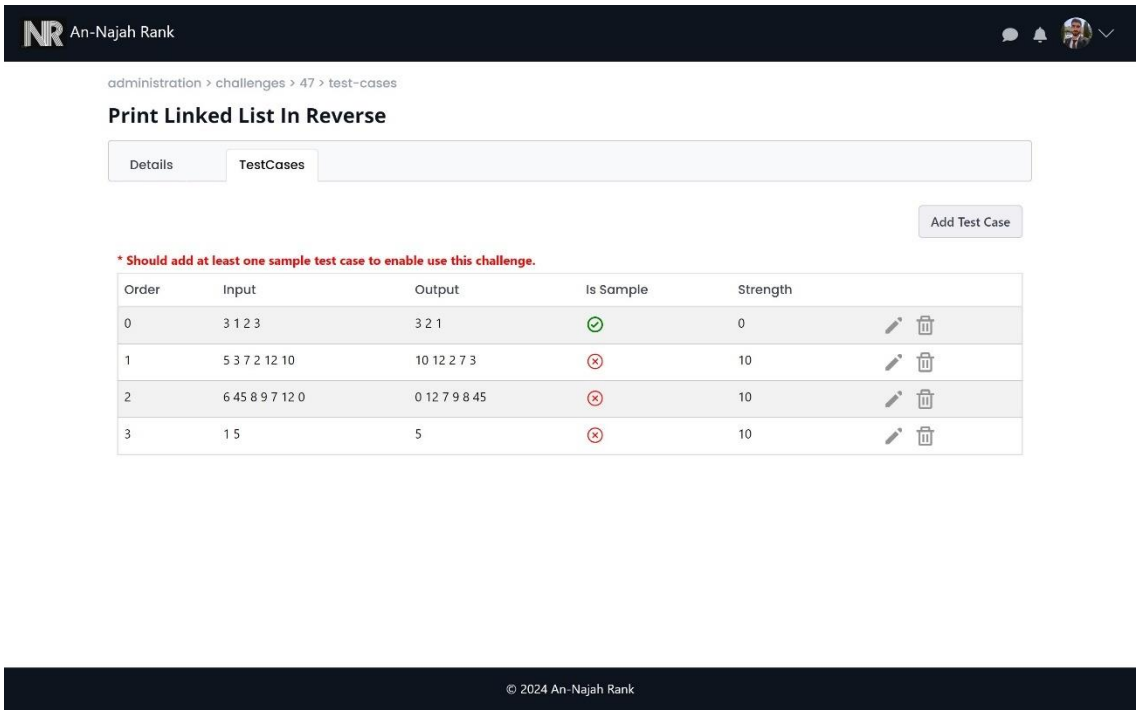


Figure 67: Manage test cases in challenge

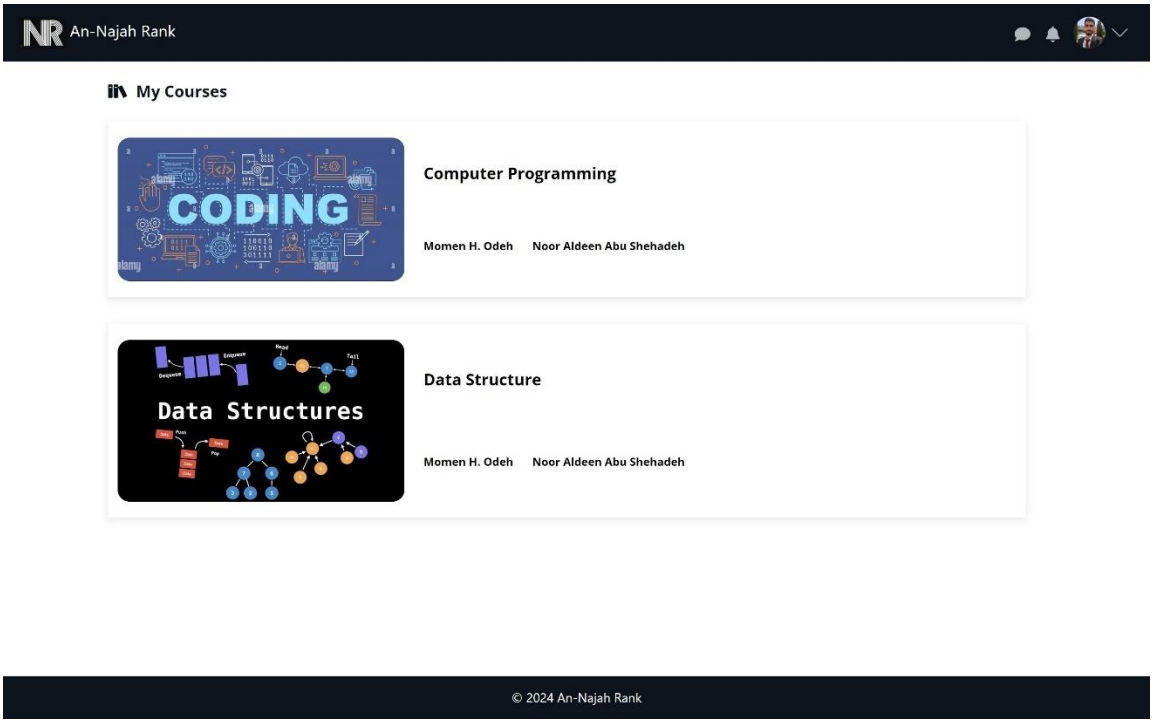


Figure 68: All courses page

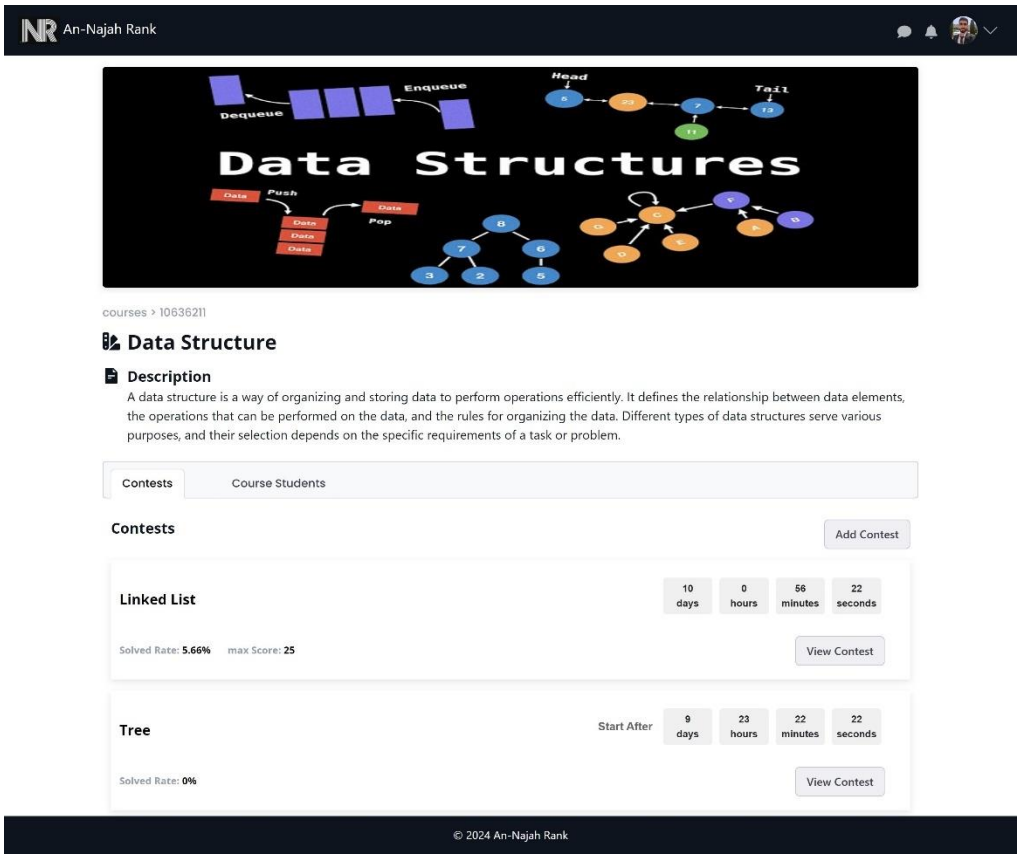


Figure 69: Course view

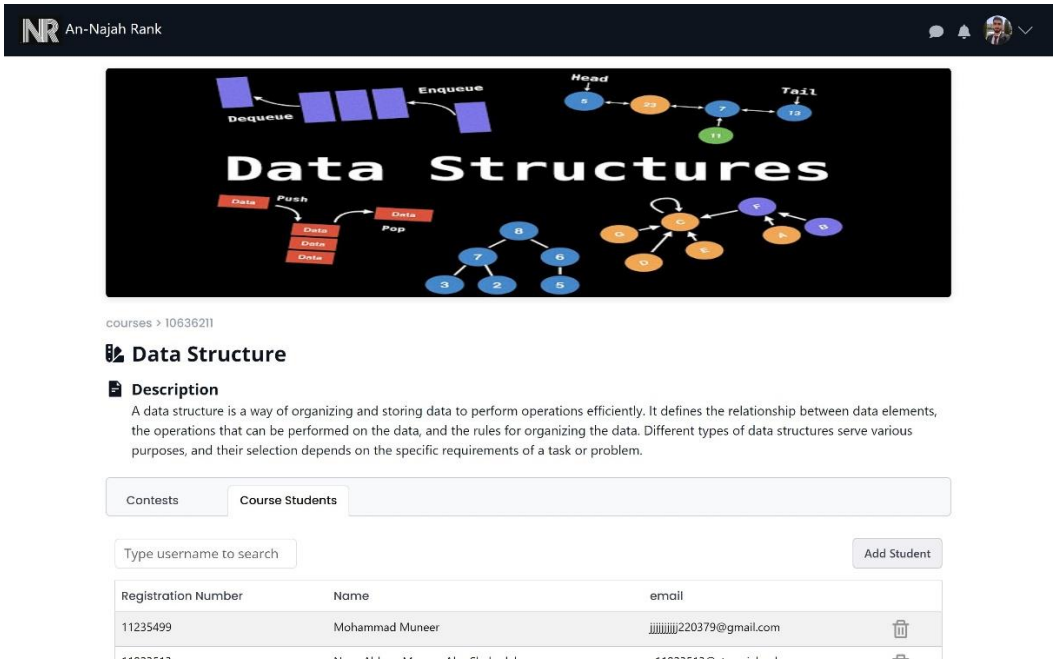


Figure 70: Manage students in course

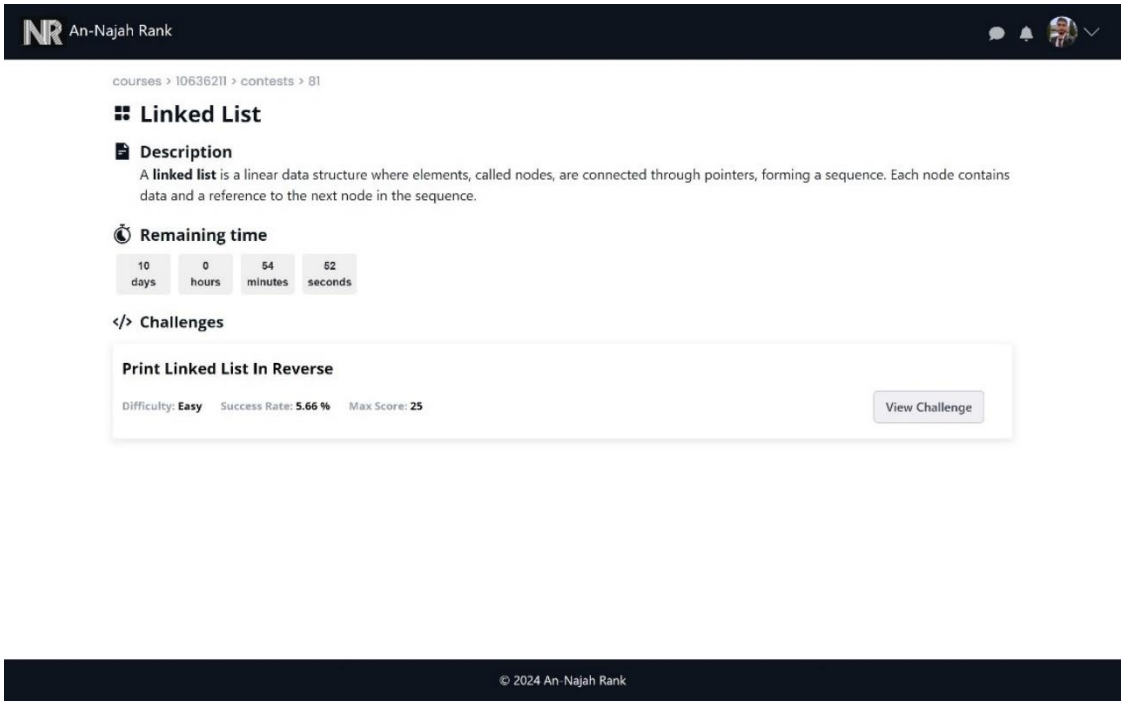


Figure 71: Contest view

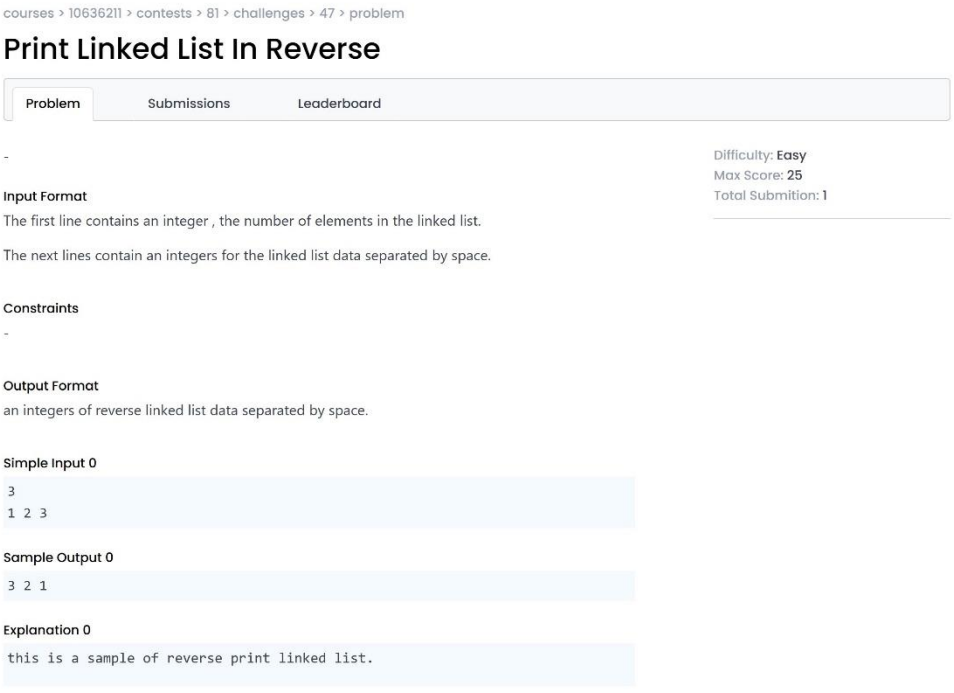


Figure 72: Problem description

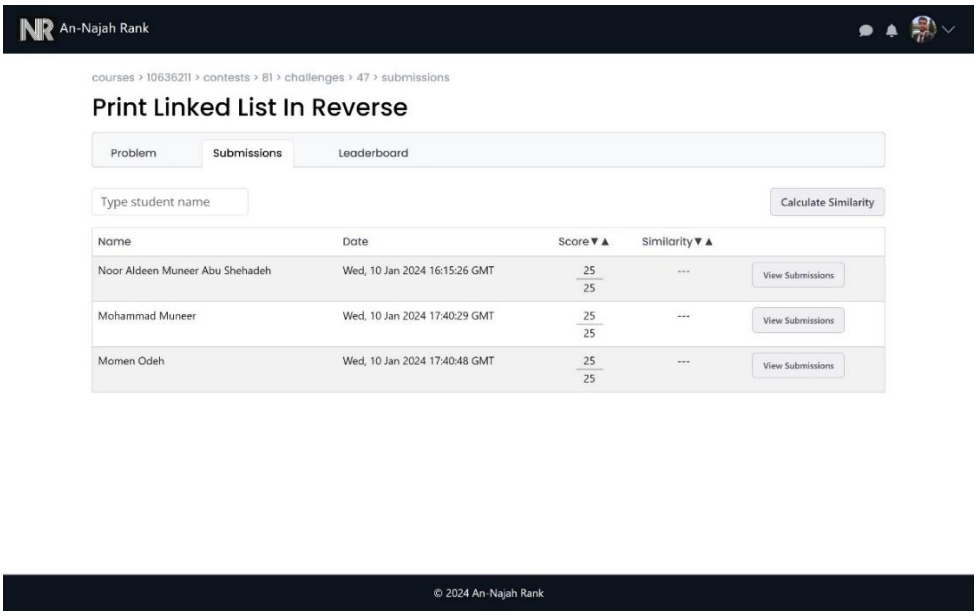


Figure 73: Students submissions from professor side

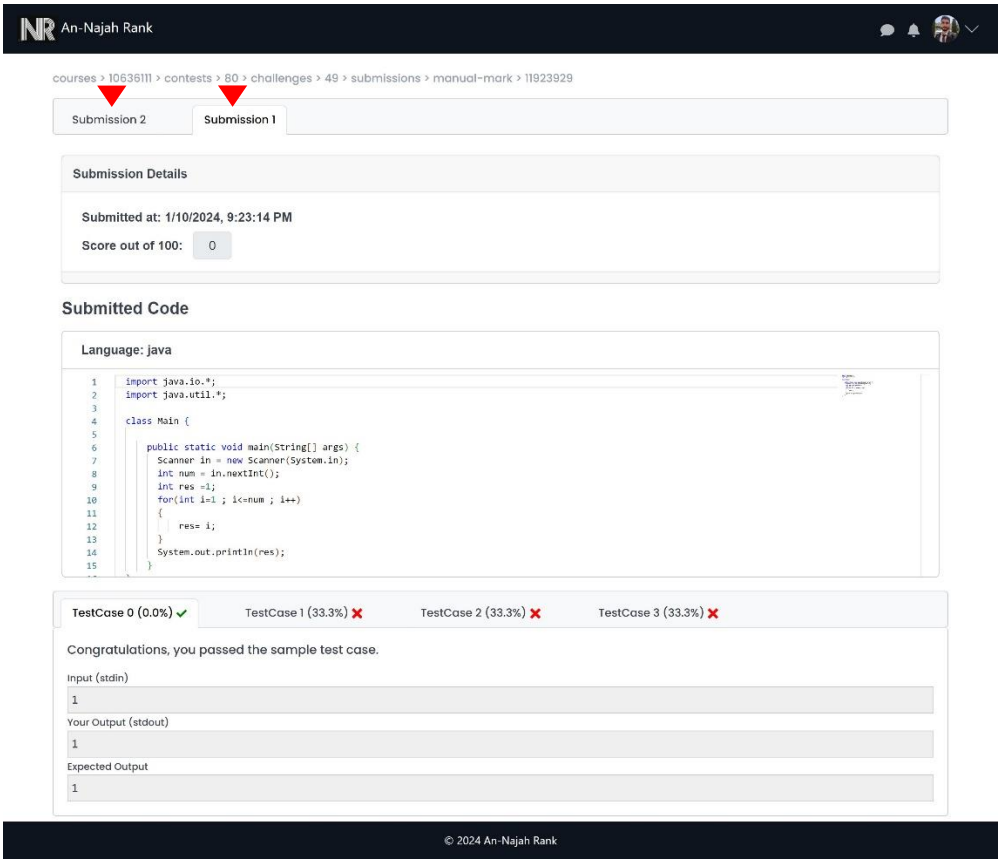


Figure 74: View student submissions

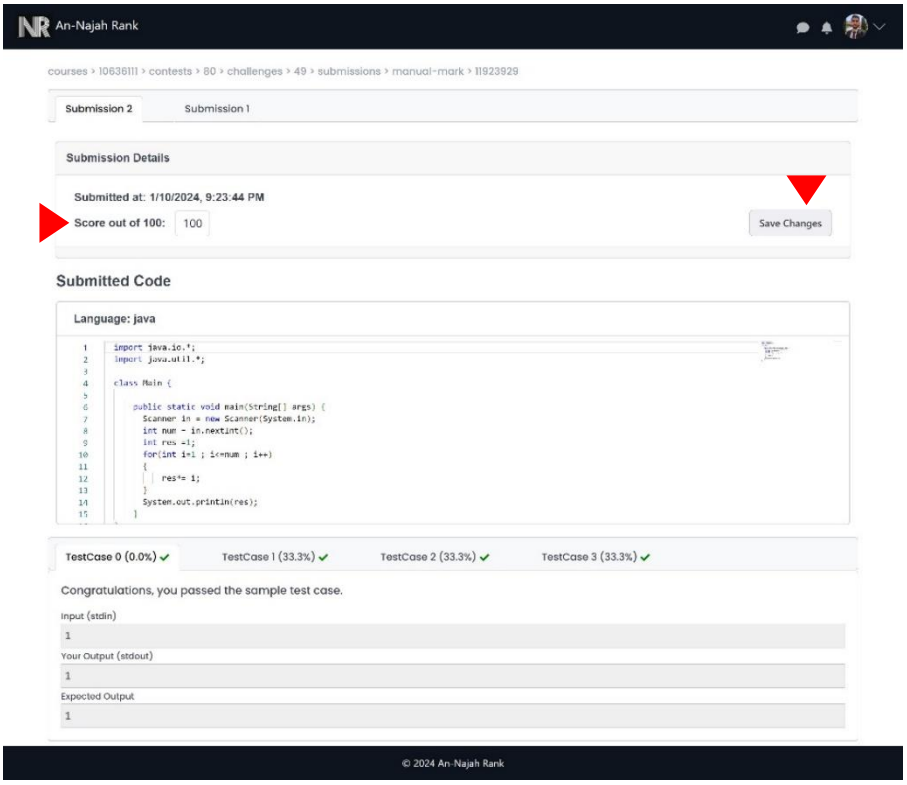


Figure 75: Last submission for student can do manual mark

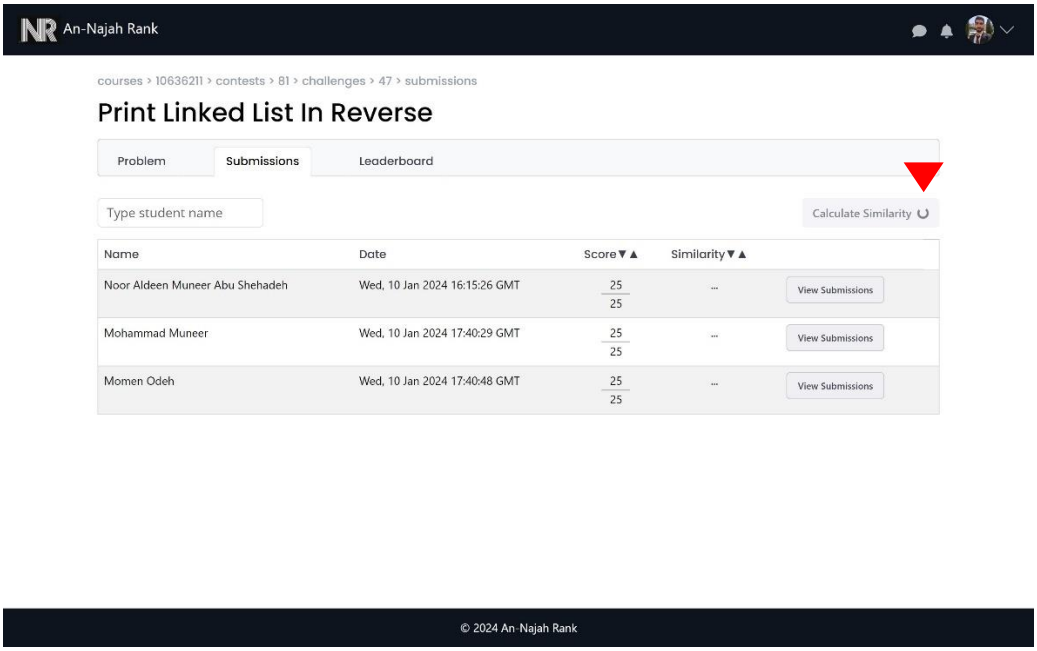


Figure 76: Start calculate similarity

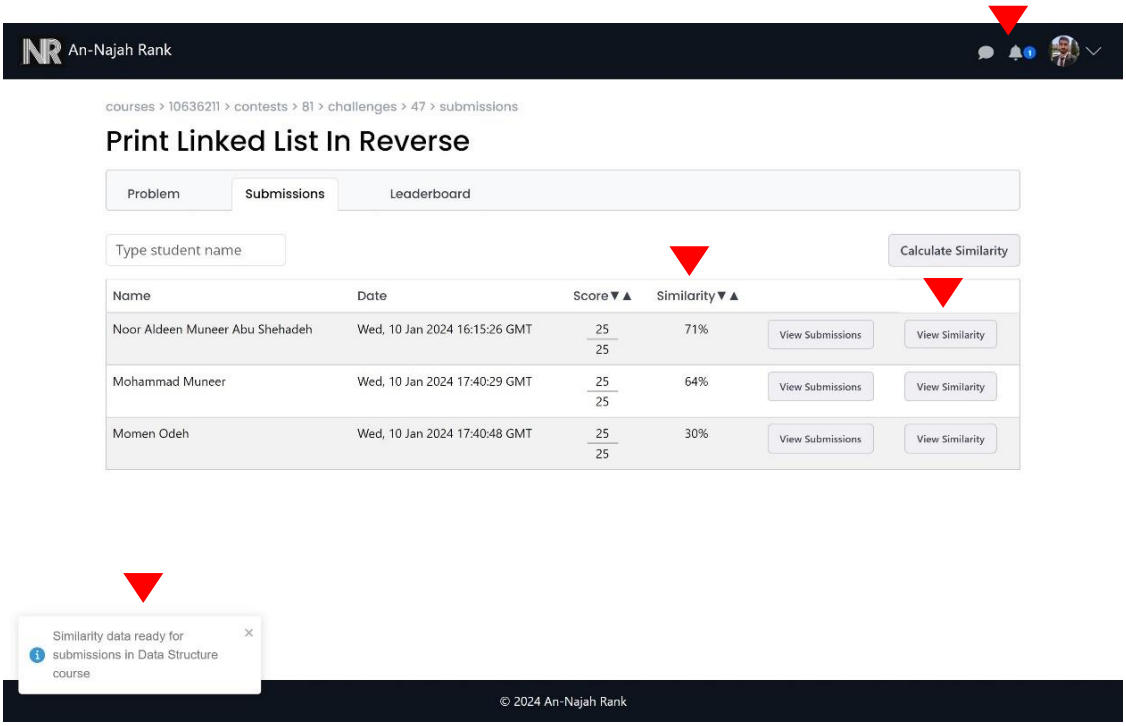


Figure 77: Received notification when similarity calculated

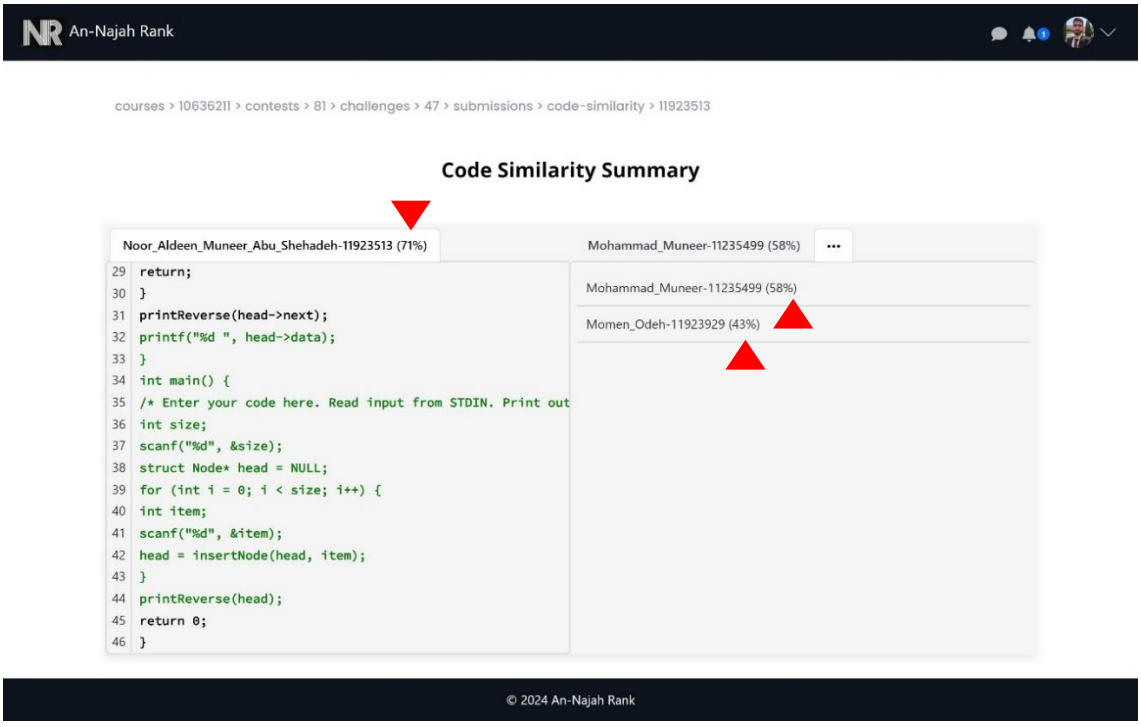


Figure 78: Code similarity page

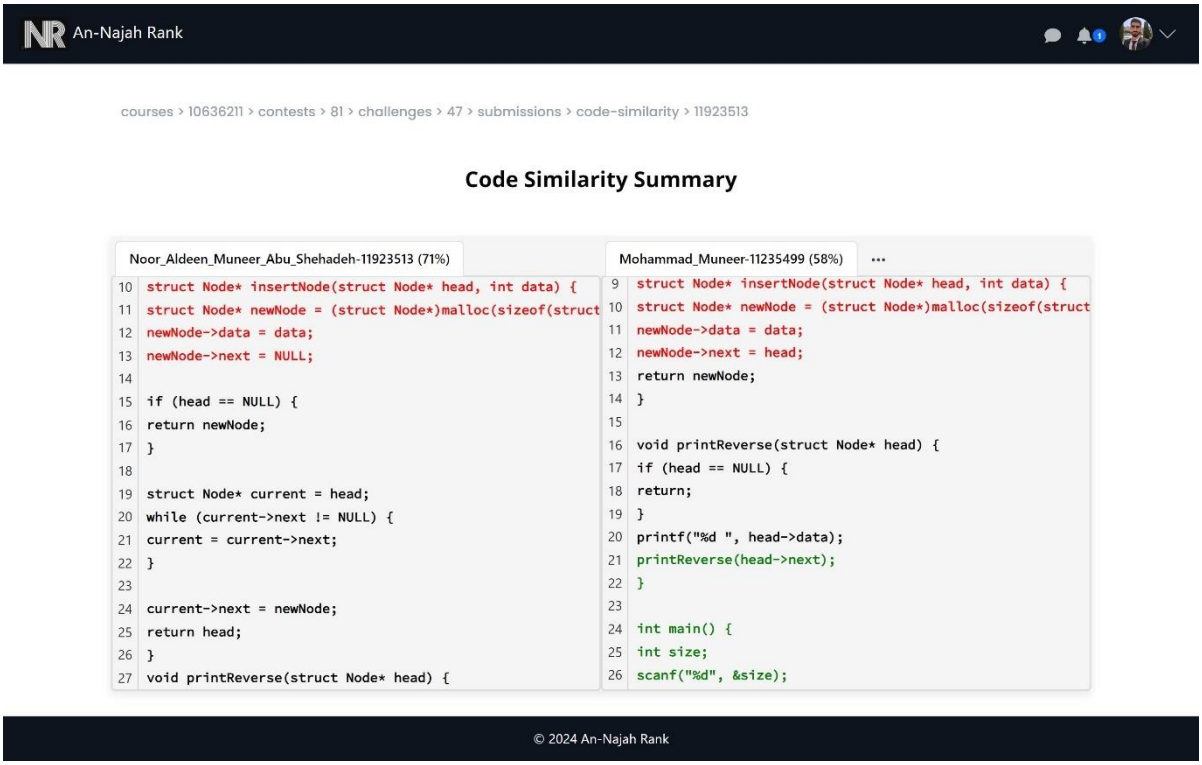


Figure 79: Code Similarity view 1

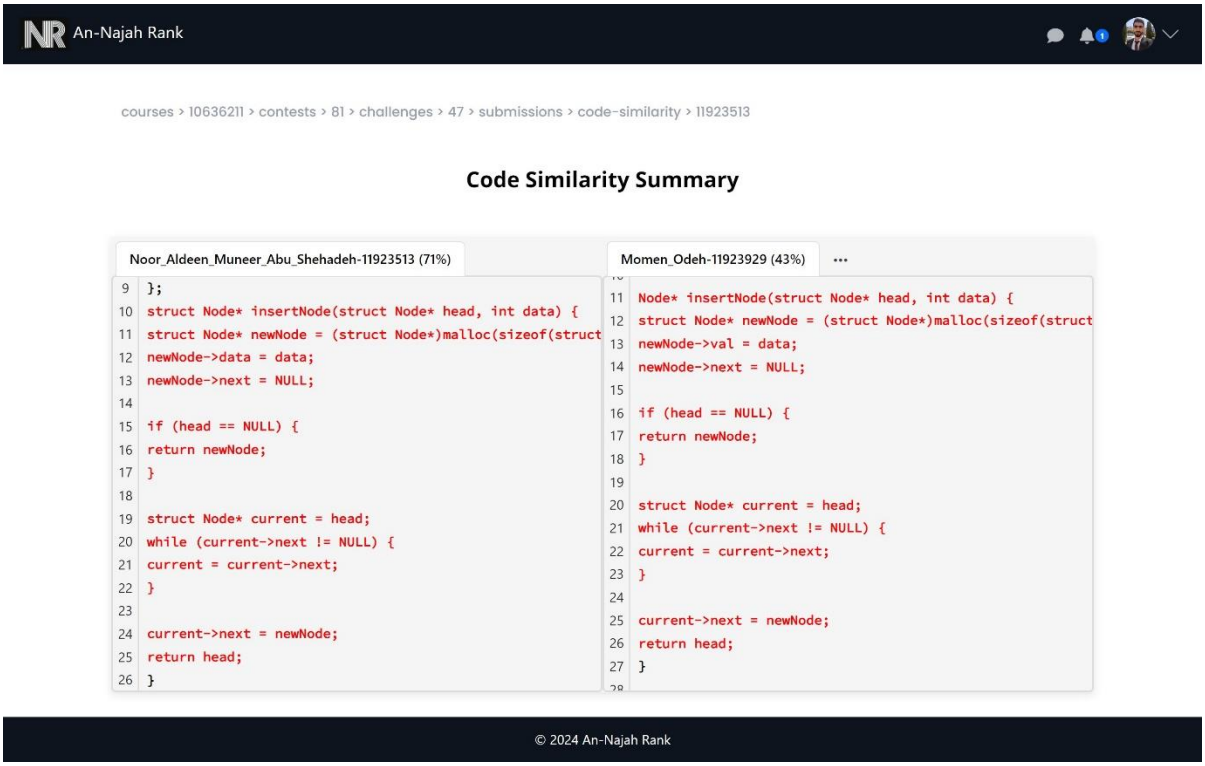


Figure 80: Code similarity view 2

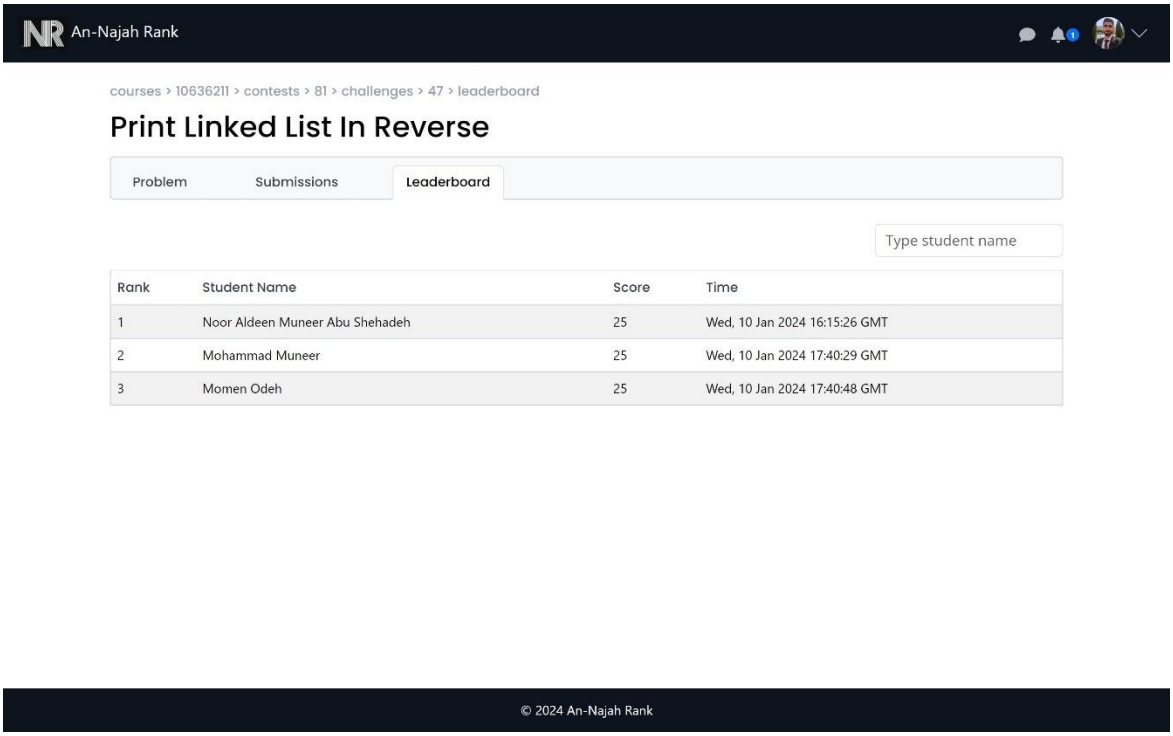


Figure 81: students leaderboards

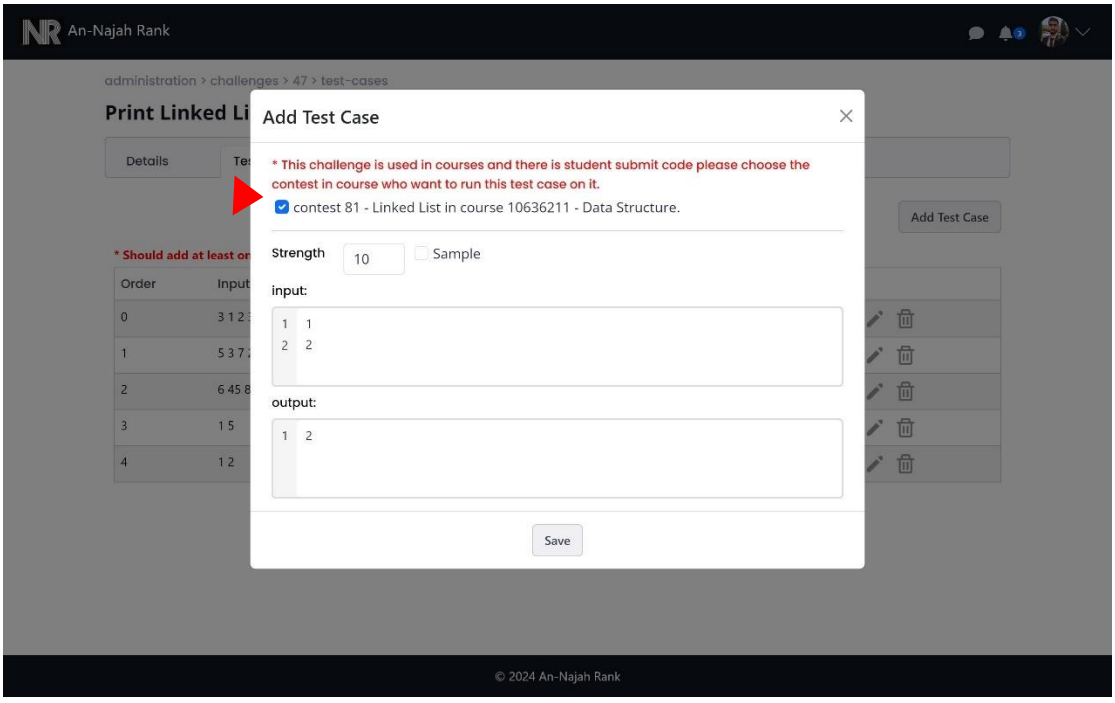


Figure 82: Add new test case when there is a submission for challenge

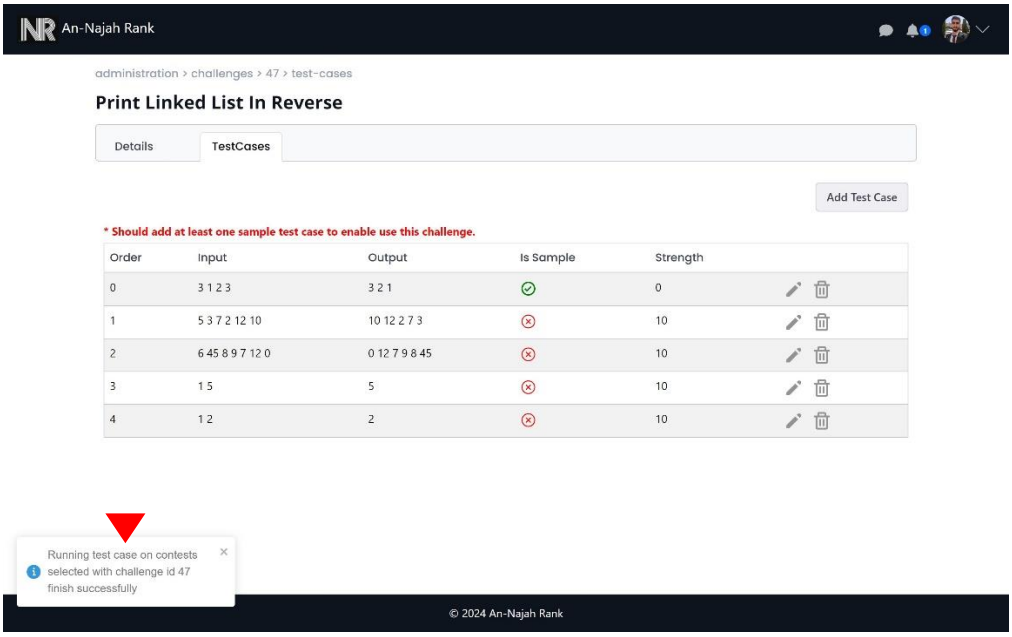


Figure 83: After add test case and run it on all student submission

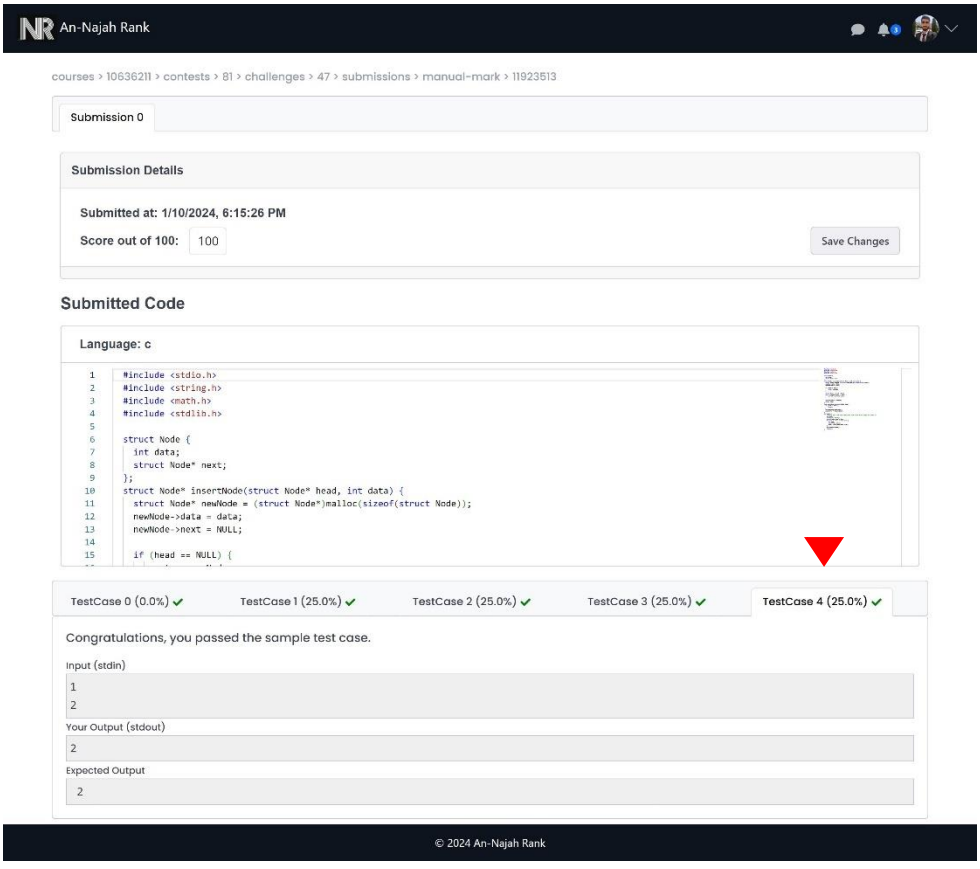


Figure 84: The submission after add new test case

3.4.4 Student Features:

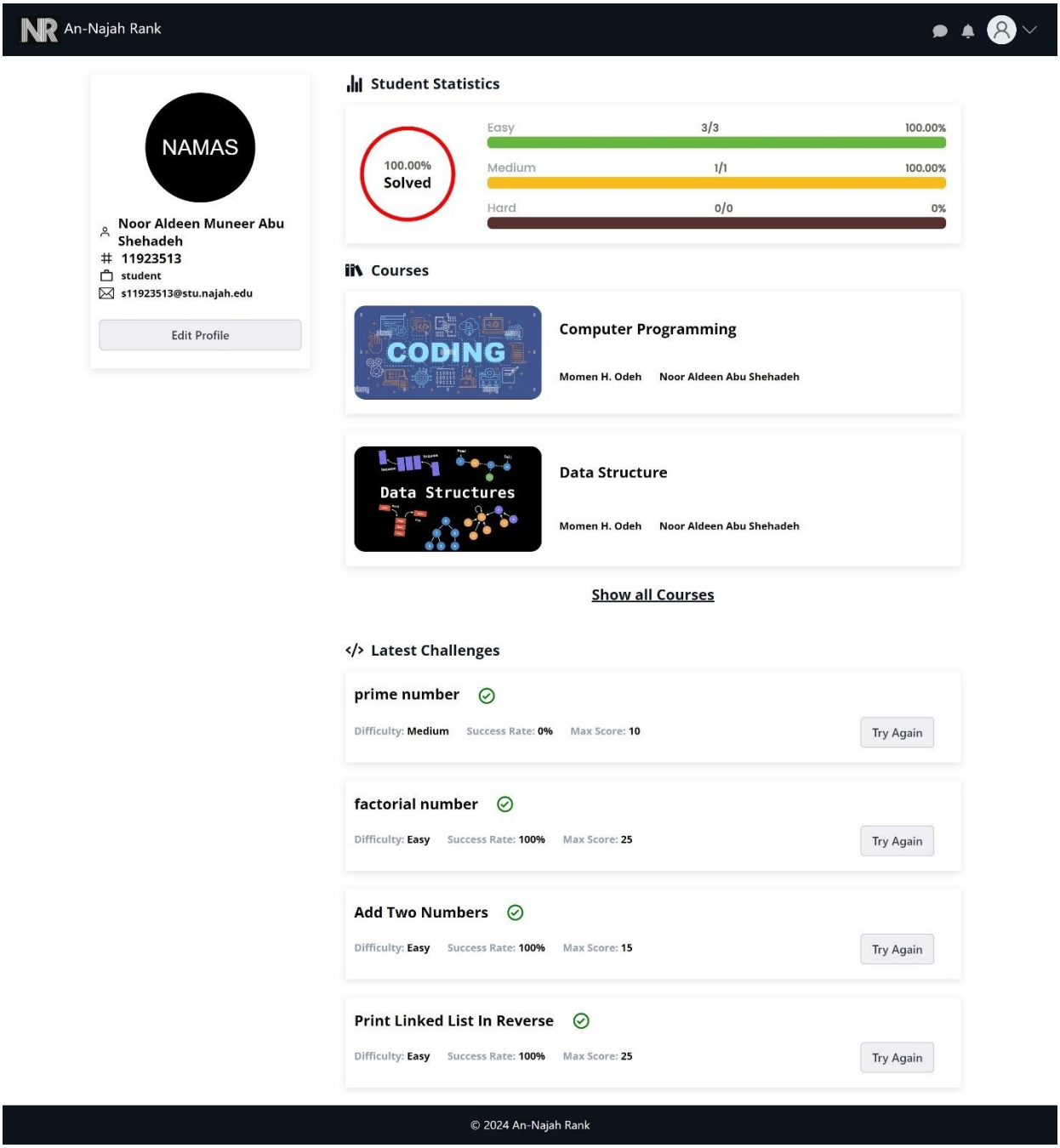


Figure 85: Student profile from student side

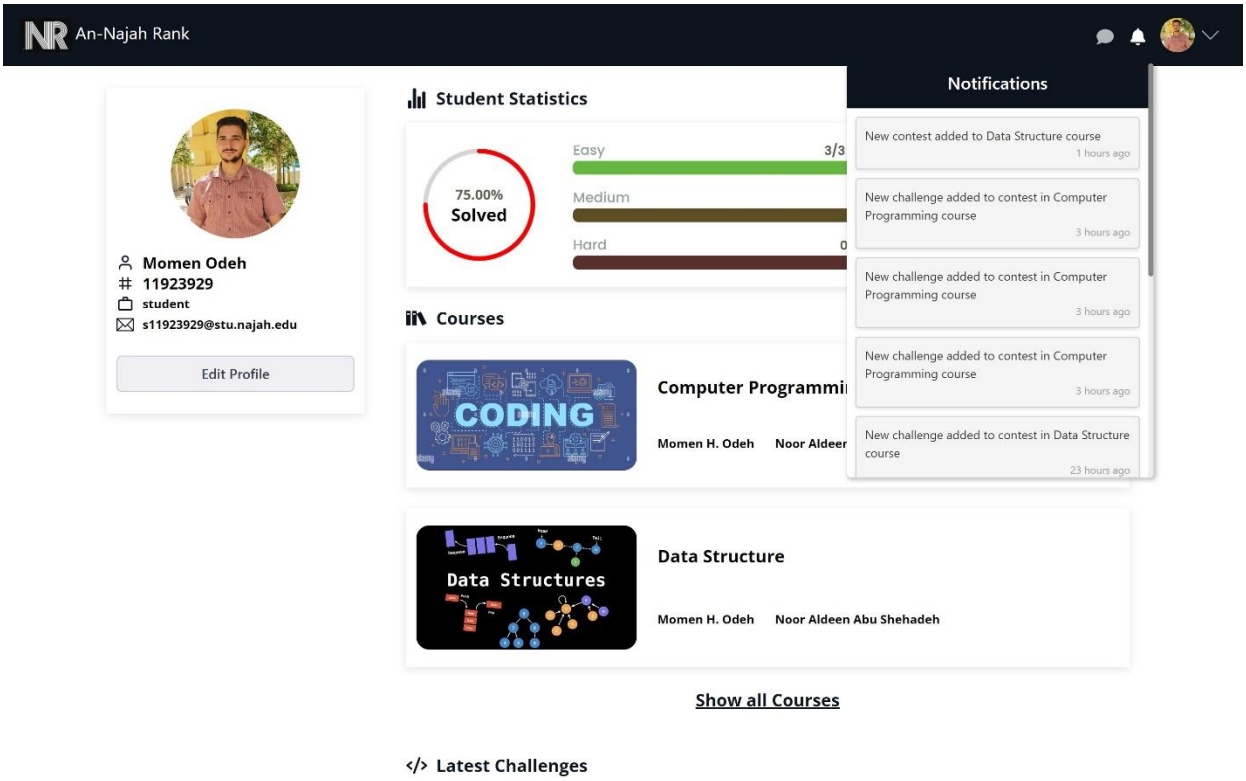


Figure 86: Notifications when add new course or contest or challenge

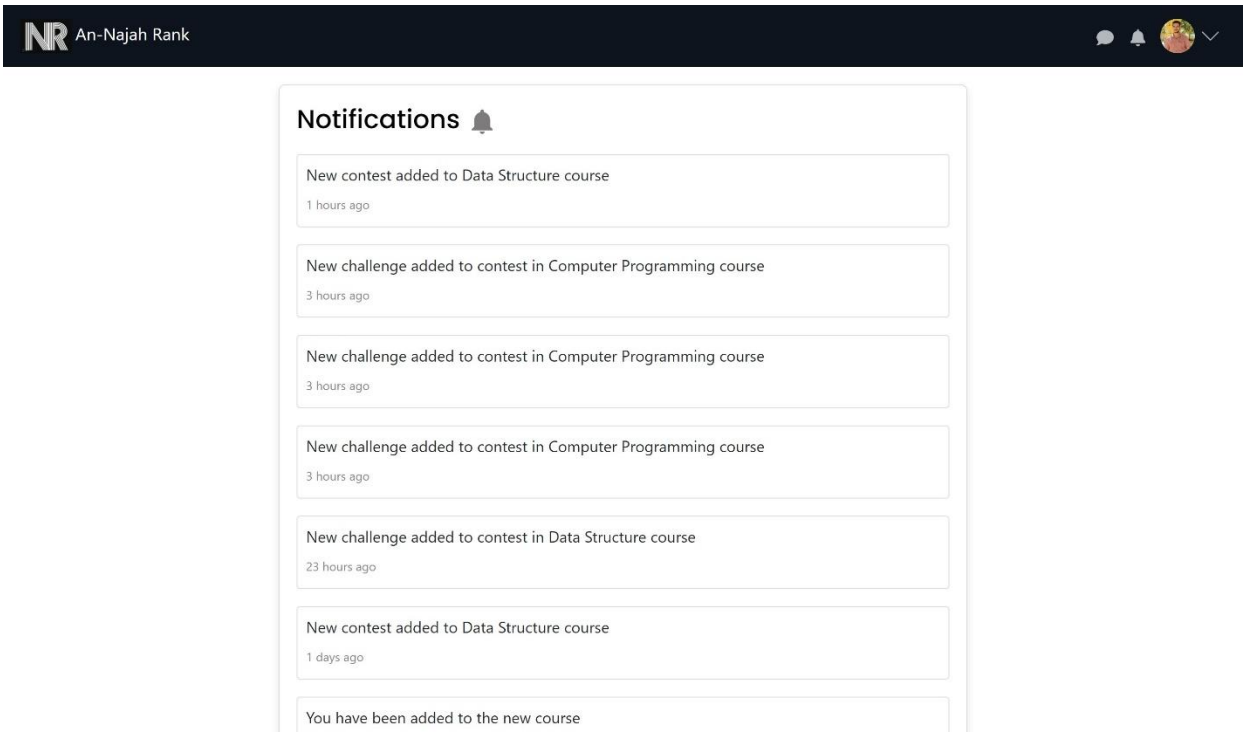


Figure 87: All notification page

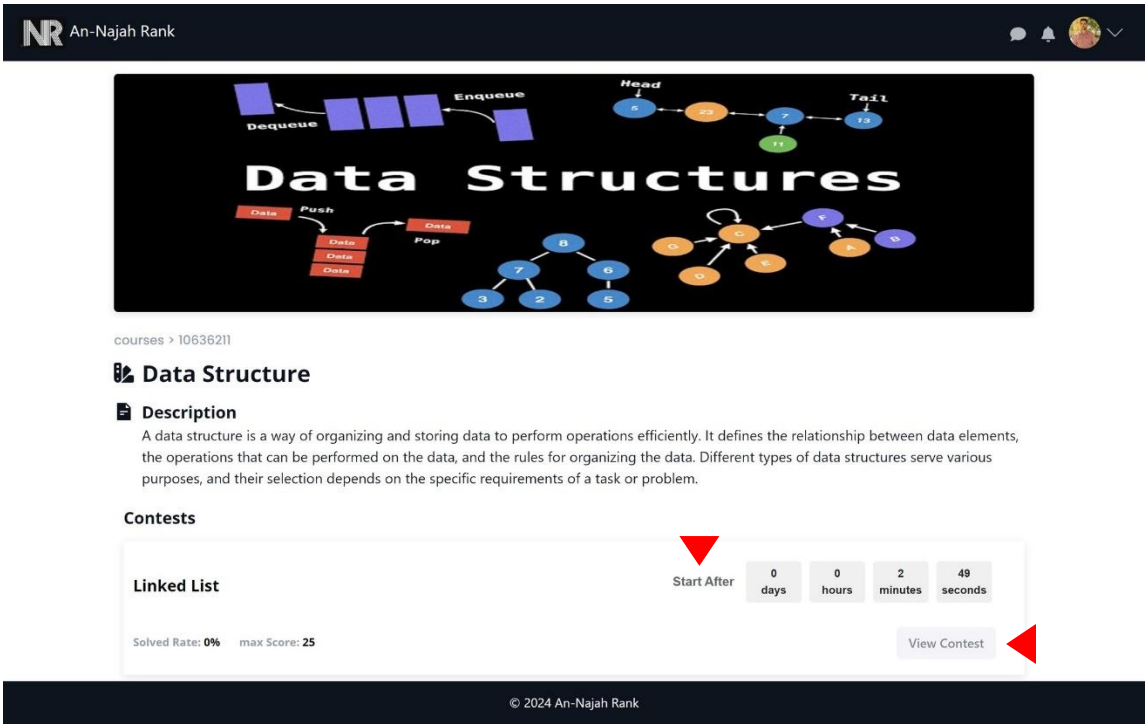


Figure 88: Course view before contest start in student side

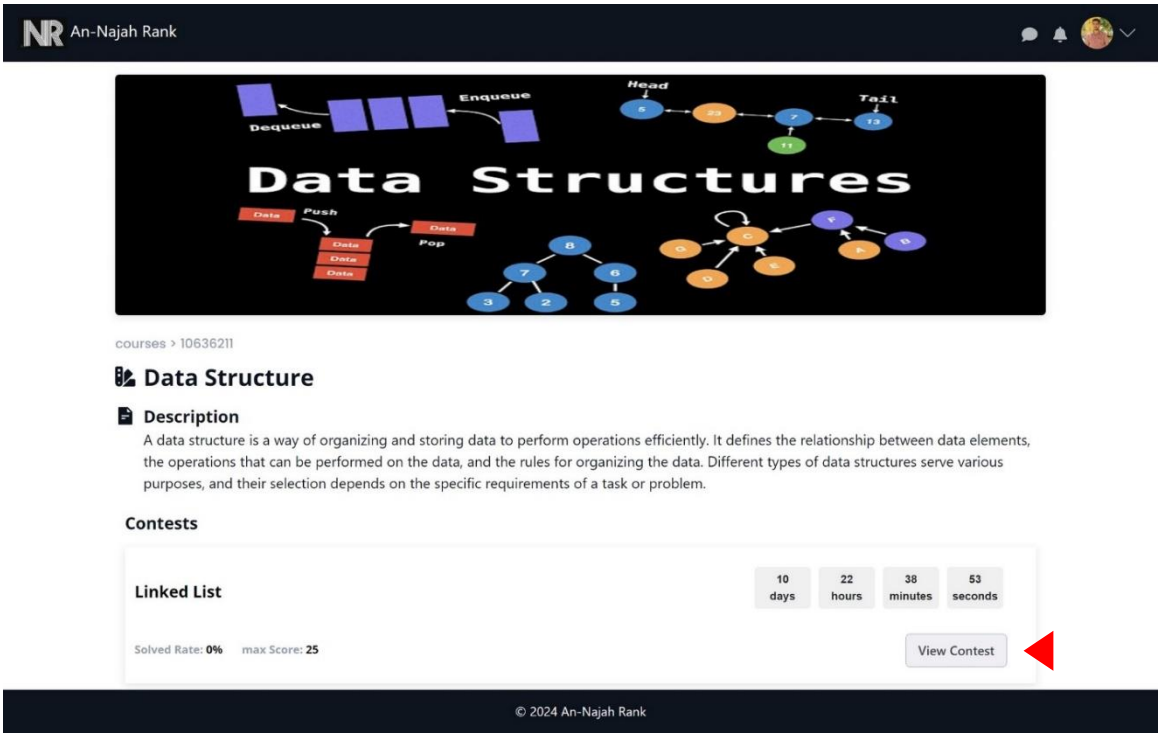


Figure 89: Course view after contest start in student side

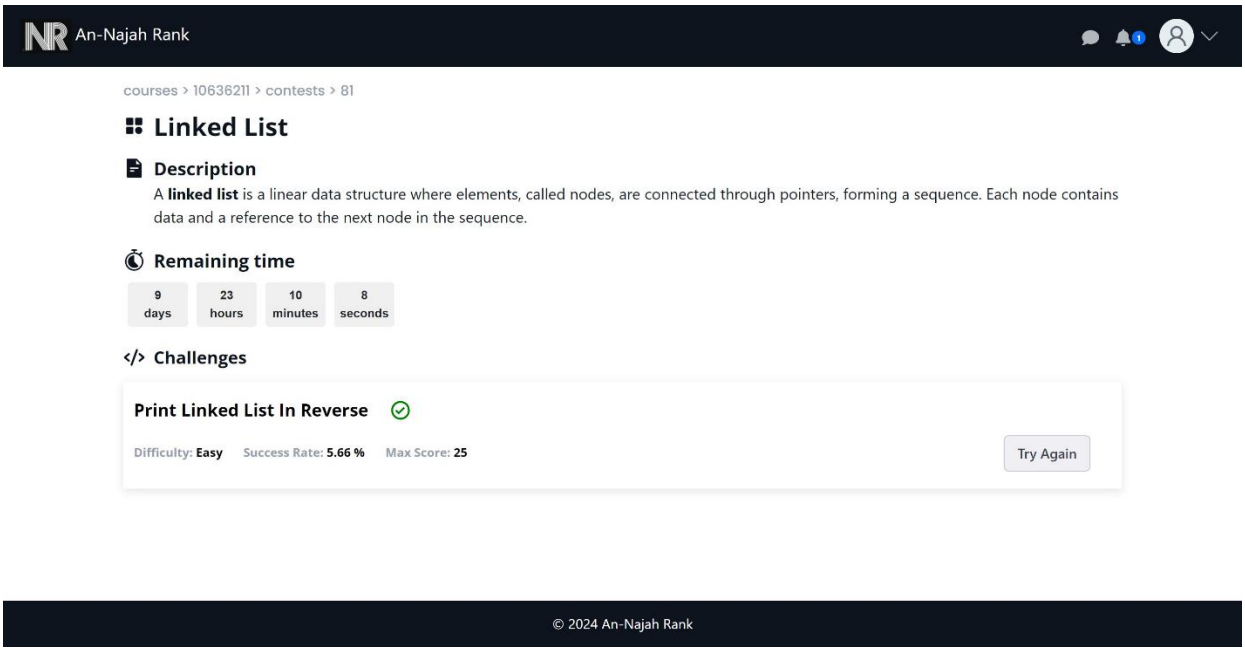
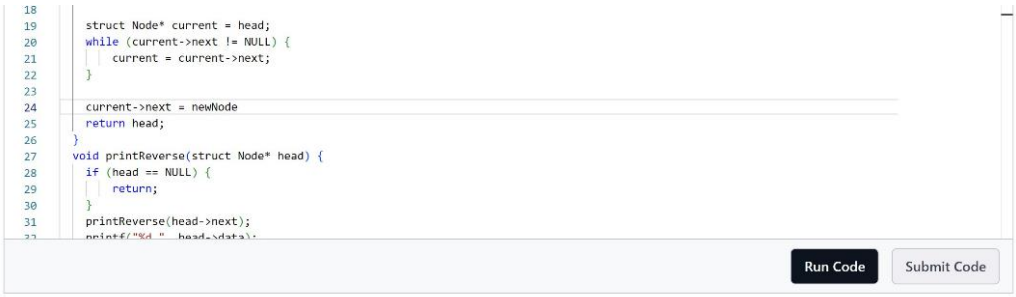


Figure 90: Contest view in student side



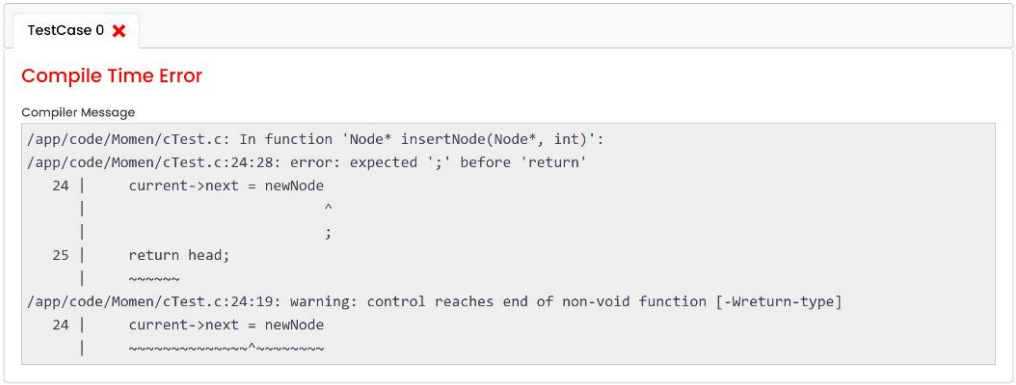


Figure 91: when run code and there is a compile error

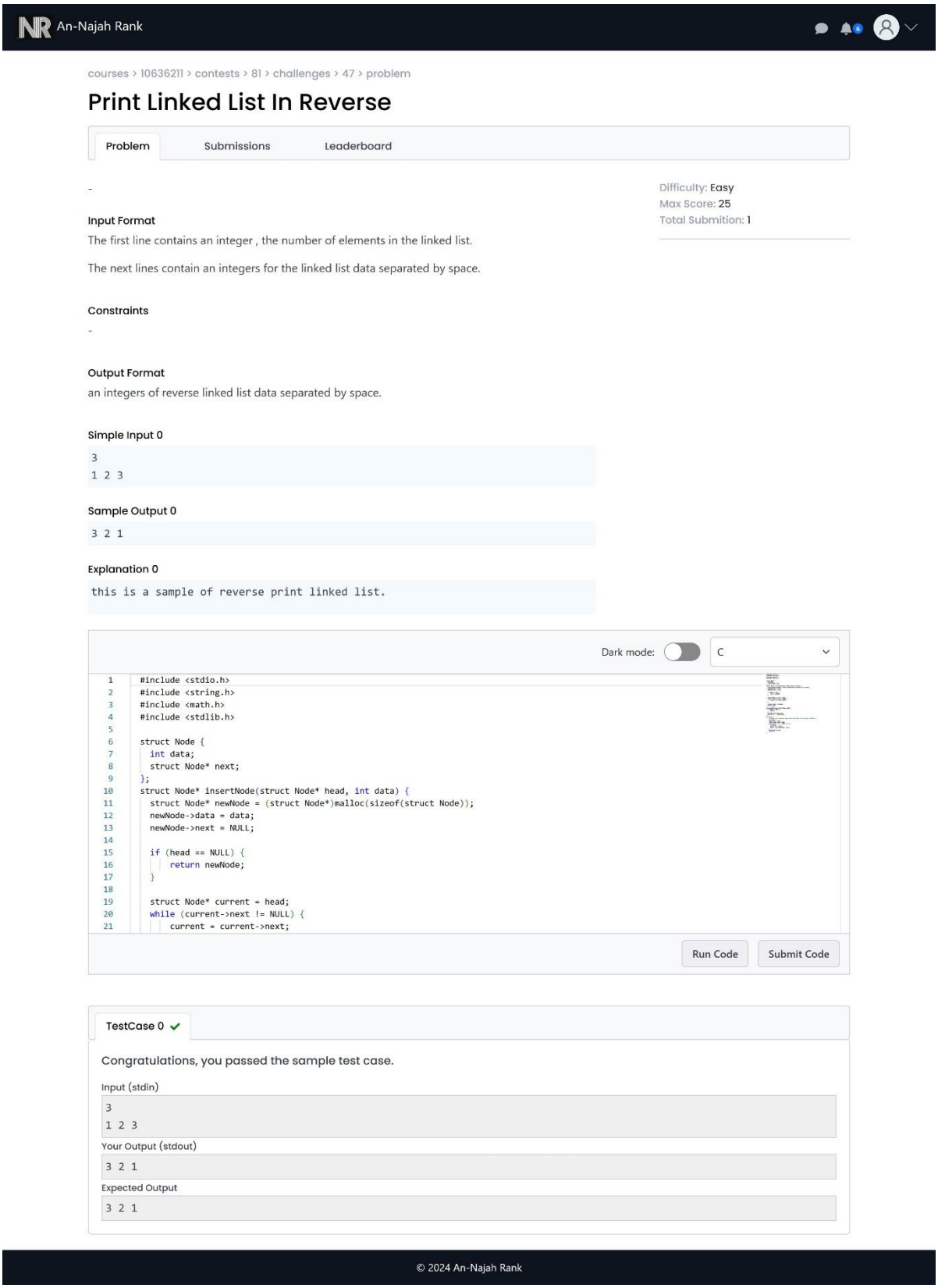


Figure 92: Challenge view and run code in student side

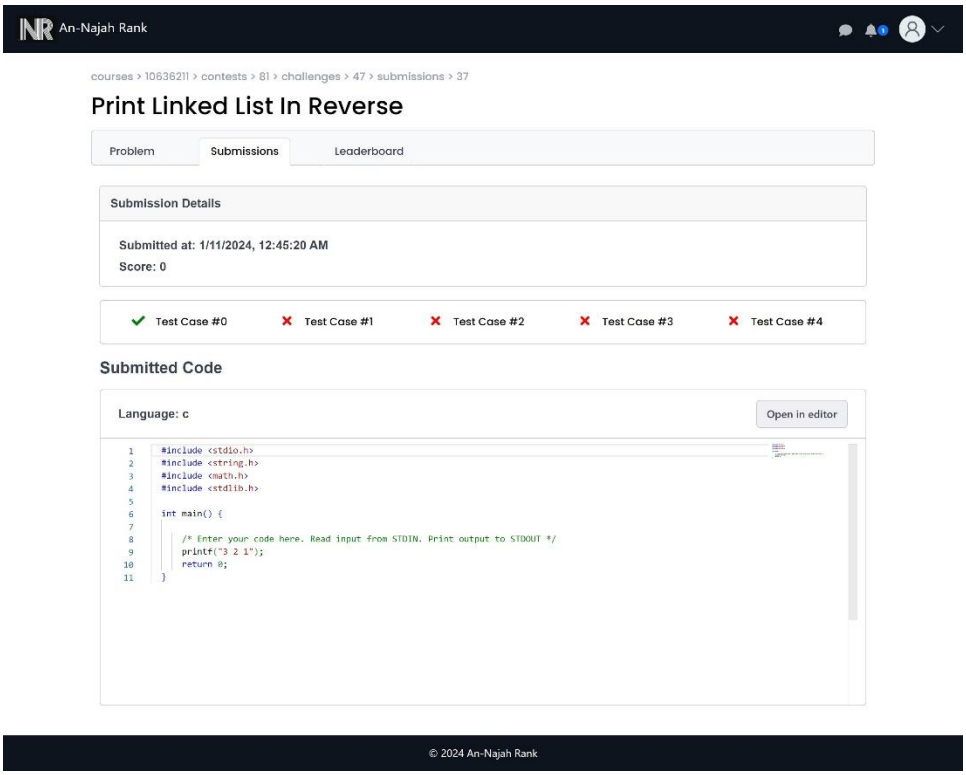


Figure 93: submit code not pass all test cases

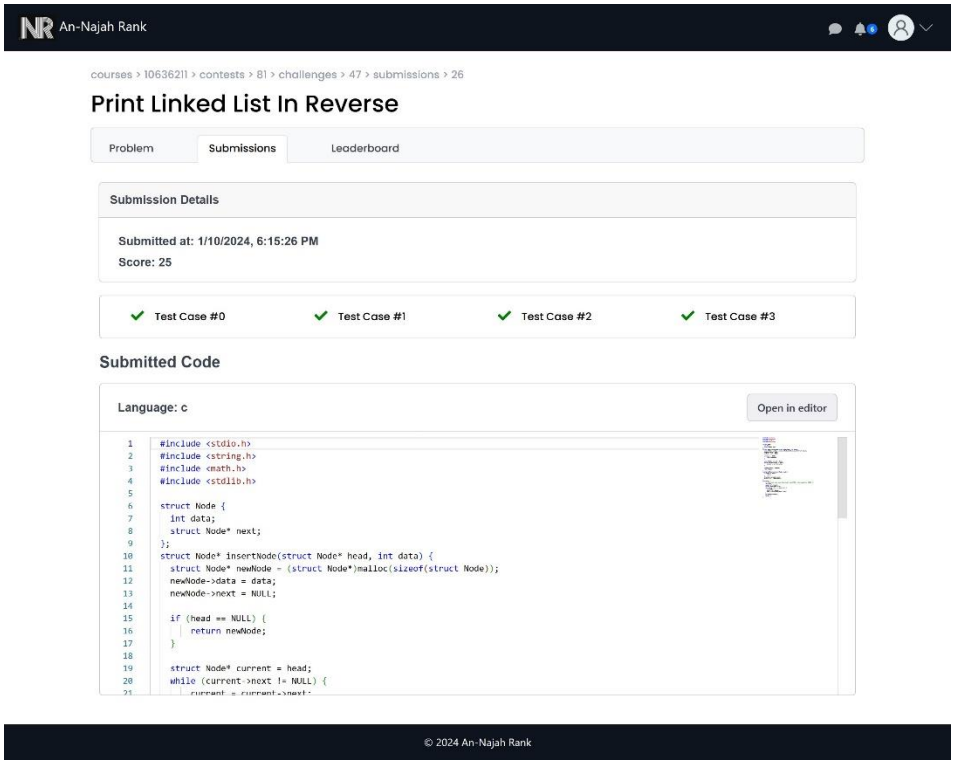


Figure 94: Submit the code

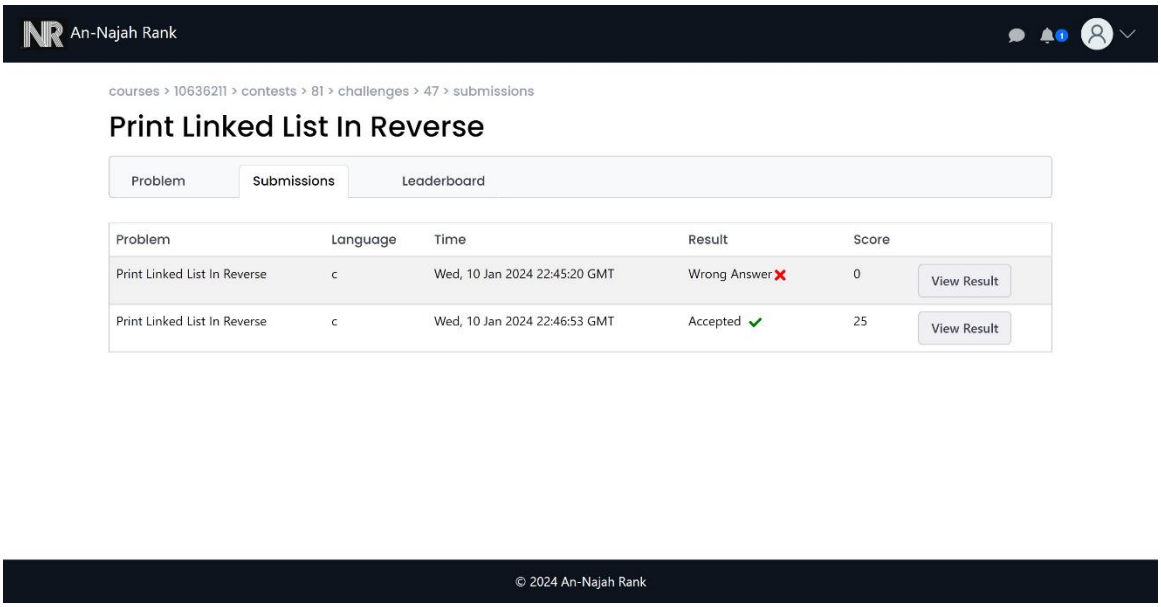


Figure 95: Student submissions in student side

3.4.5 Sample of responsive design:

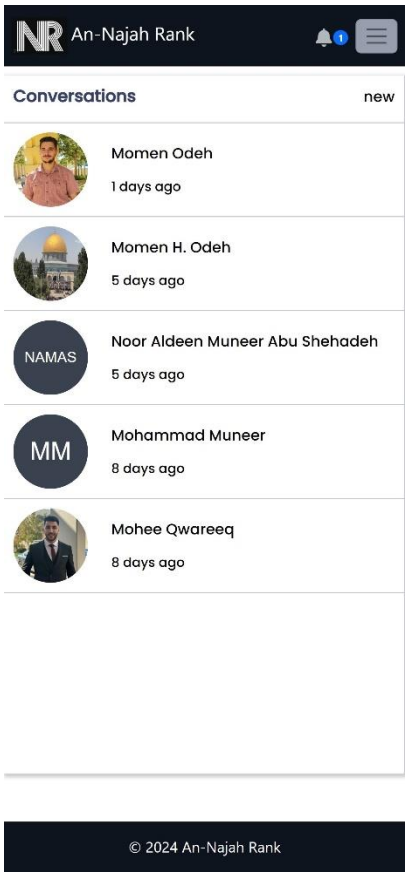


Figure 96: Conversation responsive

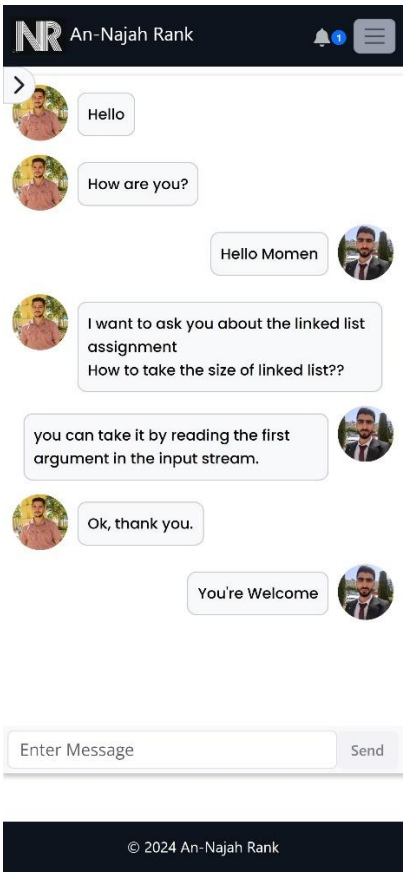


Figure 97: Chatting responsive

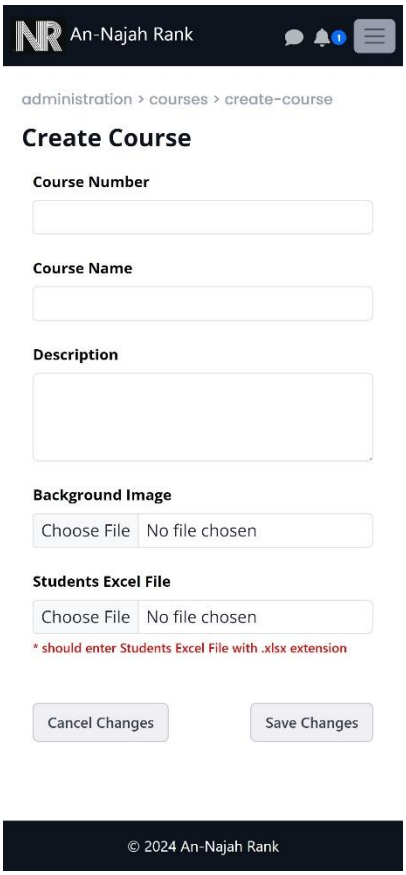


Figure 98: create course responsive

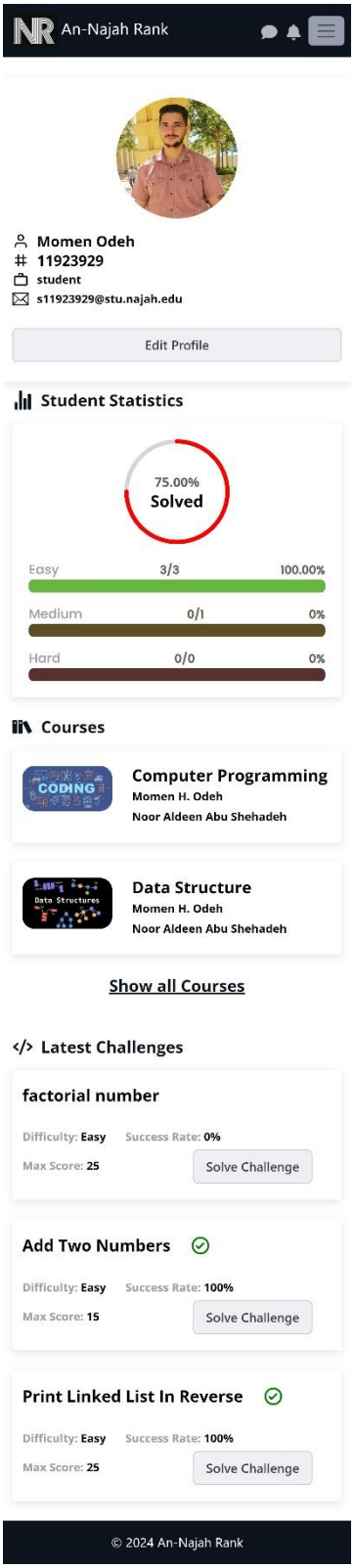


Figure 99: Profile responsive

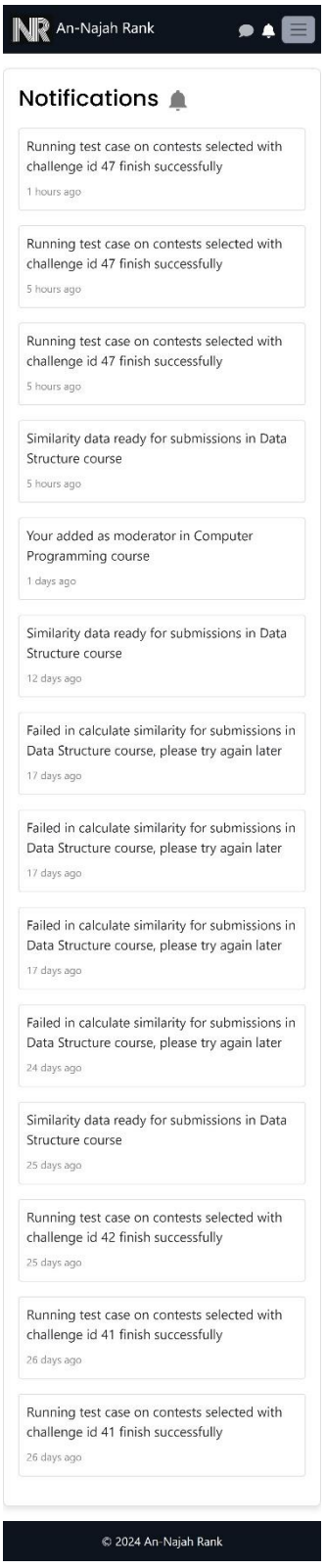


Figure 100: Notification responsive

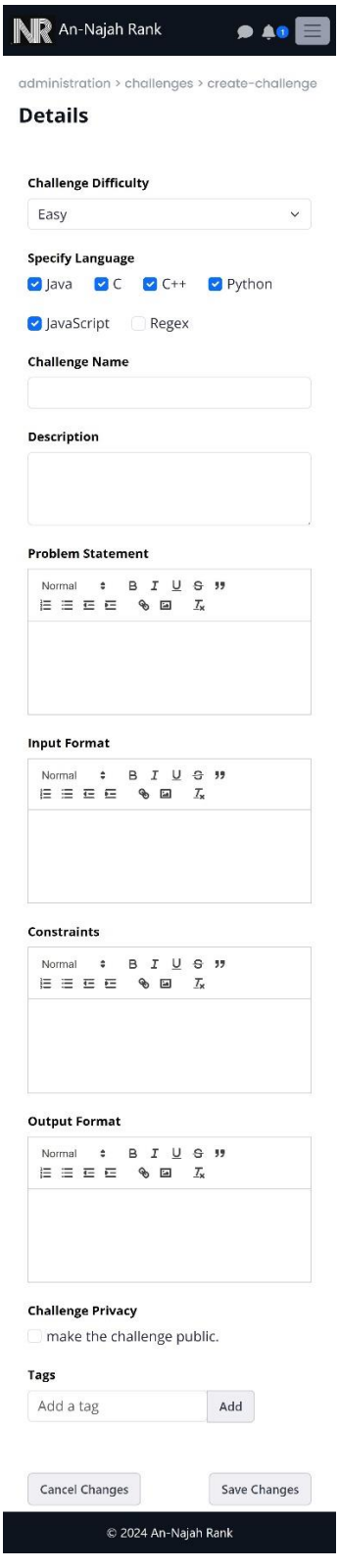


Figure 101: Create challenge responsive

3.4.6 Features details:

3.4.6.1 Socket IO:

Socket.IO is a real-time web application framework that enables bidirectional communication between clients and servers. It is built on top of the WebSocket protocol but provides additional features, such as support for fallback mechanisms like long polling, which ensures compatibility with various browsers and network conditions. Socket.IO simplifies the implementation of real-time, event-driven applications by offering a simple and flexible API for handling communication between the server and connected clients.

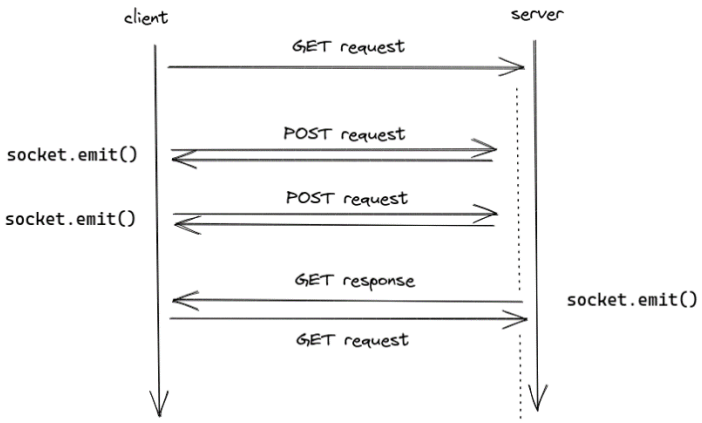


Figure 102: Socket IO

3.4.6.2 Students excel file:

When a professor creates a course, they should upload an Excel file downloaded from Zajel. The Excel file must be in .xlsx format. Subsequently, we extract all student university numbers from this file and add the students to the course.

طباعة قائمة الحضور والغياب

أول 2023 -2024

اسم المساق : أنظمة التشغيل الموزعة(10636456/1) اسم المدرس : سامر ناجح ناجي العرندى

الموعد الزمني : احد/ثلاث/خميس 2-1 رقم القاعة : 111170

رمز الكلية	رقم الطالب	اسم الطالب
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		

Figure 103: Students excel file

3.4.6.3 Code operation:

We have built a backend server capable of compiling and running code. This was achieved by installing the necessary compilers or interpreters for the languages intended to run code. We can now compile and execute code using command lines within our code.

Table 1: Supported languages

Language	Compiler/Interrupter
C/C++	
Java	
Python	
JavaScript	

3.4.6.4 Add new test case after there is a submission for challenge:

The professor is able to add or update test cases. They can then run these test cases on a specific contest selected from the user interface, applying them to all related submissions. The grades are subsequently updated based on the results of the new test cases. Once all operations are completed, a notification is sent to the user, informing them that the operation has finished successfully.

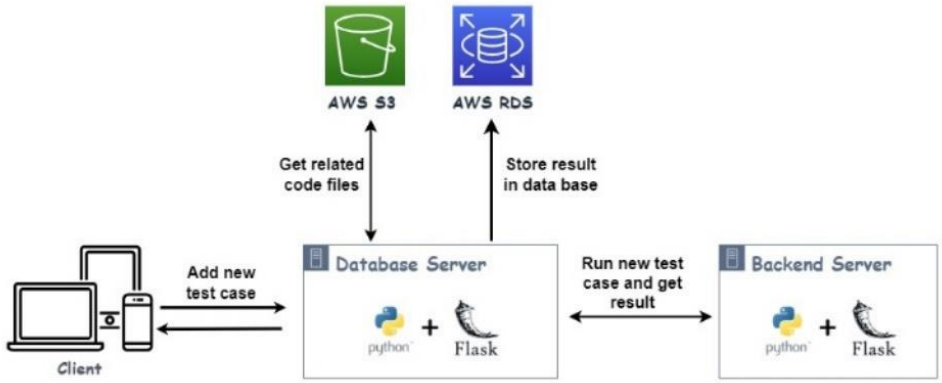


Figure 104: Add new test case diagram

3.4.6.5 Manual mark:

The professor can view all student submissions for a specific challenge. For each user, all submissions can be displayed. Afterward, the professor can manually mark the latest submission by assigning a new grade. To streamline this process, we have added a percent grade for each test case, and the total grade is calculated out of 100.

3.4.6.6 Similarity:

One of the most important features added to the system is calculating code similarity. We decided to use an open-source service to perform this task. Initially, we considered Turnitin, but after thorough research, we discovered that it is not suitable because it is customized for checking text similarity, not coding similarity. Further investigation led us to Moss (Measure Of Software Similarity), an automatic system designed for determining the similarity of programs.

Moss, developed in 1994, stands for Measure Of Software Similarity. It functions as an automatic system specifically tailored for assessing the similarity of programming code. Its primary application has been in detecting plagiarism in programming classes. Unlike general-purpose plagiarism detection tools, Moss is optimized for identifying similarities in coding structures and logic.

The Moss algorithm is considered a significant improvement over other cheating detection algorithms known to date. Users can submit a list of files in various programming languages, and Moss produces HTML pages listing pairs of programs with

similar code. It highlights individual passages in the programs that appear the same, facilitating a quick and efficient comparison of the submitted files.

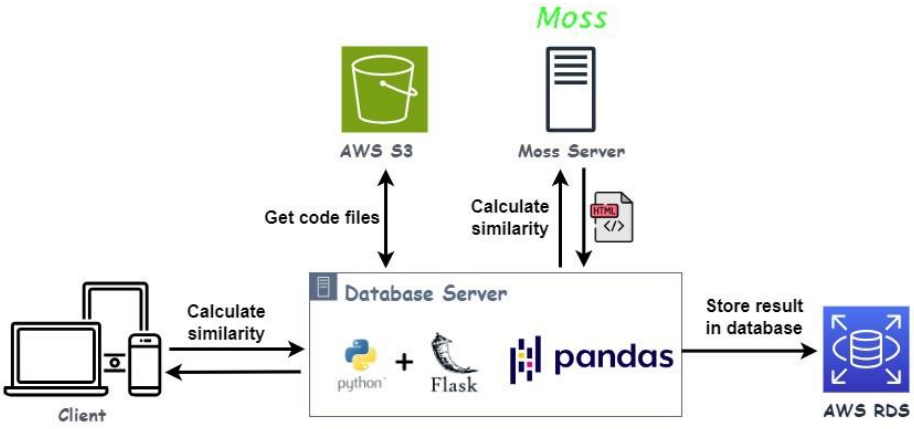
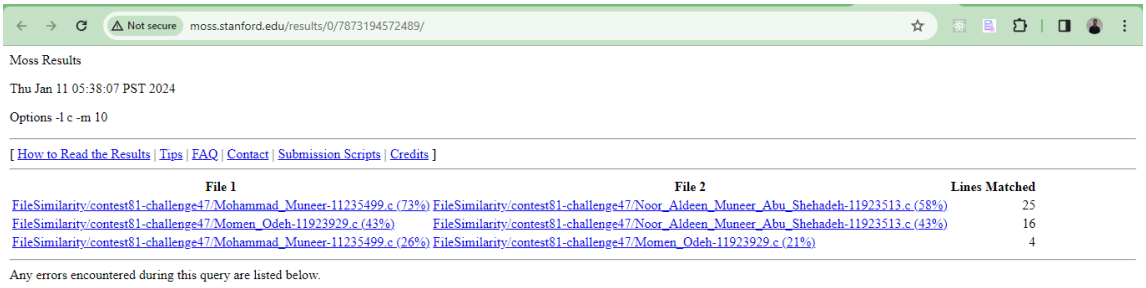


Figure 105: Calculate similarity operation

To use Moss service must send all code files that needs to calculate similarity for it that downloaded from AWS S3 service then after calculate similarity Moss return result as HTML files as shown in figures below:


Not secure moss.stanford.edu/results/0/7873194572489/

Moss Results

Thu Jan 11 05:38:07 PST 2024

Options -l c -m 10

[How to Read the Results | Tips | FAQ | Contact | Submission Scripts | Credits]

File 1	File 2	Lines Matched
FileSimilarity/contest81-challenge47/Mohammad_Muneer-11235499.c (73%)	FileSimilarity/contest81-challenge47/Noor_Aldeen_Muneer_Abu_Shehadeh-11923513.c (58%)	25
FileSimilarity/contest81-challenge47/Momen_Odeh-11923929.c (43%)	FileSimilarity/contest81-challenge47/Noor_Aldeen_Muneer_Abu_Shehadeh-11923513.c (43%)	16
FileSimilarity/contest81-challenge47/Mohammad_Muneer-11235499.c (26%)	FileSimilarity/contest81-challenge47/Momen_Odeh-11923929.c (21%)	4

Any errors encountered during this query are listed below.

Figure 106: Moss similarity result

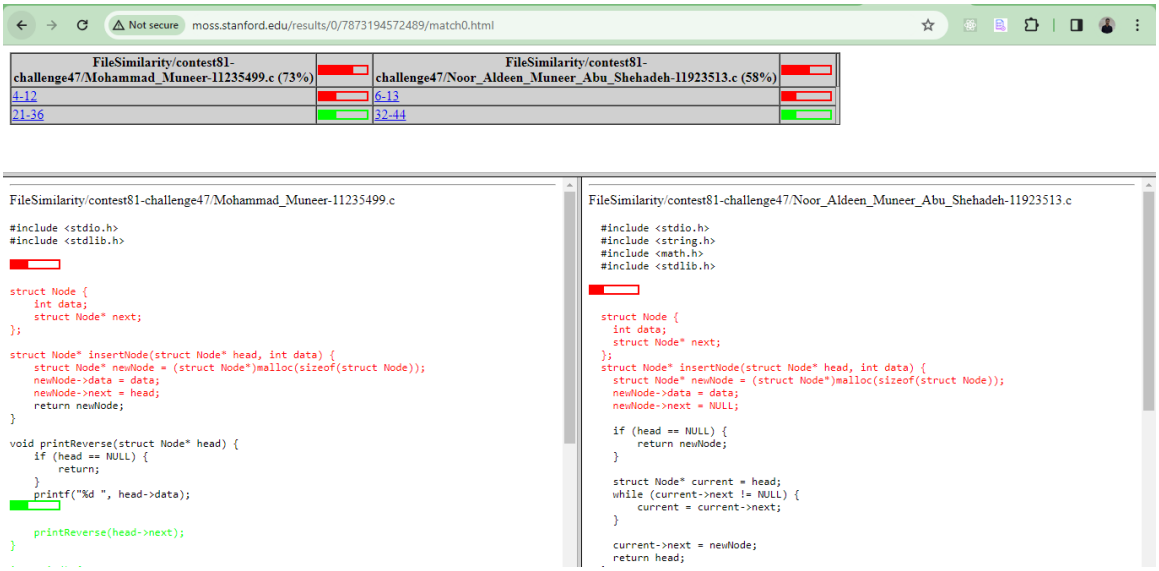


Figure 107: Moss similarity details

Now, we retrieve data from Moss in the form of HTML files and proceed to analyze the data using the **pandas** library. The analyzed data is stored in arrays. Next, we calculate the total similarity for each file by collecting all similarity data associated with that file. We extract all lines with similarity, and the total similarity is computed by dividing the total number of lines with similarity by the total number of lines in the file. Now, after performing these steps for all files, we store the final results in the database. Subsequently, a notification is sent to the professor to inform them that the similarity has been calculated successfully.

3.4.6.7 Statistics:

We have implemented a feature in the student profile that displays the number of problems the student has solved in each category out of the total challenges. Additionally, it shows a success rate, indicating the percentage of solved challenges out of all challenges for that student. In contests and challenges, statistics are provided, showing success rates and maximum scores. Furthermore, on the admin page, there is a statistics section showcasing the best students in the entire system who have solved the largest number of problems.

3.5 Deployment:

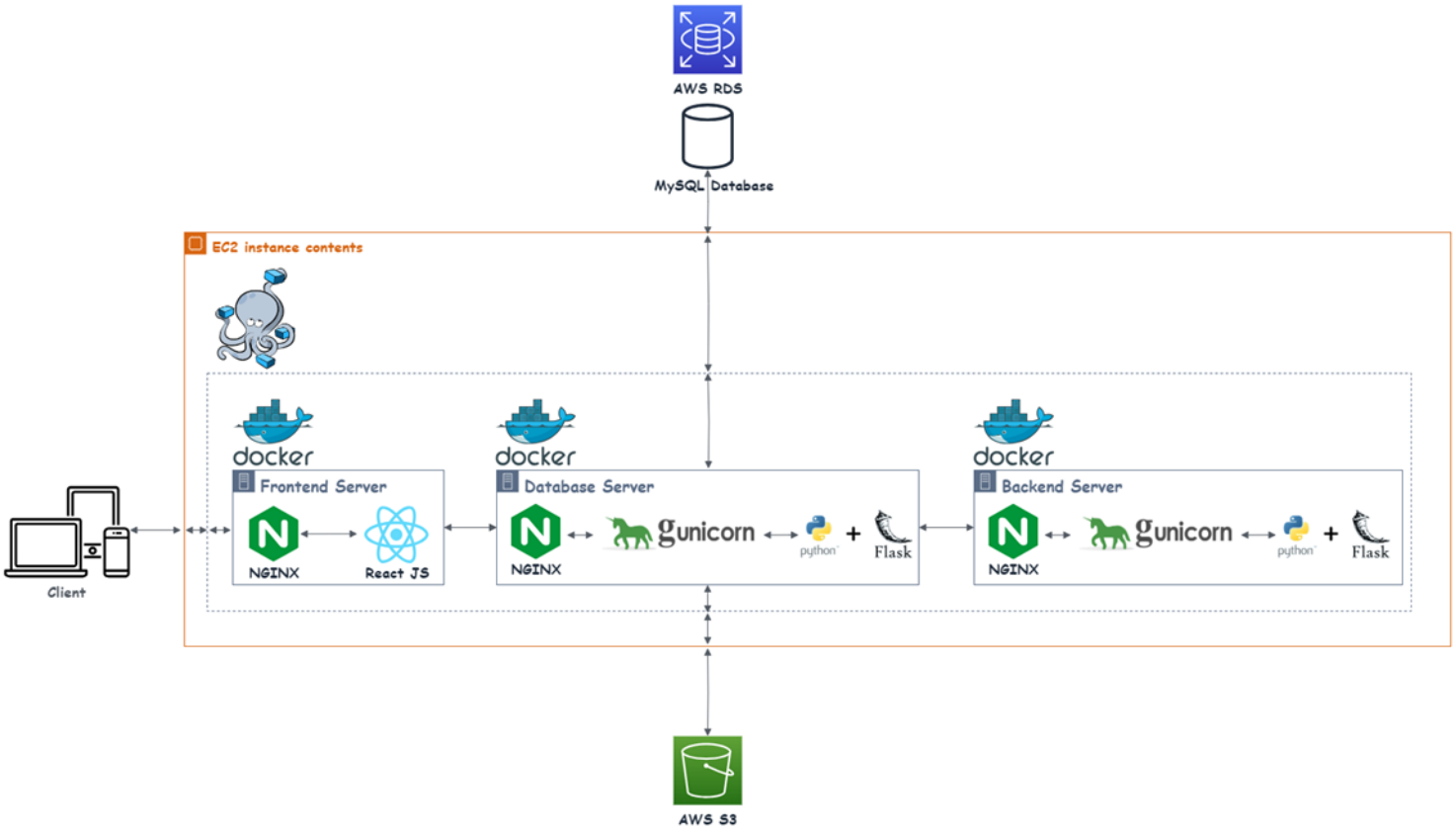


Figure 108: Deployment process

In the deployment structure for the frontend, Nginx serves as a crucial web server, playing a key role in handling the dynamic runtime of the React application.

In the database and backend deployment structure optimized for APIs, Flask serves as the foundational framework, responsible for handling application logic and dynamically generating content. Gunicorn is employed as the WSGI server, efficiently managing communication and concurrent requests through multiple worker processes. While Nginx, traditionally recognized as a reverse proxy for web applications, is considered optional in this API-centric setup, it remains a valuable component for potential load balancing and additional security measures. The refined architecture emphasizes a modular separation of concerns, with Flask managing API routes, Gunicorn overseeing WSGI interactions, and Nginx, when utilized, contributing to load balancing and potential security enhancements. This streamlined structure establishes a dependable, scalable, and secure foundation specifically designed for deploying a Flask backend focused on API functionalities in production environments, combining the strengths of Flask, Gunicorn, and Nginx for optimal performance and security.

3.6 Testing:

Testing is a critical phase in the software development lifecycle aimed at ensuring the quality, reliability, and functionality of a software product. The primary objective of testing is to identify defects or issues within the application, allowing developers to address them before the software is deployed to end-users. After implementing the project, we conduct manual testing for all features in the system to ensure that all features work correctly.

Table 2: Manul testing table

Feature Name	Status	Failure Description
Sign Up	Pass	
Verification Code	Fail	Error solved when the role is a professor; fetching result two times causes an error.
Admin Approve Professor	Pass	
Forget Password	Pass	
Update Info	Fail	Error occurs when updating or adding an image, no synchronization between events.
Create Challenge	Pass	
Update Challenge	Fail	Error when updating tags, and initially, there are no tags.
Add Test Case	Pass	
Update Test Case	Pass	
Remove Test Case	Pass	
Create Course	Pass	
Create Contest	Pass	
Update Course	Pass	
Update Contest	Pass	
Add Challenge in Contest	Pass	
Get All Courses	Fail	When the role is a professor, there is nothing to retrieve; retrieves courses for students when there are no moderators. Update SQL statements to retrieve course moderators only, not the owner.
Get Specific Course	Pass	
Get Specific Contest	Pass	

Get Specific Challenge	Pass	
Notification	Pass	
Add Moderator	Pass	
Remove Moderator	Pass	
Run Code	Pass	
Submit Right Code	Fail	Problem retrieving student university number in the backend.
Profile Statistics	Fail	Problem when there is more than one submission pass.
Contests Statistics	Fail	Problem when there is more than one submission pass.
When No Sample Test Case	Pass	
Submit Compile Error Code	Pass	
Manual Mark	Pass	
Submit When Time Ends	Pass	
Update End Time of Contest	Fail	Problem appears when the start time is greater than the current time.
Calculate Similarity	Fail	Error 'NoneType' object is not subscriptable (the /file-Similarity API in excluded_routes).
Add Test Case When There Are Submissions	Pass	
Chatting	Pass	

After that we resolve the failures then we deploy the project again and ensure all features works correctly.

3.7 Constraints:

In our AWS environment, not all services come without costs; certain services like RDS, EC2, and Elastic IP Addresses require payment. Additionally, there are limitations associated with the free tier services. Another challenge we face involves a third-party API we use for similarity calculations. This API returns data in an incompatible format, necessitating thorough analysis and refactoring to align it with our requirements.

Chapter 4: Result and Analysis

In our project, we have successfully developed a user-friendly problem-solving web application that stands out in competition with other similar platforms. The implementation of this web application provides our professors with a seamless mechanism to effortlessly track student submissions. They can also calculate the similarity of student submissions with ease by simply clicking a button, revealing similarity scores for all submission files. Moreover, the system facilitates professors in adding new test cases even after submissions have been received for a specific challenge. When incorporating a new test case, professors have the flexibility to choose the contest for which they want to run the code on existing submissions.

Furthermore, the system extends its functionality by allowing professors to perform manual marking for the last submission of each student. Alongside these advanced features, our application offers fundamental capabilities such as creating challenges, contests, and courses, and effectively managing them. The inclusion of notification features ensures timely updates for both professors and students, enabling users to track new events. Additionally, the application provides a chat feature for seamless communication. Students can submit code and monitor their submissions.

Chapter 5: Discussion

In our project, the software development life cycle starts with planning and collecting requirements, continuing through testing and deployment. Throughout these phases, we adhere to the agile methodology. We establish criteria for maintainability, scalability, and other key attributes to ensure the software meets the highest standards. This iterative and collaborative approach allows us to respond effectively to changing requirements and deliver a product that aligns with both user expectations and industry best practices.

In traditional deployment processes, migrating servers or switching hosting providers typically involves laborious server configuration, including the setup of dependencies and ensuring compatibility. This procedure is not only time-consuming but also prone to errors.

Docker streamlines this process by encapsulating the application and its dependencies into a container, along with a Dockerfile specifying the necessary environment. This containerization results in a standardized and reproducible deployment environment. When transitioning to a new server or changing hosting providers, deploying the Docker container simplifies the deployment task, ensuring that the application runs consistently across diverse environments.

Essentially, Docker abstracts the intricacies of the underlying infrastructure, offering a more portable and efficient method for deploying applications. This proves especially advantageous when quick transitions or replications of deployment environments are required, saving time and mitigating the risks associated with configuration discrepancies.

Chapter 6: Conclusions and Recommendation

In conclusion, the process of building a web application is a multifaceted journey that traverses various crucial phases, each playing a pivotal role in the project's success. From the planning stages to the final deployment, every step demands a substantial investment of effort and effective management processes. The challenges encountered throughout the development cycle underscore the necessity of a well-structured approach, emphasizing the importance of adhering to best practices.

A cornerstone of successful web application development lies in robust requirements gathering and meticulous planning. This foundational phase sets the tone for subsequent stages, aligning the development team with project goals. The design and prototyping stages further refine the vision, ensuring that the application's interface and user experience resonate seamlessly with end-users.

During the development phase, the actual coding and programming come to life. The adoption of agile methodologies enhances adaptability, fostering iterative development to accommodate evolving requirements. good testing is integral to identifying and rectifying issues, ensuring a reliable and bug-free application.

Deployment marks the culmination of development efforts, releasing the application to users. Automation and containerization technologies streamline this process, promoting consistency across different environments. Post-deployment, a continuous feedback loop, along with iterative improvements, enables the application to evolve in response to user needs.

In essence, by following a well-structured and adaptive approach, incorporating best practices, and prioritizing user feedback, developers can realize a web application's full potential and deliver enhanced value in a shorter timeframe. The challenges inherent in the development journey become opportunities for growth, leading to the creation of a resilient and user-centric web application.

Future Works :

1- Support time complexity as constrain for challenge.
2- Support creating a challenge related to image processing.

References

- **React.** Available: https://react.dev/learn .
- **React Bootstrap**. Available: https://react-bootstrap.netlify.app/docs/getting-started/introduction .
- **React JSS**. Available: https://cssinjs.org/react-jss/?v=v10.3.0 .
- **React Router.** Available: https://reactrouter.com/en/main/start/tutorial .
- **Flask python**. Available: https://flask.palletsprojects.com/en/3.0.x/ .
- **Pandas**. Available: https://pandas.pydata.org/docs/user_guide/index.html .
- **flask SocketIO**. Available: https://flask-socketio.readthedocs.io/en/latest/ .
- **Docker**. Available : https://docs.docker.com/guides/get-started/ .
- **Docker Compose**. Available: https://docs.docker.com/compose/ .
- **AWS**. Available: https://docs.aws.amazon.com/?nc2=h_ql_doc_do .
- **AXIOS library**. Available: https://axios-http.com/docs/intro .
- **Flask Mai**. Available : https://pythonhosted.org/Flask-Mail/ .
- **NGINX**. Available : https://nginx.org/en/docs/?_ga=2.53640360.1813303284.1705009802-144038734.1703448924 .
- **GUNICORN**. Available : https://docs.gunicorn.org/en/stable/ .
- **pyJWT**. Available : https://pyjwt.readthedocs.io/en/stable/ .
- **Flask-Cors**. Available : https://flask-cors.readthedocs.io/en/latest/ .
- **Moss**. Available : https://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf .