

بِسْمِ اللَّهِ الرَّحْمَنِ 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

| | |
|---|----|
| Dedication | 2 |
| Acknowledgment | 3 |
| Disclaimer | 4 |
| Table of content | 5 |
| List of Figures | 6 |
| List of tables..... | 9 |
| Abstract..... | 10 |
| Chapter 1: Introduction | 11 |
| ▪ Statement of the problem | 11 |
| ▪ Objective | 12 |
| ▪ Scope of the work | 13 |
| ▪ Importance of the work..... | 14 |
| ▪ Organization of the report | 15 |
| Chapter 2: Theoretical Background and Previous Work | 16 |
| Chapter 3: Methodology | 17 |
| 3.1 Planning:..... | 18 |
| 3.2 Analysis:..... | 18 |
| 3.3 Design: | 21 |
| 3.4 Implementation: | 30 |
| 3.5 Deployment:..... | 70 |
| 3.6 Testing:..... | 71 |
| 3.7 Constraints:..... | 73 |
| Chapter 4: Result and Analysis..... | 74 |
| Chapter 5: Discussion | 75 |
| Chapter 6: Conclusions and Recommendation | 76 |
| Future Works : | 77 |
| References..... | 78 |

List of Figures

| | |
|---|----|
| Figure 1: Software Development Life Cycle | 17 |
| Figure 2: Agile Methodology | 17 |
| Figure 3: UML Diagram | 19 |
| Figure 4: User Stories | 20 |
| Figure 5: React | 21 |
| Figure 6: React Bootstrap | 21 |
| Figure 7: React JSS | 21 |
| Figure 8: React Router | 22 |
| Figure 9: Flask python | 22 |
| Figure 10: MySQL | 22 |
| Figure 11: Pandas | 23 |
| Figure 12: SocketIO | 23 |
| Figure 13: GitHub | 23 |
| Figure 14: Trello | 23 |
| Figure 15: Docker | 24 |
| Figure 16: Docker Compose | 24 |
| Figure 17: AWS CloudFormation | 24 |
| Figure 18: AWS EC2 | 25 |
| Figure 19: AWS S3 | 25 |
| Figure 20: AWS RDS | 25 |
| Figure 21: VS Code | 25 |
| Figure 22: pycharm | 26 |
| Figure 23: MySQL Workbench | 26 |
| Figure 24: Postman | 26 |
| Figure 25: Docker Desktop | 26 |
| Figure 26: Draw io | 26 |
| Figure 27: Restful Architectural Style | 27 |
| Figure 28: AXIOS library | 27 |
| Figure 29: Microservice Architectural pattern | 27 |
| Figure 30: Project Structure | 28 |
| Figure 31: Flask Mai | 29 |
| Figure 32: pyJWT | 29 |
| Figure 33: Flask-Cors | 29 |
| Figure 34: sign up | 30 |
| Figure 35: verification code | 30 |
| Figure 36: email verification message | 31 |
| Figure 37: log in | 31 |
| Figure 38: forget password | 32 |
| Figure 39: verification code for reset password | 32 |
| Figure 40: verification code message for reset password | 32 |
| Figure 41: set new password | 32 |
| Figure 42: User account settings | 33 |
| Figure 43: Password settings | 33 |

| | |
|---|----|
| Figure 44: Create new Chat | 34 |
| Figure 45: Chatting notification..... | 34 |
| Figure 46: See chatting notification pop-up..... | 35 |
| Figure 47: Chatting conversations | 35 |
| Figure 48: pending professors in admin page | 36 |
| Figure 49: approve professor | 36 |
| Figure 50: all professors in the system..... | 37 |
| Figure 51: all students in the system..... | 37 |
| Figure 52: Students statistics | 38 |
| Figure 53: Viewing student profiles from the admin side..... | 38 |
| Figure 54: Manage Courses in administration page..... | 39 |
| Figure 55: Create course | 39 |
| Figure 56: Manage course details | 40 |
| Figure 57: Manage course moderators..... | 40 |
| Figure 58: Manage students in course..... | 41 |
| Figure 59: Manage contests | 41 |
| Figure 60: Create contest | 42 |
| Figure 61: Manage contest details | 42 |
| Figure 62: Add challenge to contest | 43 |
| Figure 63: Manage challenges in contest..... | 43 |
| Figure 64: Manage challenges | 44 |
| Figure 65: Create challenge | 44 |
| Figure 66: Manage challenge details | 45 |
| Figure 67: Add test case to challenge | 45 |
| Figure 68: Manage test cases in challenge | 46 |
| Figure 69: All courses page | 46 |
| Figure 70: Course view | 47 |
| Figure 71: Manage students in course..... | 47 |
| Figure 72: Contest view | 48 |
| Figure 73: Contest Grades..... | 48 |
| Figure 74: Problem description..... | 49 |
| Figure 75: Students submissions from professor side..... | 49 |
| Figure 76: View student submissions | 50 |
| Figure 77: Last submission for student can do manual mark..... | 51 |
| Figure 78: Start calculate similarity | 51 |
| Figure 79: Received notification when similarity calculated..... | 52 |
| Figure 80: Code similarity page..... | 52 |
| Figure 81: Code Similarity view 1 | 53 |
| Figure 82: Code similarity view 2 | 53 |
| Figure 83: students leaderboards | 54 |
| Figure 84: Add new test case when there is a submission for challenge | 54 |
| Figure 85: After add test case and run it on all student submission..... | 55 |
| Figure 86: The submission after add new test case..... | 55 |
| Figure 87: Student profile from student side | 56 |

| | |
|---|----|
| Figure 88: Notifications when add new course or contest or challenge..... | 57 |
| Figure 89: All notification page | 57 |
| Figure 90: Course view before contest start in student side..... | 58 |
| Figure 91: Course view after contest start in student side | 58 |
| Figure 92: Contest view in student side | 59 |
| Figure 93: when run code and there is a compile error..... | 59 |
| Figure 94: Challenge view and run code in student side | 60 |
| Figure 95: submit code not pass all test cases..... | 61 |
| Figure 96: Submit the code | 61 |
| Figure 97: Student submissions in student side | 62 |
| Figure 98: Conversation responsive..... | 63 |
| Figure 99: Chatting responsive | 63 |
| Figure 100: create course responsive..... | 63 |
| Figure 101: Profile responsive | 64 |
| Figure 102: Notification responsive..... | 64 |
| Figure 103: Create challenge responsive | 64 |
| Figure 104: Socket IO | 65 |
| Figure 105: Students excel file..... | 65 |
| Figure 106: Add new test case diagram | 67 |
| Figure 107: Calculate similarity operation..... | 68 |
| Figure 108: Moss similarity result | 68 |
| Figure 109: Moss similarity details..... | 69 |
| Figure 110: Deployment process | 70 |

List of tables

| | |
|------------------------------------|----|
| Table 1: Supported languages | 66 |
| Table 2: Manul testing table | 71 |

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

CHAPTER

1

- **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 traverse 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

CHAPTER **2**

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

CHAPTER

3

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.

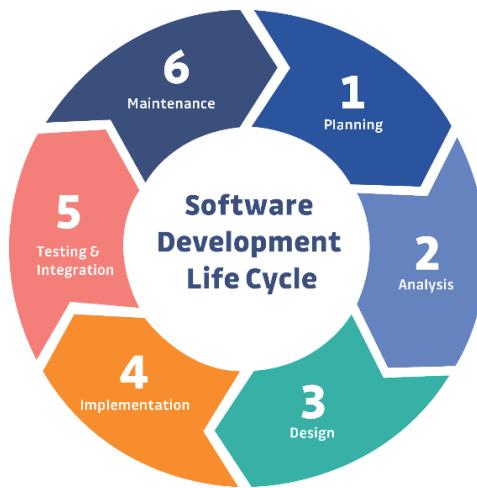


Figure 1: Software Development Life Cycle

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 supervisor, 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:

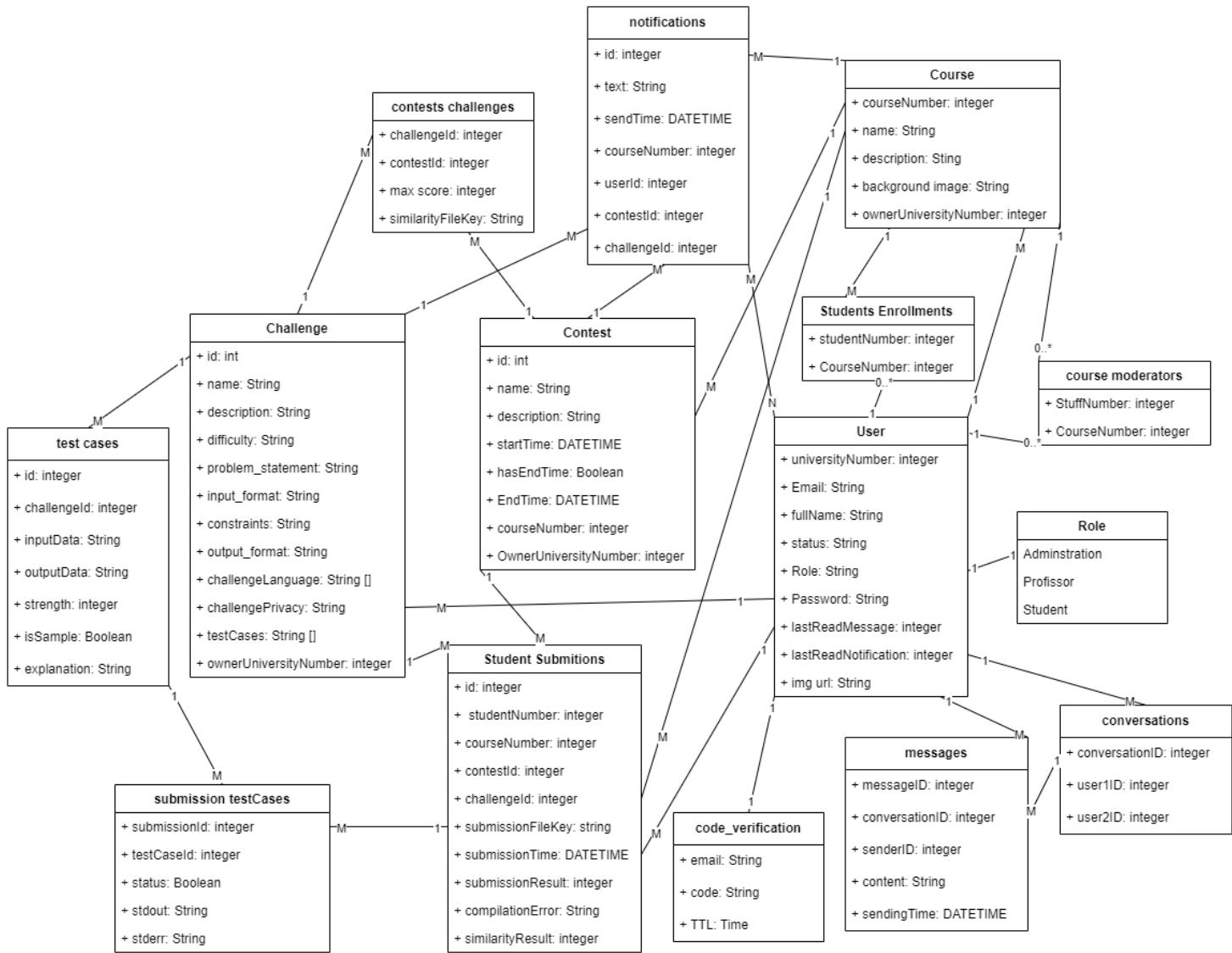


Figure 3: UML Diagram

3.2.2 User Stories:

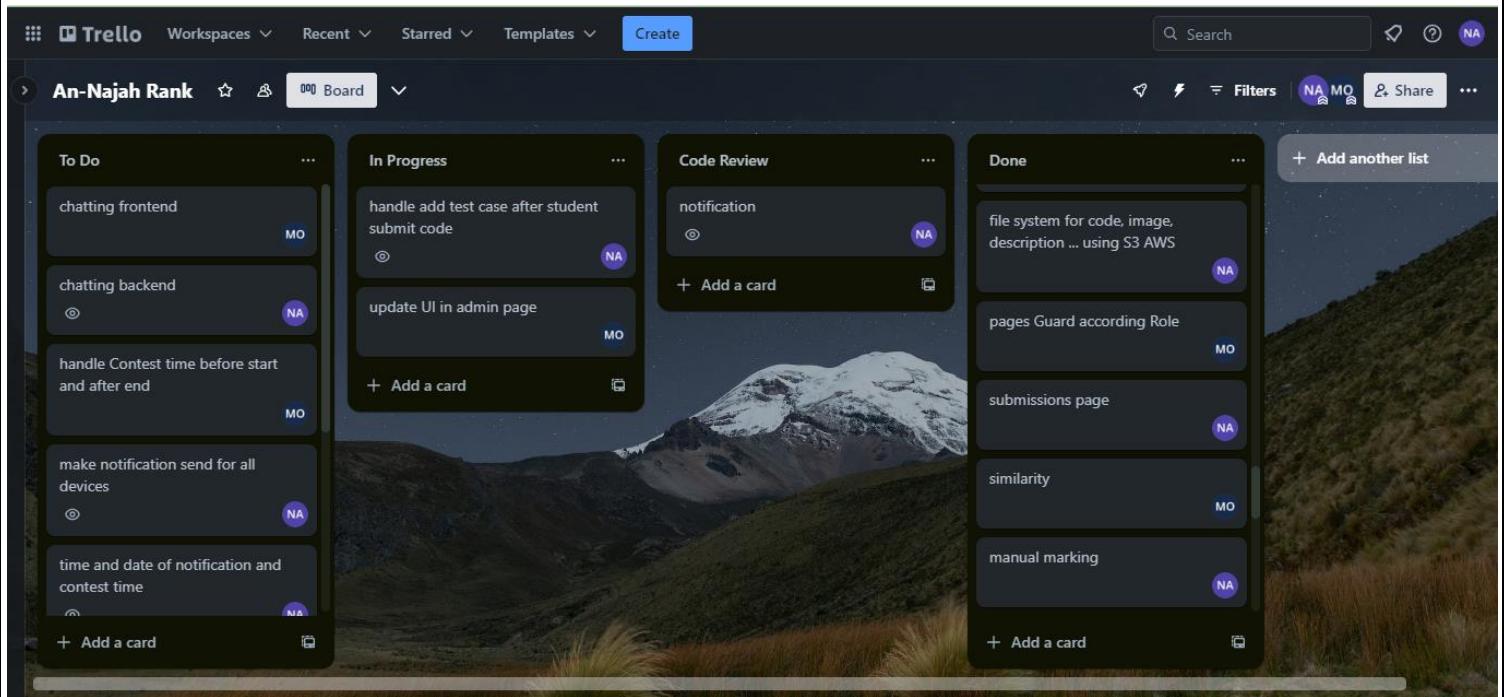


Figure 4: User Stories

3.3 Design:

3.3.1 Tools:

3.3.1.1 Frontend tools:

3.3.1.1 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.

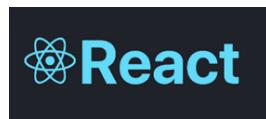


Figure 5: React

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 props.



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 MySQL Database:

MySQL is an open-source relational database management system (RDBMS).



Figure 10: MySQL

3.3.1.2.3 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 11: Pandas

3.3.1.2.4 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 12: 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 13: 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: 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.

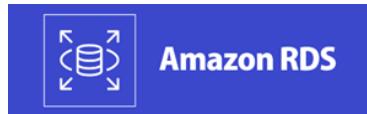


Figure 20: AWS RDS

3.3.1.5 Development tools:

3.3.1.5.1 VS Code:

Used for React development.



Figure 21: VS Code

3.3.1.5.2 pycharm:

Used for Flask development.



Figure 22: pycharm

3.3.1.5.3 MySQL Workbench:

Used for building and monitoring database.



Figure 23: MySQL Workbench

3.3.1.5.4 Postman:

Used for test backend APIs.



Figure 24: Postman

3.3.1.5.5 Docker Desktop:

Used for managing images and containers.



Figure 25: Docker Desktop

3.3.1.5.6 Draw io:

Used for design UML diagram.



Figure 26: 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.

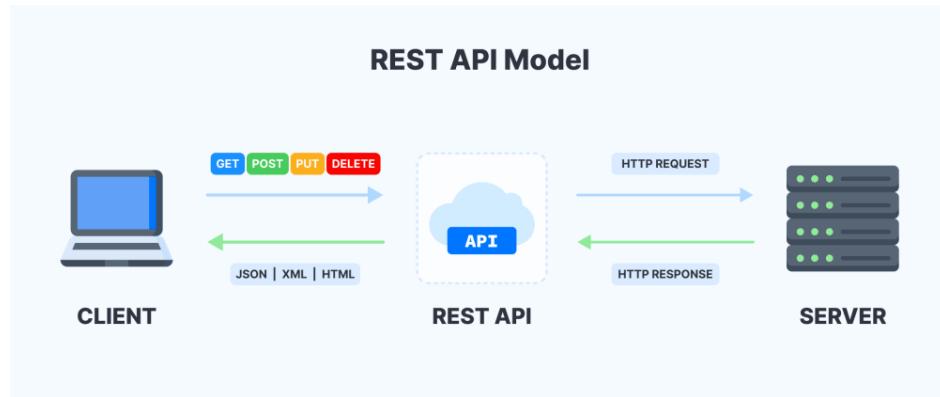


Figure 27: Restful Architectural Style

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 28: 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.

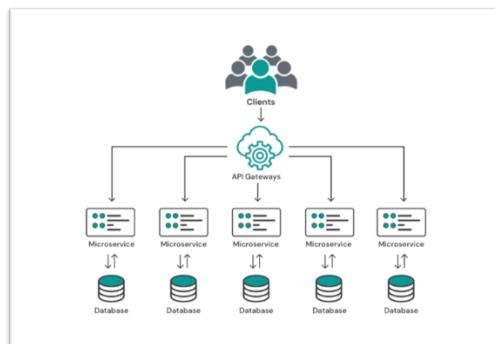


Figure 29: Microservice Architectural pattern

3.3.2.3 Project Structure:

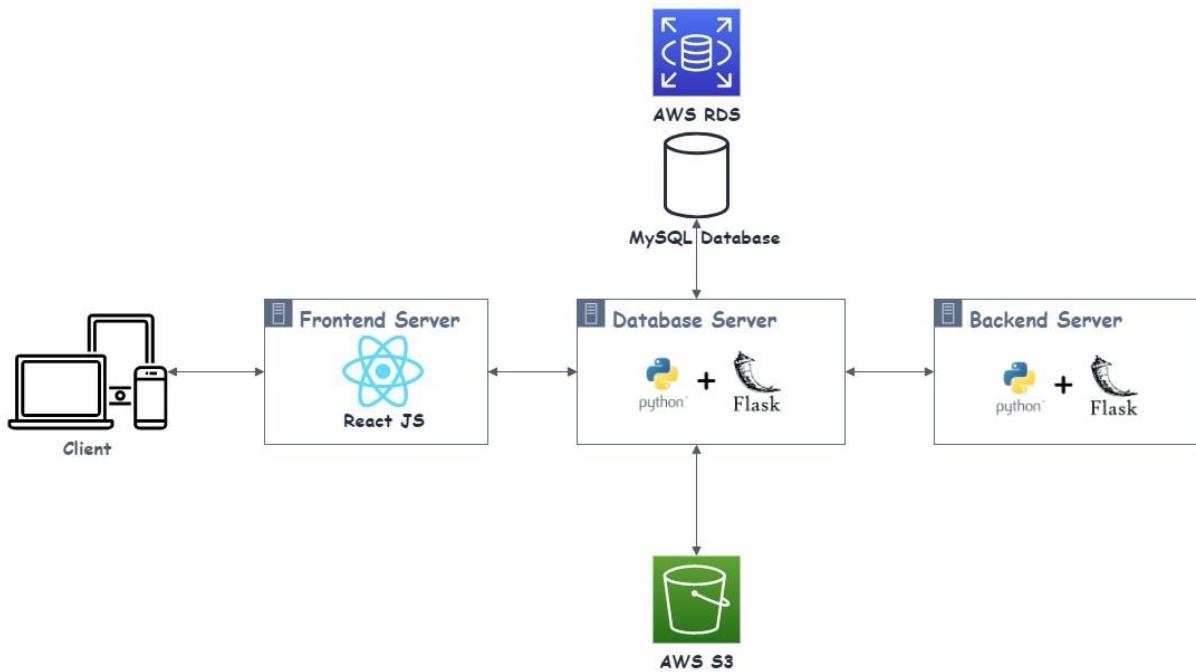


Figure 30: Project Structure

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 logs 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 Authorization:

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 31: Flask Mai



Figure 32: pyJWT



Figure 33: 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.

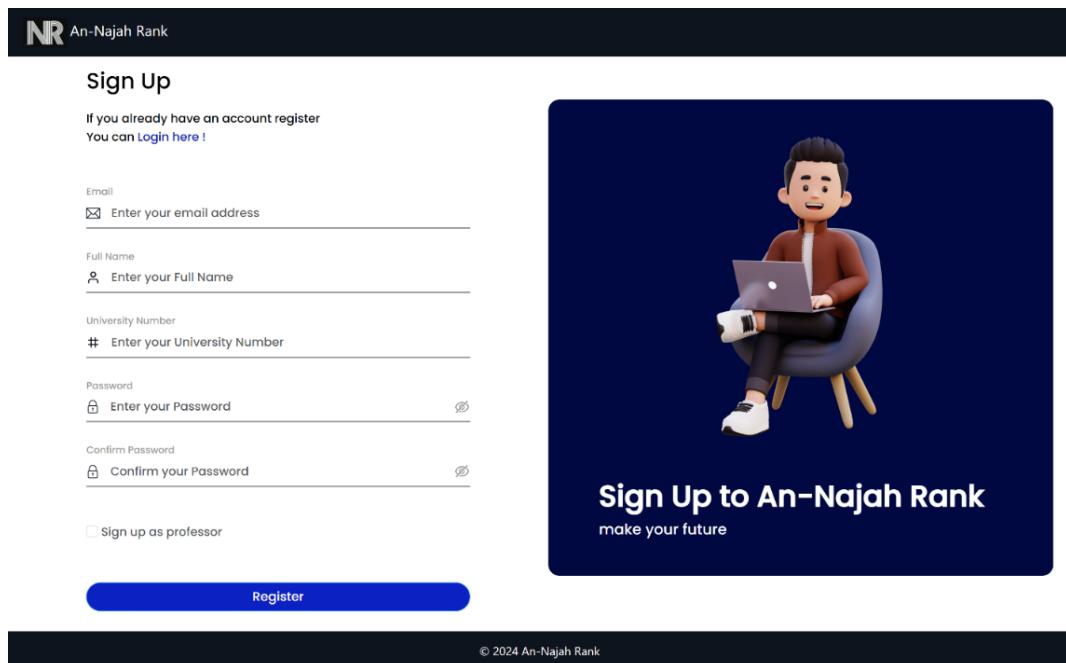


Figure 34: sign up

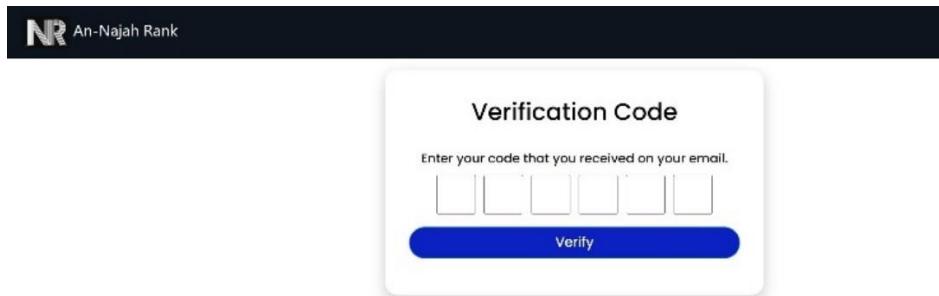


Figure 35: verification code

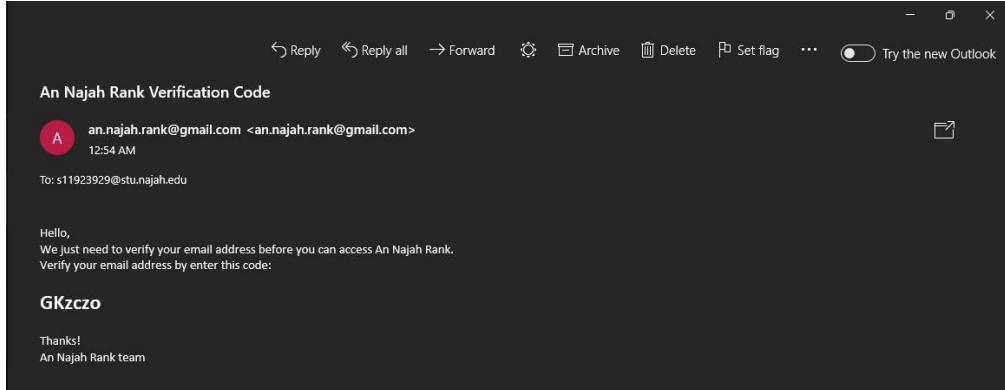


Figure 36: email verification message

3.4.1.2 Sign in:

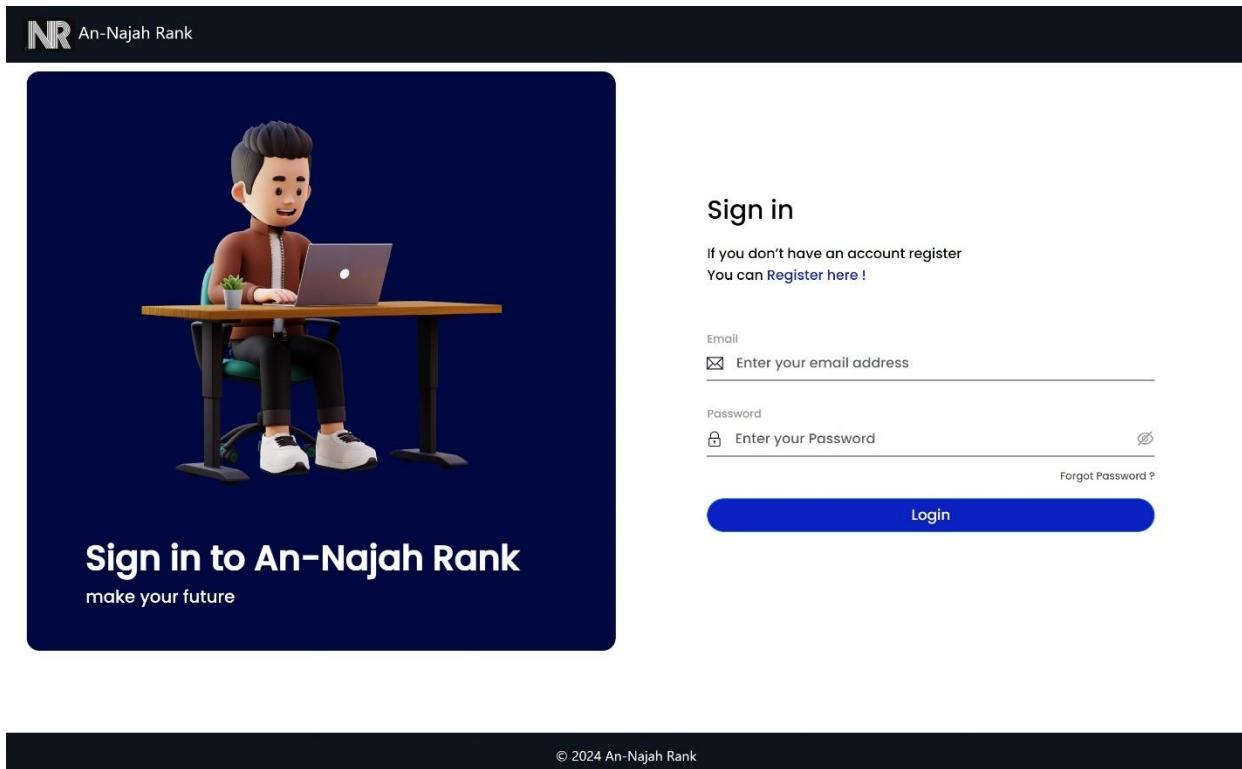


Figure 37: log in

3.4.1.3 Forget password:

The screenshot shows a 'Forget Password' page. At the top, there is a header with the logo 'NR An-Najah Rank'. Below the header, the title 'Forget Password' is centered. A message below the title reads: 'No Problem! Enter your email below and we will send you an Code with instruction to reset your password.' There is an input field containing the email address 's11923929@stu.najah.edu'. Below the input field is a blue 'Reset Password' button. At the bottom of the form, there is a link 'Back to Login'.

Figure 38: forget password

The screenshot shows a 'Verification Code' page. At the top, there is a header with the logo 'NR An-Najah Rank'. Below the header, the title 'Verification Code' is centered. A message below the title reads: 'Enter your code that you received on your email.' Below this message is a row of six empty input boxes for entering a six-digit verification code. Below the input boxes is a blue 'Verify' button.

Figure 39: verification code for reset password

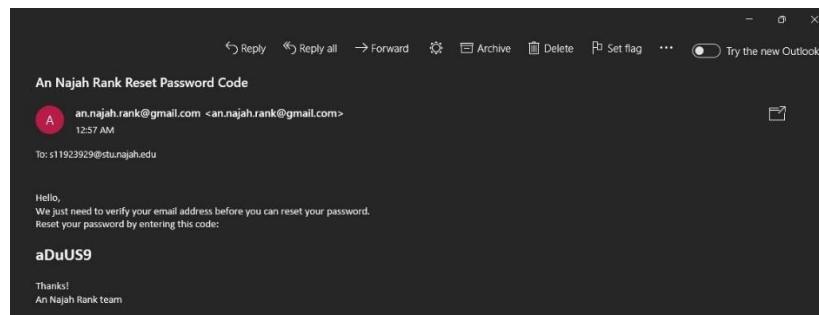


Figure 40: verification code message for reset password

The screenshot shows a 'New Password' page. At the top, there is a header with the logo 'NR An-Najah Rank'. Below the header, the title 'New Password' is centered. A message below the title reads: 'Set the new password for your account so you can login and access all features.' There are two input fields: 'New Password' and 'Confirm Password', both with masked entries. Below the input fields is a blue 'UPDATE PASSWORD' button.

Figure 41: set new password

3.4.1.4 Account Settings:

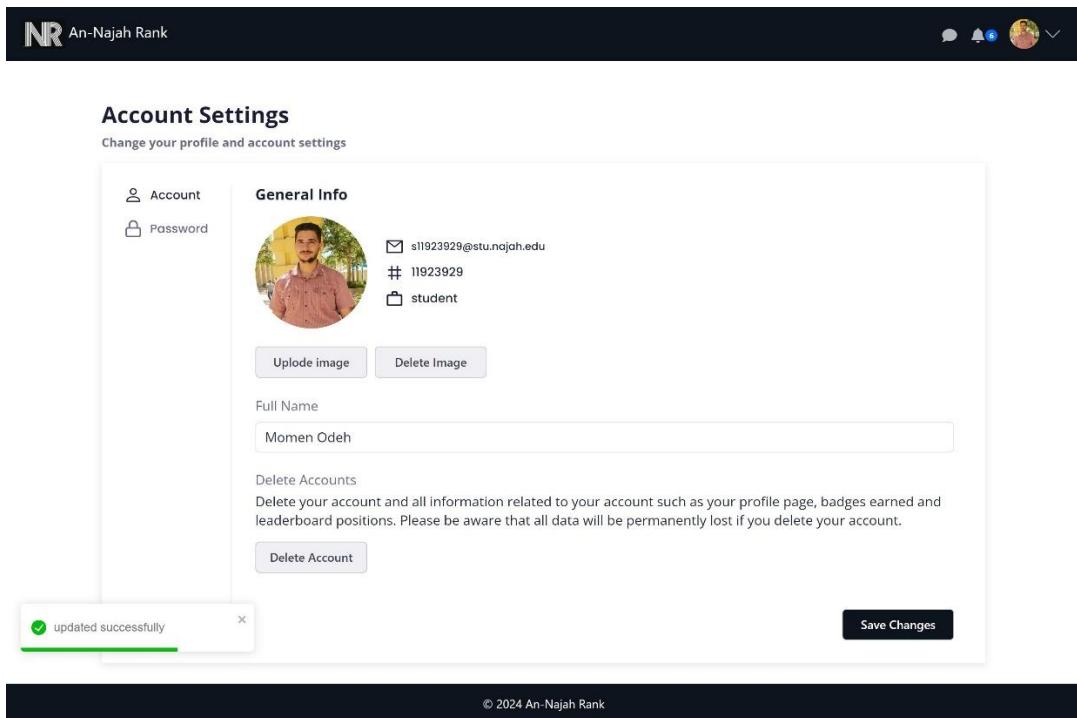


Figure 42: User account settings

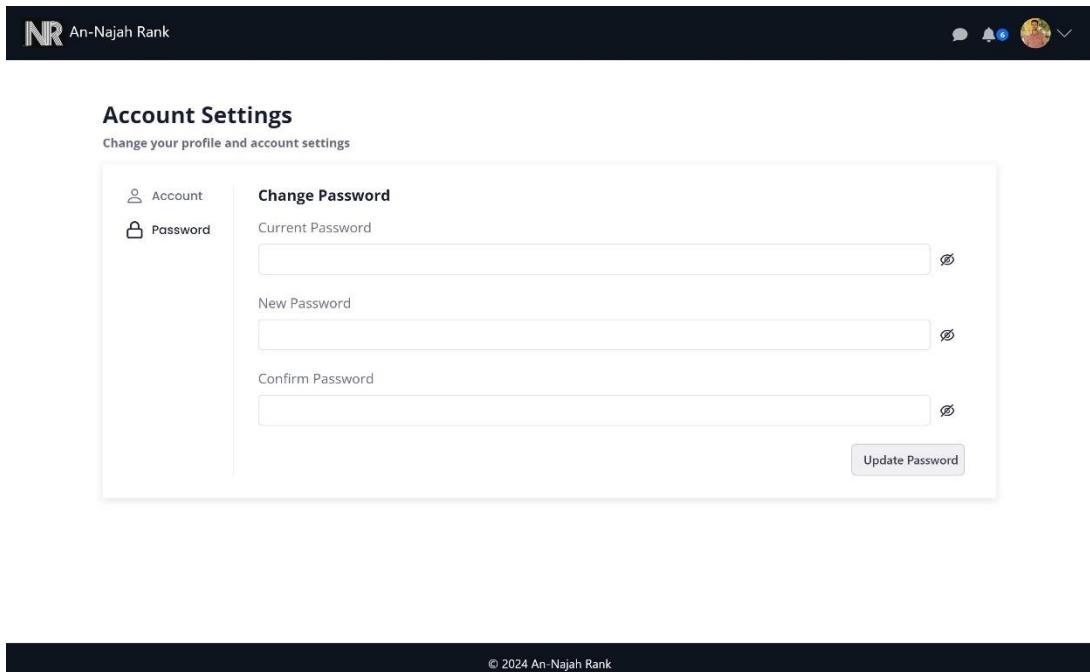


Figure 43: Password settings

3.4.1.5 Chatting:

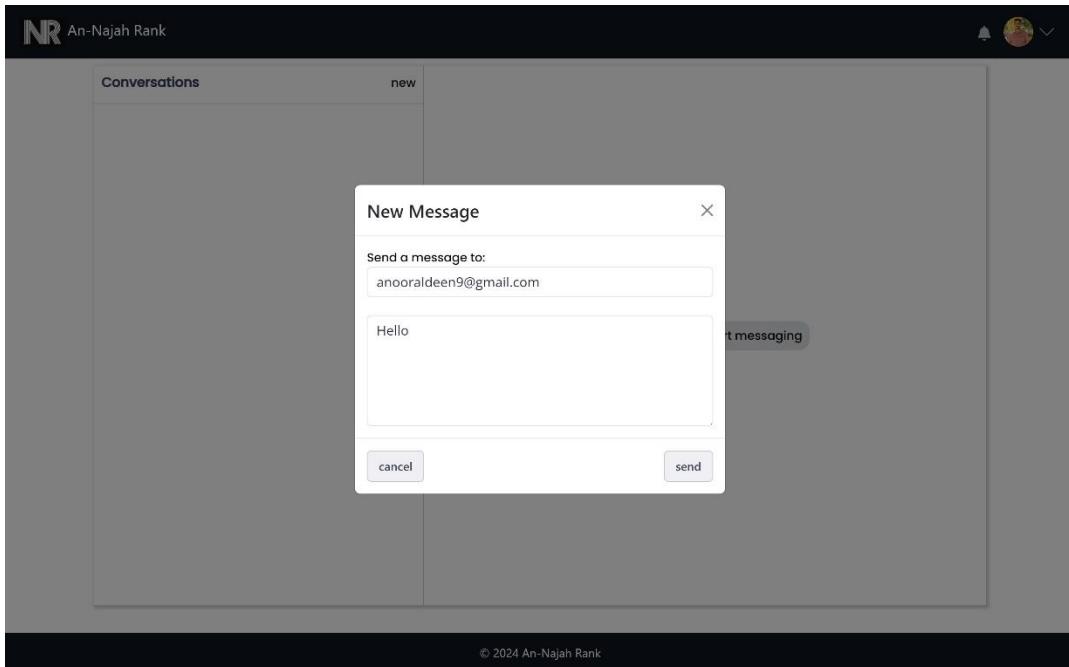


Figure 44: Create new Chat

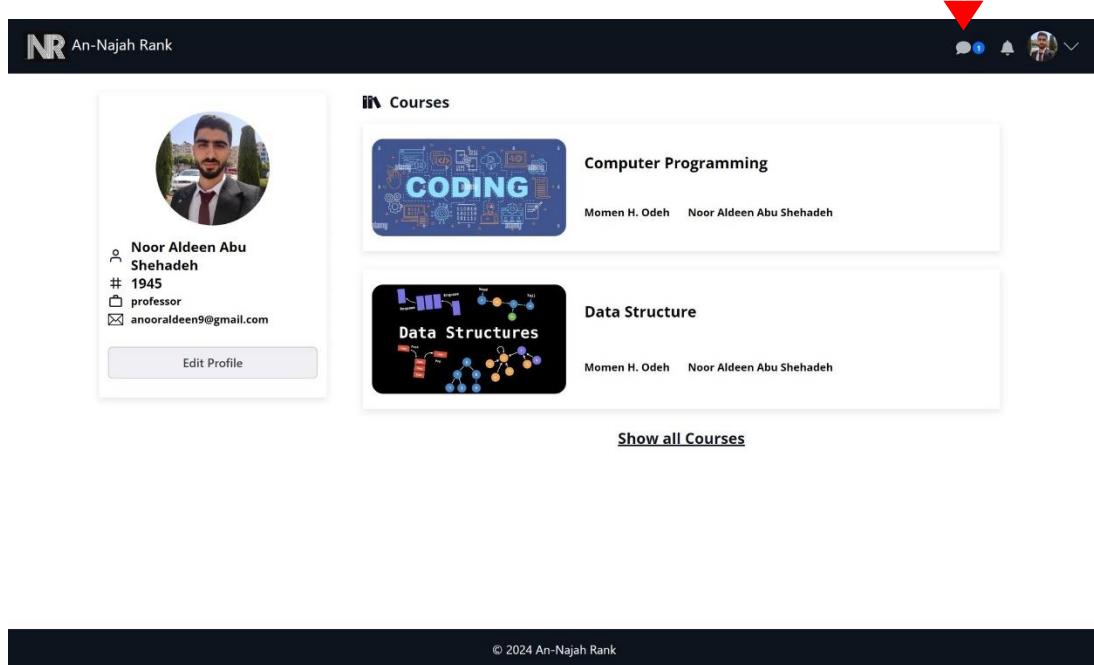


Figure 45: Chatting notification

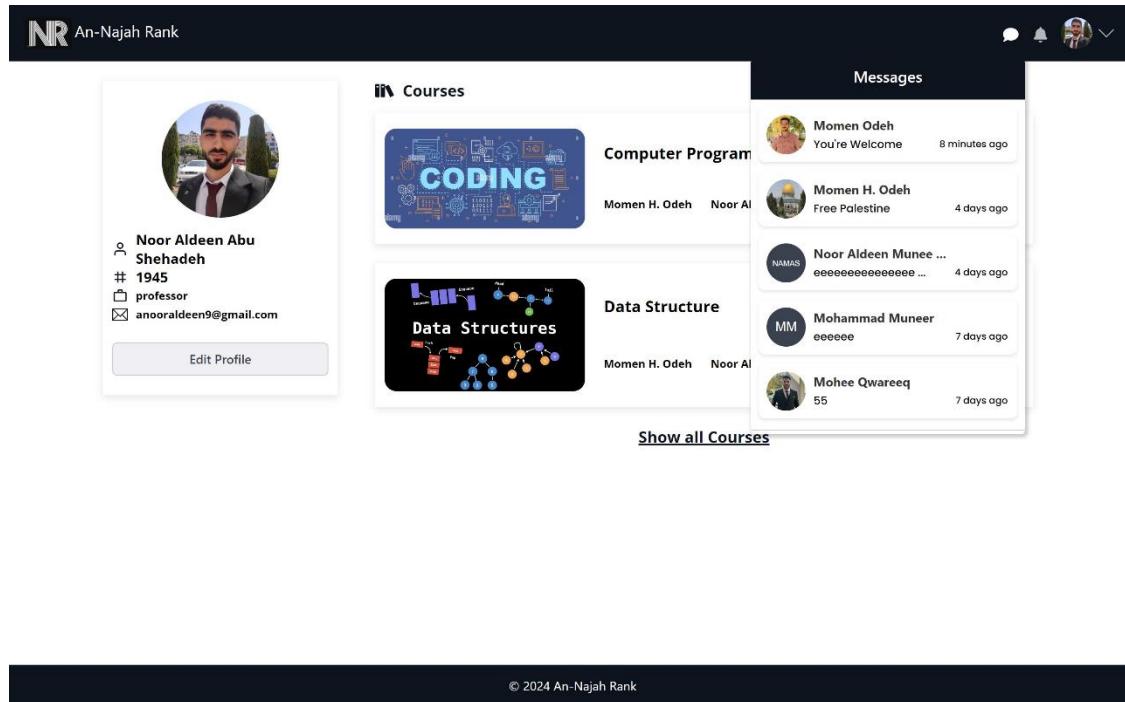


Figure 46: See chatting notification pop-up

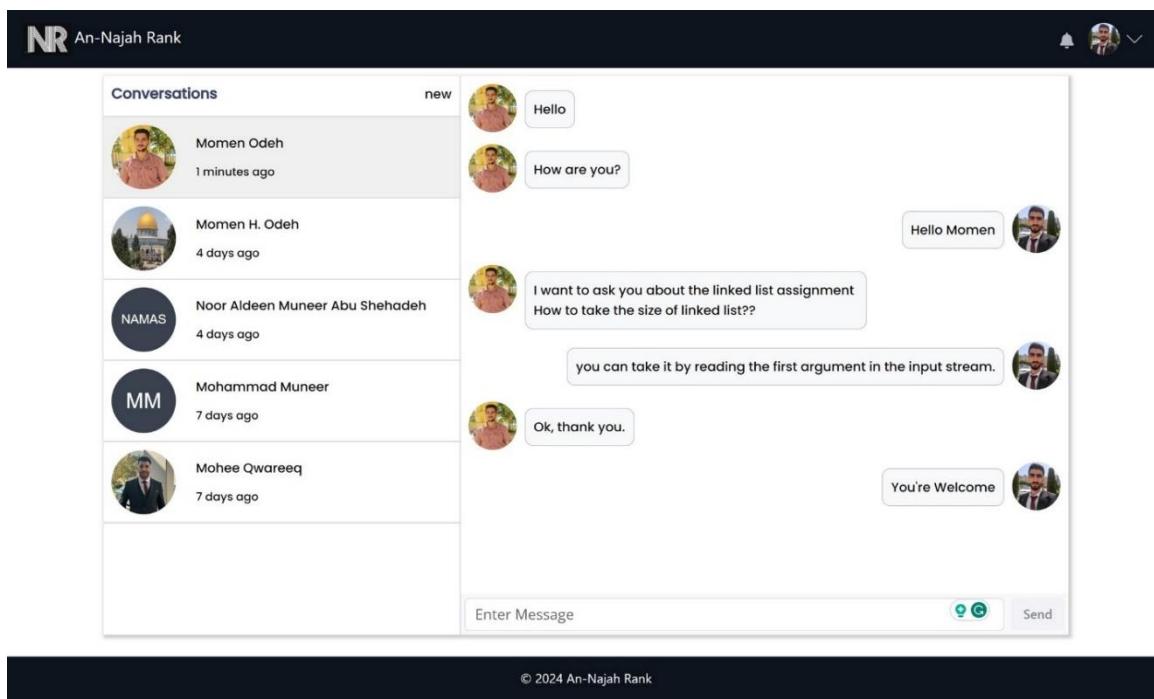


Figure 47: Chatting conversations

3.4.2 Admin features:

The screenshot shows the 'professors-requests' section of the admin dashboard. At the top, there's a navigation bar with the NR logo, 'An-Najah Rank', and user icons. Below it, a breadcrumb trail shows 'admin > professors-requests'. The main area is titled 'Welcome Back' and has tabs for 'Professors Requests', 'Professors', 'Students', and 'Submissions'. A search bar for 'Type Professor Name' is on the right. A table lists two pending professor requests:

| Professor Name | University Number | Email | ✓ | ✗ |
|--------------------------|-------------------|------------------------|---|---|
| Noor Aldeen Abu Shehadeh | 1945 | anooraldeen9@gmail.com | ✓ | ✗ |
| Momen H. Odeh | 11923 | momen.odeh74@gmail.com | ✓ | ✗ |

Figure 48: pending professors in admin page

This screenshot shows a modal dialog box over the admin interface. The dialog is titled 'Add professor' and contains the message 'are you sure that want to accept the professor with id 1945?'. At the bottom right of the dialog is a 'Yes' button. The background shows the same 'professors-requests' table from Figure 48, with the second row (Noor Aldeen Abu Shehadeh) being highlighted.

Figure 49: approve professor

The screenshot shows the 'Professors' section of the An-Najah Rank system. At the top, there is a navigation bar with the logo 'NR An-Najah Rank' and icons for messaging, notifications, and user profile. Below the navigation bar, the URL 'admin > professors' is displayed. A 'Welcome Back' message is shown above a navigation menu with tabs: 'Professors Requests', 'Professors' (which is selected), 'Students', and 'Submissions'. To the right of the menu is a search bar labeled 'Type Professor Name'. A table lists four professors with columns for 'Professor Name', 'University Number', and 'Email'. The professors listed are Noor Aldeen Abu Shehadeh, Saleh Rami, and Momen H. Odeh.

| Professor Name | University Number | Email |
|--------------------------|-------------------|------------------------|
| Noor Aldeen Abu Shehadeh | 1945 | anooraldeen9@gmail.com |
| Saleh Rami | 8597 | saleh@gmail.com |
| Momen H. Odeh | 11923 | momen.odeh74@gmail.com |

© 2024 An-Najah Rank

Figure 50: all professors in the system

The screenshot shows the 'Students' section of the An-Najah Rank system. At the top, there is a navigation bar with the logo 'NR An-Najah Rank' and icons for messaging, notifications, and user profile. Below the navigation bar, the URL 'admin > students' is displayed. A 'Welcome Back' message is shown above a navigation menu with tabs: 'Professors Requests', 'Professors', 'Students' (which is selected), and 'Submissions'. To the right of the menu is a search bar labeled 'Type Student Name'. A table lists five students with columns for 'Student Name', 'University Number', and 'Email'. The students listed are Mohammad Muneer, Mohee Qwareeq, Noor Aldeen Muneer Abu Shehadeh, and Momen Odeh.

| Student Name | University Number | Email |
|---------------------------------|-------------------|--------------------------|
| Mohammad Muneer | 11235499 | jjjjjjjj220379@gmail.com |
| Mohee Qwareeq | 11821353 | moheedaeb16@gmail.com |
| Noor Aldeen Muneer Abu Shehadeh | 11923513 | s11923513@stu.najah.edu |
| Momen Odeh | 11923929 | s11923929@stu.najah.edu |

© 2024 An-Najah Rank

Figure 51: all students in the system

The screenshot shows the 'Submissions' section of the admin dashboard. At the top, there are tabs for 'Professors Requests', 'Professors', 'Students', and 'Statistics'. The 'Statistics' tab is active. Below the tabs, a search bar contains the placeholder 'Type Student Name'. A table lists student submissions:

| Student Name | University Number | Total Submission | Total Success Submission | Rate |
|---------------------------------|-------------------|------------------|--------------------------|---------|
| Noor Aldeen Muneer Abu Shehadeh | 11923513 | 4 | 4 | 100.00% |
| Momen Odeh | 11923929 | 4 | 3 | 75.00% |
| Mohammad Muneer | 11235499 | 1 | 1 | 100.00% |
| Mohammad Zaed | 11924574 | 4 | 0 | 0% |

In the bottom right corner of the dashboard, there is a footer bar with the text '© 2024 An-Najah Rank'.

Figure 52: Students statistics

The screenshot shows a student profile for 'Noor Aldeen Muneer Abu Shehadeh' (ID: 11923513). The profile includes a circular icon labeled 'NAMAS', basic information like name, ID, and email, and a 'Student Statistics' section. The 'Solved' status is highlighted with a red circle.

Student Statistics

| Difficulty | Count | Percentage |
|------------|-------|------------|
| Easy | 3/3 | 100.00% |
| Medium | 1/1 | 100.00% |
| Hard | 0/0 | 0% |

Courses

- Computer Programming**: Momen H. Odeh, Noor Aldeen Abu Shehadeh
- Data Structure**: Momen H. Odeh, Noor Aldeen Abu Shehadeh

In the bottom right corner of the dashboard, there is a footer bar with the text '© 2024 An-Najah Rank'.

Figure 53: Viewing student profiles from the admin side.

3.4.3 Professor Features:

The screenshot shows the 'Administration' section of the An-Najah Rank website. At the top, there are navigation links for 'Manage Courses' and 'Manage Challenges'. Below this is a search bar labeled 'Type course name' and a 'Create Course' button. A table lists three courses: 'problem solving' owned by Noor Aldeen Abu Shehadeh, 'Data Structure' owned by Noor Aldeen Abu Shehadeh, and 'Computer Programming' owned by Momen H. Odeh. Each row in the table includes a delete icon. The footer contains the copyright notice '© 2024 An-Najah Rank'.

Figure 54: Manage Courses in administration page

The screenshot shows the 'Create Course' form. It includes fields for 'Course Number' (empty), 'Course Name' (empty), 'Description' (empty), 'Background Image' (choose file, no file chosen), and 'Students Excel File' (choose file, no file chosen). A note states '* should enter Students Excel File with .xlsx extension'. At the bottom are 'Cancel Changes' and 'Save Changes' buttons. The footer contains the copyright notice '© 2024 An-Najah Rank'.

Figure 55: Create course

NR An-Najah Rank

administration > courses > 106362II > details

Data Structure

Details Moderators Course Students Manage Contests

Course Number 106362II

Course Name Data Structure

Description A data structure is a way of organizing and storing data to perform operations efficiently. It defines the relationship between data elements, the operations that can be performed on the data, and the rules for organizing

Background Image Choose File | No file chosen

Cancel Changes **Save Changes**

© 2024 An-Najah Rank

Figure 56: Manage course details

NR An-Najah Rank

administration > courses > 106362II > moderators

Data Structure

Details **Moderators** Course Students Manage Contests

Moderators

| | |
|------------------------------|-----|
| S | Add |
| Saleh Rami saleh@gmail.com | |

Noor Aldeen Abu Shehadeh
anooraldeen9@gmail.com
owner

Momen H. Odeh
momen.odeh74@gmail.com
moderator

© 2024 An-Najah Rank

Figure 57: Manage course moderators

NR An-Najah Rank

administration > courses > 10636211 > members

Data Structure

| Details | Moderators | Course Students | Manage Contests |
|---|---------------------------------|--------------------------|-----------------|
| <input type="text" value="Type username to search"/> <input type="button" value="Add Student"/> | | | |
| Registration Number | Name | email | |
| 11235499 | Mohammad Muneer | jijijijj220379@gmail.com | |
| 11923513 | Noor Aldeen Muneer Abu Shehadeh | s11923513@stu.najah.edu | |
| 11923929 | Momen Odeh | s11923929@stu.najah.edu | |
| 11924574 | Mohammad Zaed | s11924574@stu.najah.edu | |
| 11612344 | not registered in system yet | | |
| 11715286 | not registered in system yet | | |
| 11819424 | not registered in system yet | | |
| 11821711 | not registered in system yet | | |
| 11822163 | not registered in system yet | | |
| 11822687 | not registered in system yet | | |
| 11822841 | not registered in system yet | | |

Figure 58: Manage students in course

NR An-Najah Rank

administration > courses > 10636211 > contests

Data Structure

| Details | Moderators | Course Students | Manage Contests | |
|--|--------------------------|---------------------|---------------------|--|
| <input type="text" value="Type contest name"/> <input type="button" value="Create Contest"/> | | | | |
| Contest Name | Contest Owner | Start Date | End Date | |
| Linked List | Noor Aldeen Abu Shehadeh | 2024-01-10 17:36:00 | 2024-01-20 23:59:00 | |
| Tree | Noor Aldeen Abu Shehadeh | 2024-01-20 22:25:00 | 2024-01-30 22:25:00 | |
| Assignment 99 | Momen H. Odeh | 2024-01-30 17:33:00 | 2024-01-30 17:34:00 | |

© 2024 An-Najah Rank

Figure 59: Manage contests

NR An-Najah Rank

administration > courses > 116087564 > contests > create-contest

Contest Details

| | |
|---|--|
| Contest Name | <input type="text"/> |
| Start Time | <input type="text"/> dd/mm/yyyy --::-- -- <input type="button" value=""/> |
| End Time | <input type="text"/> dd/mm/yyyy --::-- -- <input type="button" value=""/> |
| <input checked="" type="checkbox"/> This contest has end time. | |
| Description | <input type="text"/> <div style="border: 1px solid #ccc; padding: 2px;"> Normal ‡ B I U S ” ≡ ≡ ≡ ↖ ↗ Tx </div> |
| <input type="button" value="Cancel Changes"/> <input type="button" value="Save Changes"/> | |

© 2024 An-Najah Rank

Figure 60: Create contest

NR An-Najah Rank

administration > courses > 10636211 > contests > 81 > details

Linked List

| | | | | | | | | | | | | | |
|--|--|---------------------|----------------------------------|-------------------|--|-----------------|--|--|--|--------------------|--|---|--|
| <input type="button" value="Details"/> | <input type="button" value="Challenges"/> | | | | | | | | | | | | |
| <h4>Contest Details</h4> <table border="1"> <tr> <td>Contest Name</td> <td><input type="text"/> Linked List</td> </tr> <tr> <td>Start Time</td> <td><input type="text"/> 10/01/2024 01:20 AM <input type="button" value=""/></td> </tr> <tr> <td>End Time</td> <td><input type="text"/> 20/01/2024 11:59 PM <input type="button" value=""/></td> </tr> <tr> <td colspan="2"><input checked="" type="checkbox"/> This contest has end time.</td> </tr> <tr> <td>Description</td> <td> <input type="text"/> <div style="border: 1px solid #ccc; padding: 2px;"> A linked list is a linear data structure where elements, called nodes, are connected through pointers, forming a sequence. Each node contains data and a reference to the next node in the sequence. </div> </td> </tr> <tr> <td colspan="2" style="text-align: center;"> <input type="button" value="Cancel Changes"/> <input type="button" value="Save Changes"/> </td> </tr> </table> | | Contest Name | <input type="text"/> Linked List | Start Time | <input type="text"/> 10/01/2024 01:20 AM <input type="button" value=""/> | End Time | <input type="text"/> 20/01/2024 11:59 PM <input type="button" value=""/> | <input checked="" type="checkbox"/> This contest has end time. | | Description | <input type="text"/> <div style="border: 1px solid #ccc; padding: 2px;"> A linked list is a linear data structure where elements, called nodes, are connected through pointers, forming a sequence. Each node contains data and a reference to the next node in the sequence. </div> | <input type="button" value="Cancel Changes"/> <input type="button" value="Save Changes"/> | |
| Contest Name | <input type="text"/> Linked List | | | | | | | | | | | | |
| Start Time | <input type="text"/> 10/01/2024 01:20 AM <input type="button" value=""/> | | | | | | | | | | | | |
| End Time | <input type="text"/> 20/01/2024 11:59 PM <input type="button" value=""/> | | | | | | | | | | | | |
| <input checked="" type="checkbox"/> This contest has end time. | | | | | | | | | | | | | |
| Description | <input type="text"/> <div style="border: 1px solid #ccc; padding: 2px;"> A linked list is a linear data structure where elements, called nodes, are connected through pointers, forming a sequence. Each node contains data and a reference to the next node in the sequence. </div> | | | | | | | | | | | | |
| <input type="button" value="Cancel Changes"/> <input type="button" value="Save Changes"/> | | | | | | | | | | | | | |

© 2024 An-Najah Rank

Figure 61: Manage contest details

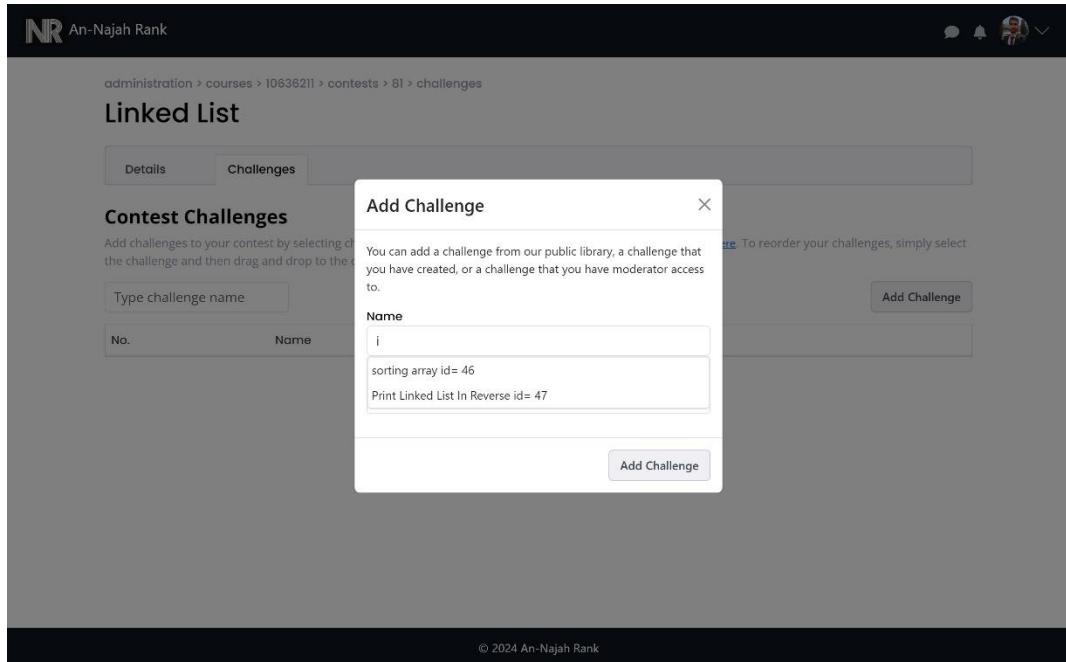


Figure 62: Add challenge to contest

The screenshot shows a page titled "Assignment 1" with a navigation path: administration > courses > 106361II > contests > 80 > challenges. The title "Contest Challenges" is visible. Below it is a note: "Add challenges to your contest by selecting challenges from our library or create and add your own challenges [here](#). To reorder your challenges, simply select the challenge and then drag and drop to the desired location." A "Type challenge name" input field and an "Add Challenge" button are present. A table lists three challenges:

| No. | Name | Max Score | Actions |
|-----|-------------------------|-----------|---------|
| 0 | Add Two Numbers id= 48 | 15 | |
| 1 | factorial number id= 49 | 25 | |
| 2 | prime number id= 50 | 10 | |

Figure 63: Manage challenges in contest

administration > challenges

Administration

[Manage Courses](#) [Manage Challenges](#)

Type challenge name Create Challenge

| Challenge Name | Challenge tags | Challenge Owner |
|------------------------------|----------------|--------------------------|
| Print Linked List In Reverse | data structure | Noor Aldeen Abu Shehadeh |
| Add Two Numbers | | Noor Aldeen Abu Shehadeh |
| factorial number | | Noor Aldeen Abu Shehadeh |
| prime number | | Noor Aldeen Abu Shehadeh |

© 2024 An-Najah Rank

Figure 64: Manage challenges

administration > challenges > create-challenge

Details

Challenge Difficulty Easy

Specify Language

- Java
- C
- C++
- Python
- JavaScript
- Regex

Java Base File Use default file Upload base file

C Base File Use default file Upload base file

Challenge Name

Description

Problem Statement

Input Format

Constraints

Output Format

Challenge Privacy make the challenge public.

Tags Add

[Cancel Changes](#) [Save Changes](#)

© 2024 An-Najah Rank

Figure 65: Create challenge

An-Najah Rank

administration > challenges > 47

Print Linked List In Reverse

[Details](#) [TestCases](#)

Challenge Difficulty: Easy

Specify Language: Java C C++ Python JavaScript Regex

Java Base File: Use default file Upload base file
Choose file: No file chosen

C Base File: Use default file Upload base file
Choose file: No file chosen

C++ Base File: Use default file Upload base file

Challenge Name: Print Linked List In Reverse

Description: -

Problem Statement: Normal t B I U G # E = = = = = = = = = =
get data from input screen and build a linked list then print the linked list in reverse

Input Format: Normal t B I U G # E = = = = = = = = = =
The first line contains an integer , the number of elements in the linked list.
The next lines contain integers for the linked list data separated by spaces.

Constraints: Normal t B I U S # E = = = = = = = = = =
-

Output Format: Normal t B I U G # E = = = = = = = = = =
an integer of reverse linked list data separated by spaces

Challenge Privacy: make the challenge public.

Tags: Add a tag [Add](#)
data structures

[Cancel Changes](#) [Save Changes](#)

© 2024 An-Najah Rank

Figure 66: Manage challenge details

An-Najah Rank

administration > challenges > 47

Print Linked List In Reverse

[Details](#) [TestCases](#)

* Should add at least one test case.

Order:

Add Test Case

Strength: 0 Sample

Input:
1 3
2 1 2 3

Output:
1 3 2 1

Explanation:
this is a sample of reverse print linked list.

[Save](#)

© 2024 An-Najah Rank

Figure 67: Add test case to challenge

NR An-Najah Rank

administration > challenges > 47 > test-cases

Print Linked List In Reverse

Details TestCases

Add Test Case

* Should add at least one sample test case to enable use this challenge.

| Order | Input | Output | Is Sample | Strength | Edit | Delete |
|-------|-----------------|---------------|-----------|----------|------|--------|
| 0 | 3 1 2 3 | 3 2 1 | ✓ | 0 | | |
| 1 | 5 3 7 2 12 10 | 10 12 2 7 3 | ✗ | 10 | | |
| 2 | 6 45 8 9 7 12 0 | 0 12 7 9 8 45 | ✗ | 10 | | |
| 3 | 1 5 | 5 | ✗ | 10 | | |

© 2024 An-Najah Rank

Figure 68: Manage test cases in challenge

NR An-Najah Rank

My Courses

Computer Programming

Momen H. Odeh Noor Aldeen Abu Shehadeh

Data Structure

Momen H. Odeh Noor Aldeen Abu Shehadeh

© 2024 An-Najah Rank

Figure 69: All courses page

NR An-Najah Rank

The screenshot shows the course page for 'Data Structures'. At the top, there's a navigation bar with icons for messages, notifications, and user profile. Below the header is a banner featuring diagrams of a queue (enqueue/dequeue), a linked list (Head/Tail), a stack (push/pop), and two binary search trees. The main title 'Data Structures' is centered above these diagrams.

Description:

A data structure is a way of organizing and storing data to perform operations efficiently. It defines the relationship between data elements, the operations that can be performed on the data, and the rules for organizing the data. Different types of data structures serve various purposes, and their selection depends on the specific requirements of a task or problem.

Below the description are two tabs: 'Contests' (selected) and 'Course Students'. Under 'Contests', there are two entries:

- Linked List**: Solved Rate: 5.66%, max Score: 25. It includes a timer showing 10 days, 0 hours, 58 minutes, and 22 seconds, and a 'View Contest' button.
- Tree**: Solved Rate: 0%. It includes a timer showing 9 days, 23 hours, 22 minutes, and 22 seconds, and a 'View Contest' button.

At the bottom of the page is a footer with the text '© 2024 An-Najah Rank'.

Figure 70: Course view

NR An-Najah Rank

This screenshot shows the 'Manage students in course' interface for the 'Data Structures' course. At the top, there's a navigation bar with icons for messages, notifications, and user profile. Below the header is a banner featuring the same four data structure diagrams as Figure 70.

Description:

A data structure is a way of organizing and storing data to perform operations efficiently. It defines the relationship between data elements, the operations that can be performed on the data, and the rules for organizing the data. Different types of data structures serve various purposes, and their selection depends on the specific requirements of a task or problem.

Below the description are two tabs: 'Contests' (selected) and 'Course Students'. A search bar allows users to 'Type username to search'. A table lists student information:

| Registration Number | Name | email |
|---------------------|-----------------|-------------------------|
| 11235499 | Mohammad Muneer | jjjjjjj220379@gmail.com |

Each row in the table has a delete icon (trash bin) at the end.

Figure 71: Manage students in course

The screenshot shows the contest view for challenge 81. At the top, there's a navigation bar with the NR logo, course information (courses > 106362II > contests > 81), and user icons for notifications, messages, profile, and a dropdown menu. Below the navigation is a section titled "Linked List" with a "Description" box containing a definition of a linked list. A "Remaining time" box shows "Time's up!". Below these are tabs for "Challenges" and "Grades". A challenge card for "Print Linked List In Reverse" is displayed, showing it's an "Easy" challenge with a success rate of 5.66% and a max score of 25. A "View Challenge" button is at the bottom right of the card. The footer contains the copyright notice "© 2024 An-Najah Rank".

Figure 72: Contest view

The screenshot shows the contest grades view for challenge 81. It has the same header and navigation as Figure 72. Below the navigation is a "Linked List" section with a "Description" box. A "Remaining time" box shows "Time's up!". Below these are "Challenges" and "Grades" tabs. A search bar for "Type student name" and a "Export grades to excel file" button are present. A table displays student grades:

| University Number | Name | Print Linked List In Reverse/25 | Total Result/25 |
|-------------------|---------------------------------|---------------------------------|-----------------|
| 11235499 | Mohammad Muneer | 25 | 25 |
| 11923513 | Noor Aldeen Muneer Abu Shehadeh | 25 | 25 |
| 11923929 | Momen Odeh | 25 | 25 |

The footer contains the copyright notice "© 2024 An-Najah Rank".

Figure 73: Contest Grades

The screenshot shows a web-based programming challenge interface. At the top, there's a navigation bar with the 'An-Najah Rank' logo and user icons. Below it, the URL indicates the path: courses > 10636211 > contests > 81 > challenges > 47 > problem. The main title is 'Print Linked List In Reverse'. A navigation bar below the title has tabs for 'Problem' (which is selected), 'Submissions', and 'Leaderboard'.
Input Format:
The first line contains an integer , the number of elements in the linked list.
The next lines contain an integers for the linked list data separated by space.
Constraints:
-
Output Format:
an integers of reverse linked list data separated by space.
Simple Input 0
3
1 2 3
Sample Output 0
3 2 1
Explanation 0
this is a sample of reverse print linked list.

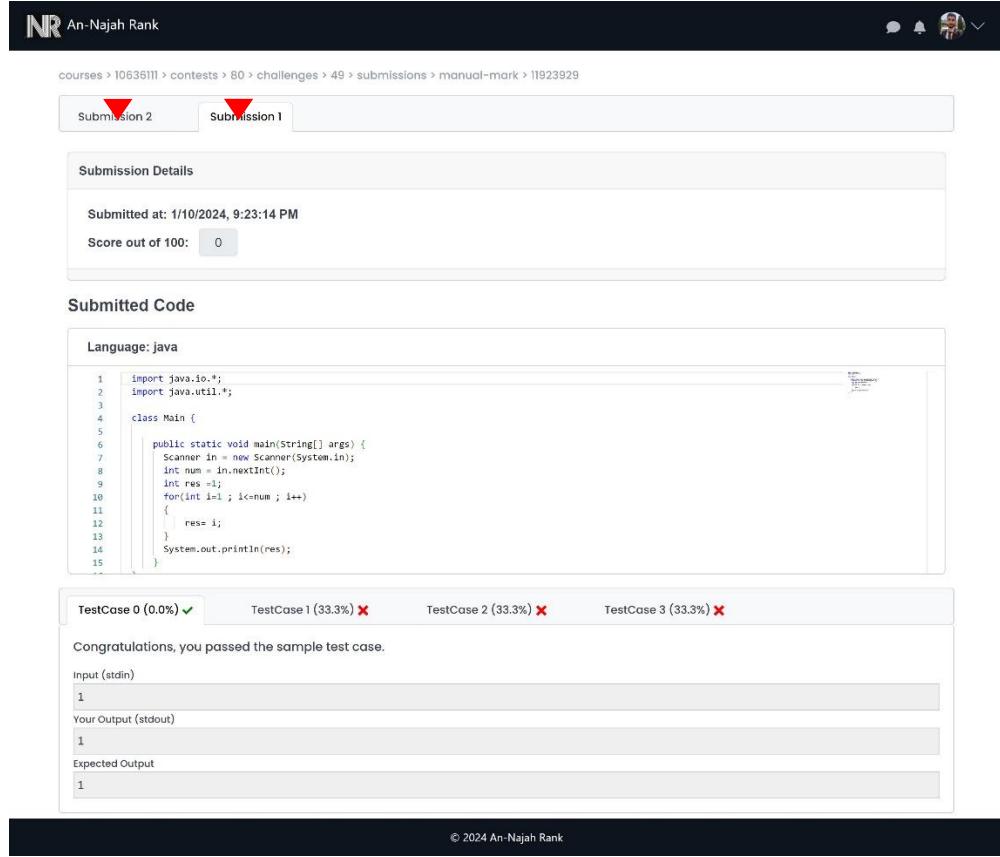
Figure 74: Problem description

The screenshot shows the 'Submissions' tab of the challenge interface. The URL at the top is courses > 10636211 > contests > 81 > challenges > 47 > submissions. The title 'Print Linked List In Reverse' is displayed again. A search bar 'Type student name' and a 'Calculate Similarity' button are present. Below is a table of student submissions:

| Name | Date | Score ▼▲ | Similarity ▼▲ | |
|---------------------------------|-------------------------------|----------|---------------|----------------------------------|
| Noor Aldeen Muneer Abu Shehadeh | Wed, 10 Jan 2024 16:15:26 GMT | 25 25 | --- | View Submissions |
| Mohammad Muneer | Wed, 10 Jan 2024 17:40:29 GMT | 25 25 | --- | View Submissions |
| Momen Odeh | Wed, 10 Jan 2024 17:40:48 GMT | 25 25 | --- | View Submissions |

At the bottom, a footer bar contains the text '© 2024 An-Najah Rank'.

Figure 75: Students submissions from professor side



The screenshot shows a web-based submission viewer for a Java challenge. At the top, there's a navigation bar with the An-Najah Rank logo and user icons. Below it, the URL indicates the submission is under manual marking. Two tabs are visible: 'Submission 2' (selected) and 'Submission 1'. The 'Submission Details' section shows the submission was submitted at 1/10/2024, 9:23:14 PM, and has a score of 0 out of 100. The 'Submitted Code' section displays the following Java code:

```
1 import java.io.*;
2 import java.util.*;
3
4 class Main {
5
6     public static void main(String[] args) {
7         Scanner in = new Scanner(System.in);
8         int num = in.nextInt();
9         int res =1;
10        for(int i=1 ; i<=num ; i++)
11        {
12            res*= i;
13        }
14        System.out.println(res);
15    }
}
```

Test results for four test cases are shown: TestCase 0 (0.0%) passed (green checkmark), TestCase 1 (33.3%) failed (red X), TestCase 2 (33.3%) failed (red X), and TestCase 3 (33.3%) failed (red X). Below the code editor, there are input, output, and expected output fields. The input field contains '1', the output field contains '1', and the expected output field also contains '1'. The footer of the page includes a copyright notice: © 2024 An-Najah Rank.

Figure 76: View student submissions

The screenshot shows a submission page for a challenge. At the top, there are navigation links: courses > I0636111 > contests > 80 > challenges > 49 > submissions > manual-mark > I1923929. Below this, there are two tabs: "Submission 2" and "Submission 1". A red arrow points to the "Score out of 100:" field, which contains "100". Another red arrow points to the "Save Changes" button. The "Submitted Code" section shows Java code for calculating the factorial of a number. Below the code, four test cases are listed: TestCase 0 (0.0%) ✅, TestCase 1 (33.3%) ✅, TestCase 2 (33.3%) ✅, and TestCase 3 (33.3%) ✅. A message says "Congratulations, you passed the sample test case." Input and output fields are shown for each test case. At the bottom, there is a copyright notice: © 2024 An-Najah Rank.

Figure 77: Last submission for student can do manual mark

The screenshot shows a leaderboard page for a challenge titled "Print Linked List In Reverse". At the top, there are three tabs: "Problem", "Submissions" (which is selected), and "Leaderboard". A red arrow points to the "Calculate Similarity" button. Below the tabs, there is a search bar labeled "Type student name" and a "Calculate Similarity" button. The main area displays a table of submissions:

| Name | Date | Score ▼▲ | Similarity ▼▲ |
|---------------------------------|-------------------------------|----------|---------------|
| Noor Aldeen Muneer Abu Shehadeh | Wed, 10 Jan 2024 16:15:26 GMT | 25 25 | ... |
| Mohammad Muneer | Wed, 10 Jan 2024 17:40:29 GMT | 25 25 | ... |
| Momen Odeh | Wed, 10 Jan 2024 17:40:48 GMT | 25 25 | ... |

Each row has a "View Submissions" button. At the bottom, there is a copyright notice: © 2024 An-Najah Rank.

Figure 78: Start calculate similarity

An-Najah Rank

courses > 10636211 > contests > 81 > challenges > 47 > submissions

Print Linked List In Reverse

Problem Submissions Leaderboard

Type student name Calculate Similarity

| Name | Date | Score $\downarrow \uparrow$ | Similarity $\downarrow \uparrow$ |
|---------------------------------|-------------------------------|-----------------------------|----------------------------------|
| Noor Aldeen Muneer Abu Shehadeh | Wed, 10 Jan 2024 16:15:26 GMT | $\frac{25}{25}$ | 71% |
| Mohammad Muneer | Wed, 10 Jan 2024 17:40:29 GMT | $\frac{25}{25}$ | 64% |
| Momen Odeh | Wed, 10 Jan 2024 17:40:48 GMT | $\frac{25}{25}$ | 30% |

Similarity data ready for submissions in Data Structure course

© 2024 An-Najah Rank

Figure 79: Received notification when similarity calculated

An-Najah Rank

courses > 10636211 > contests > 81 > challenges > 47 > submissions > code-similarity > l1923513

Code Similarity Summary

Noor_Aldeen_Muneer_Abu_Shehadeh-11923513 (71%)

```
29 return;
30 }
31 printReverse(head->next);
32 printf("%d ", head->data);
33 }
34 int main() {
35 /* Enter your code here. Read input from STDIN. Print out
36 int size;
37 scanf("%d", &size);
38 struct Node* head = NULL;
39 for (int i = 0; i < size; i++) {
40 int item;
41 scanf("%d", &item);
42 head = insertNode(head, item);
43 }
44 printReverse(head);
45 return 0;
46 }
```

Mohammad_Muneer-11235499 (58%)

Mohammad_Muneer-11235499 (58%)

Momen_Odeh-11923929 (43%)

© 2024 An-Najah Rank

Figure 80: Code similarity page

NR An-Najah Rank

courses > 10636211 > contests > 81 > challenges > 47 > submissions > code-similarity > 11923513

Code Similarity Summary

| Noor_Aldeen_Muneer_Abu_Shehadeh-11923513 (71%) | Mohammad_Muneer-11235499 (58%) |
|--|--|
| <pre> 10 struct Node* insertNode(struct Node* head, int data) { 11 struct Node* newNode = (struct Node*)malloc(sizeof(struct 12 newNode->data = data; 13 newNode->next = NULL; 14 15 if (head == NULL) { 16 return newNode; 17 } 18 19 struct Node* current = head; 20 while (current->next != NULL) { 21 current = current->next; 22 } 23 24 current->next = newNode; 25 return head; 26 } 27 void printReverse(struct Node* head) {</pre> | <pre> 9 struct Node* insertNode(struct Node* head, int data) { 10 struct Node* newNode = (struct Node*)malloc(sizeof(struct 11 newNode->data = data; 12 newNode->next = head; 13 return newNode; 14 } 15 16 void printReverse(struct Node* head) { 17 if (head == NULL) { 18 return; 19 } 20 printf("%d ", head->data); 21 printReverse(head->next); 22 } 23 24 int main() { 25 int size; 26 scanf("%d", &size);</pre> |

© 2024 An-Najah Rank

Figure 81: Code Similarity view 1

NR An-Najah Rank

courses > 10636211 > contests > 81 > challenges > 47 > submissions > code-similarity > 11923513

Code Similarity Summary

| Noor_Aldeen_Muneer_Abu_Shehadeh-11923513 (71%) | Momen_Odeh-11923929 (43%) |
|---|--|
| <pre> 9 }; 10 struct Node* insertNode(struct Node* head, int data) { 11 struct Node* newNode = (struct Node*)malloc(sizeof(struct 12 newNode->data = data; 13 newNode->next = NULL; 14 15 if (head == NULL) { 16 return newNode; 17 } 18 19 struct Node* current = head; 20 while (current->next != NULL) { 21 current = current->next; 22 } 23 24 current->next = newNode; 25 return head; 26 }</pre> | <pre> 11 Node* insertNode(struct Node* head, int data) { 12 struct Node* newNode = (struct Node*)malloc(sizeof(struct 13 newNode->val = data; 14 newNode->next = NULL; 15 16 if (head == NULL) { 17 return newNode; 18 } 19 20 struct Node* current = head; 21 while (current->next != NULL) { 22 current = current->next; 23 } 24 25 current->next = newNode; 26 return head; 27 }</pre> |

© 2024 An-Najah Rank

Figure 82: Code similarity view 2

An-Najah Rank

courses > 10636211 > contests > 81 > challenges > 47 > leaderboard

Print Linked List In Reverse

Problem Submissions Leaderboard

Type student name

| Rank | Student Name | Score | Time |
|------|---------------------------------|-------|-------------------------------|
| 1 | Noor Aldeen Muneer Abu Shehadeh | 25 | Wed, 10 Jan 2024 16:15:26 GMT |
| 2 | Mohammad Muneer | 25 | Wed, 10 Jan 2024 17:40:29 GMT |
| 3 | Momen Odeh | 25 | Wed, 10 Jan 2024 17:40:48 GMT |

© 2024 An-Najah Rank

Figure 83: students leaderboards

administration > challenges > 47 > test-cases

Print Linked List In Reverse

Add Test Case

contest 81 - Linked List in course 10636211 - Data Structure.

* Should add at least one test case.

Strength: 10 Sample

input:
1 1
2 2

output:
1 2

Save

© 2024 An-Najah Rank

Figure 84: Add new test case when there is a submission for challenge

NR An-Najah Rank

administration > challenges > 47 > test-cases

Print Linked List In Reverse

Details TestCases

Add Test Case

* Should add at least one sample test case to enable use this challenge.

| Order | Input | Output | Is Sample | Strength | Edit | Delete |
|-------|-----------------|---------------|-----------|----------|------|--------|
| 0 | 3 1 2 3 | 3 2 1 | ✓ | 0 | | |
| 1 | 5 3 7 2 12 10 | 10 12 2 7 3 | ✗ | 10 | | |
| 2 | 6 45 8 9 7 12 0 | 0 12 7 9 8 45 | ✗ | 10 | | |
| 3 | 1 5 | 5 | ✗ | 10 | | |
| 4 | 1 2 | 2 | ✗ | 10 | | |

© 2024 An-Najah Rank

Figure 85: After add test case and run it on all student submission

NR An-Najah Rank

courses > 10636211 > contests > 81 > challenges > 47 > submissions > manual-mark > 11923513

Submission 0

Submission Details

Submitted at: 1/10/2024, 6:15:26 PM

Score out of 100: 100

Save Changes

Submitted Code

Language: c

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <math.h>
4 #include <stdlib.h>
5
6 struct Node {
7     int data;
8     struct Node* next;
9 }
10 struct Node* insertNode(struct Node* head, int data) {
11     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
12     newNode->data = data;
13     newNode->next = NULL;
14
15     if (head == NULL) {
    
```

TestCase 0 (0.0%) ✓ TestCase 1 (25.0%) ✓ TestCase 2 (25.0%) ✓ TestCase 3 (25.0%) ✓ TestCase 4 (25.0%) ✓

Congratulations, you passed the sample test case.

Input (stdin)

```

1
2

```

Your Output (stdout)

```

2

```

Expected Output

```

2

```

© 2024 An-Najah Rank

Figure 86: The submission after add new test case

3.4.4 Student Features:

The screenshot displays a student's profile on the An-Najah Rank platform. At the top left is the NR logo and "An-Najah Rank". Top right icons include a message bubble, a bell, a user profile, and a dropdown menu. On the left, a sidebar shows a circular profile picture with "NAMAS" and the student's details: Noor Aldeen Muneer Abu Shehadeh, # 11923513, student, s11923513@stu.najah.edu, with an "Edit Profile" button. The main area starts with "Student Statistics" showing 100.00% solved for all levels (Easy, Medium, Hard). Below is a section for "Courses" featuring "Computer Programming" by Momen H. Odeh and Noor Aldeen Abu Shehadeh, and "Data Structure" by the same authors. A "Show all Courses" link is present. The next section, "Latest Challenges", lists four completed challenges: "prime number" (Medium, 0%, 10 max score), "factorial number" (Easy, 100%, 25 max score), "Add Two Numbers" (Easy, 100%, 15 max score), and "Print Linked List In Reverse" (Easy, 100%, 25 max score), each with a "Try Again" button. The footer contains a dark bar with the text "© 2024 An-Najah Rank".

Figure 87: Student profile from student side

The screenshot shows the An-Najah Rank dashboard. On the left, there's a profile section for 'Momen Odeh' with a photo, user ID '# 11923929', role 'student', and email 's11923929@stu.najah.edu'. Below it is an 'Edit Profile' button. In the center, there's a 'Student Statistics' chart showing a 75.00% success rate in solving problems, with categories Easy (3/3 solved), Medium (0/3 solved), and Hard (0/3 solved). To the right, there's a 'Courses' section featuring 'Computer Programming' by Momen H. Odeh and Noor Aldeer, and 'Data Structure' by Momen H. Odeh and Noor Aldeen Abu Shehadeh. A 'Notifications' sidebar on the right lists several recent events: 'New contest added to Data Structure course' (1 hour ago), 'New challenge added to contest in Computer Programming course' (3 hours ago), 'New challenge added to contest in Computer Programming course' (3 hours ago), 'New challenge added to contest in Computer Programming course' (3 hours ago), and 'New challenge added to contest in Data Structure course' (23 hours ago).

Figure 88: Notifications when add new course or contest or challenge

This screenshot shows the 'All notification page' from the An-Najah Rank system. It displays a list of notifications in a scrollable format. The notifications are as follows:

- New contest added to Data Structure course (1 hour ago)
- New challenge added to contest in Computer Programming course (3 hours ago)
- New challenge added to contest in Computer Programming course (3 hours ago)
- New challenge added to contest in Computer Programming course (3 hours ago)
- New challenge added to contest in Data Structure course (23 hours ago)
- New contest added to Data Structure course (1 day ago)
- You have been added to the new course

Figure 89: All notification page

NR An-Najah Rank

courses > 10636211

Data Structure

Description

A data structure is a way of organizing and storing data to perform operations efficiently. It defines the relationship between data elements, the operations that can be performed on the data, and the rules for organizing the data. Different types of data structures serve various purposes, and their selection depends on the specific requirements of a task or problem.

Contests

| | | | | | |
|-------------|-------------|--------|---------|-----------|------------|
| Linked List | Start After | 0 days | 0 hours | 2 minutes | 49 seconds |
|-------------|-------------|--------|---------|-----------|------------|

Solved Rate: 0% max Score: 25

View Contest

© 2024 An-Najah Rank

Figure 90: Course view before contest start in student side

NR An-Najah Rank

courses > 10636211

Data Structure

Description

A data structure is a way of organizing and storing data to perform operations efficiently. It defines the relationship between data elements, the operations that can be performed on the data, and the rules for organizing the data. Different types of data structures serve various purposes, and their selection depends on the specific requirements of a task or problem.

Contests

| | | | | |
|-------------|---------|----------|------------|------------|
| Linked List | 10 days | 22 hours | 38 minutes | 53 seconds |
|-------------|---------|----------|------------|------------|

Solved Rate: 0% max Score: 25

View Contest

© 2024 An-Najah Rank

Figure 91: Course view after contest start in student side

The screenshot shows a contest page for 'Linked List'. At the top, there are navigation links: 'courses > 10636211 > contests > 81'. Below this, the title 'Linked List' is displayed with a 'Description' section. The 'Description' section states: 'A linked list is a linear data structure where elements, called nodes, are connected through pointers, forming a sequence. Each node contains data and a reference to the next node in the sequence.' There is also a 'Remaining time' section showing 9 days, 23 hours, 10 minutes, and 8 seconds. A challenge card for 'Print Linked List In Reverse' is shown, indicating it is an 'Easy' challenge with a success rate of 5.66% and a max score of 25. A 'Try Again' button is visible at the bottom right of the challenge card.

Figure 92: Contest view in student side

The screenshot shows a code editor with a C program. The code defines a 'Node' structure and two functions: 'printReverse' and 'insertNode'. The 'printReverse' function prints the data of each node in reverse order. The 'insertNode' function inserts a new node at the head of the list. The code editor includes line numbers and syntax highlighting. At the bottom, there are 'Run Code' and 'Submit Code' buttons.

```

18 struct Node* current = head;
19 while (current->next != NULL) {
20     current = current->next;
21 }
22
23 current->next = newNode;
24 return head;
25 }
26 void printReverse(struct Node* head) {
27     if (head == NULL) {
28         return;
29     }
30     printReverse(head->next);
31     printf("%d ", head->data);
32 }
```

The screenshot shows the results of running the code. It indicates 'TestCase 0' failed with a 'Compile Time Error'. The error message from the compiler is displayed, showing a syntax error in the 'insertNode' function: 'error: expected ';' before 'return''. The code editor highlights the problematic line: '25 | return head; | ~~~~~'. Below the error message, a warning is shown: 'warning: control reaches end of non-void function [-Wreturn-type]'. At the bottom, there is a copyright notice: '© 2024 An-Najah Rank'.

TestCase 0 X

Compile Time Error

Compiler Message

```

/app/code/Momen/cTest.c: In function 'Node* insertNode(Node*, int)':
/app/code/Momen/cTest.c:24:28: error: expected ';' before 'return'
24 |     current->next = newNode
|          ^
|          ;
25 |     return head;
| ~~~~~
/app/code/Momen/cTest.c:24:19: warning: control reaches end of non-void function [-Wreturn-type]
24 |     current->next = newNode
| ~~~~~~^~~~~~
```

Figure 93: when run code and there is a compile error

courses > I0636211 > contests > 81 > challenges > 47 > problem

Print Linked List In Reverse

[Problem](#)[Submissions](#)[Leaderboard](#)

Difficulty: Easy

Max Score: 25

Total Submission: 1

Input Format

The first line contains an integer , the number of elements in the linked list.

The next lines contain an integers for the linked list data separated by space.

Constraints

-

Output Format

an integers of reverse linked list data separated by space.

Simple Input 0

```
3
1 2 3
```

Sample Output 0

```
3 2 1
```

Explanation 0

this is a sample of reverse print linked list.

Dark mode: C

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <math.h>
4 #include <stdlib.h>
5
6 struct Node {
7     int data;
8     struct Node* next;
9 };
10 struct Node* insertNode(struct Node* head, int data) {
11     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
12     newNode->data = data;
13     newNode->next = NULL;
14
15     if (head == NULL) {
16         return newNode;
17     }
18
19     struct Node* current = head;
20     while (current->next != NULL) {
21         current = current->next;
22     }
```

[Run Code](#)[Submit Code](#)**TestCase 0 ✓**

Congratulations, you passed the sample test case.

Input (stdin)

```
3
1 2 3
```

Your Output (stdout)

```
3 2 1
```

Expected Output

```
3 2 1
```

Figure 94: Challenge view and run code in student side

The screenshot shows a challenge submission page. At the top, the navigation bar includes the NR An-Najah Rank logo and user icons. Below the navigation, the URL path is shown: courses > 10636211 > contests > 81 > challenges > 47 > submissions > 37. The challenge title is "Print Linked List In Reverse". A navigation bar below the title has tabs for "Problem", "Submissions" (which is selected), and "Leaderboard".

The "Submission Details" section shows the submission was submitted at 1/11/2024, 12:45:20 AM and scored 0. Below this, a row of test cases indicates results: ✓ Test Case #0, ✗ Test Case #1, ✗ Test Case #2, ✗ Test Case #3, and ✗ Test Case #4.

The "Submitted Code" section shows the code in a code editor. The language is set to C. The code includes standard library includes and a main function that prints the numbers 3, 2, and 1 in reverse order. The code editor has an "Open in editor" button.

At the bottom of the page is a footer with the text "© 2024 An-Najah Rank".

Figure 95: submit code not pass all test cases

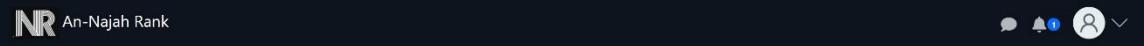
This screenshot shows a successful submission for the same challenge. The URL path is identical to Figure 95. The challenge title is "Print Linked List In Reverse". The "Submissions" tab is selected in the navigation bar.

The "Submission Details" section shows the submission was submitted at 1/10/2024, 6:15:26 PM and scored 25. Below this, a row of test cases indicates results: ✓ Test Case #0, ✓ Test Case #1, ✓ Test Case #2, and ✓ Test Case #3.

The "Submitted Code" section shows the code in a code editor. The language is set to C. The code defines a Node structure and an insertNode function that inserts a new node at the head of a linked list. The code editor has an "Open in editor" button.

At the bottom of the page is a footer with the text "© 2024 An-Najah Rank".

Figure 96: Submit the code



courses > 10636211 > contests > 81 > challenges > 47 > submissions

Print Linked List In Reverse

| Problem | Submissions | Leaderboard | | | |
|------------------------------|-------------|-------------------------------|----------------|-------|------------------------------|
| Problem | Language | Time | Result | Score | |
| Print Linked List In Reverse | c | Wed, 10 Jan 2024 22:45:20 GMT | Wrong Answer ✗ | 0 | <button>View Result</button> |
| Print Linked List In Reverse | c | Wed, 10 Jan 2024 22:46:53 GMT | Accepted ✓ | 25 | <button>View Result</button> |

© 2024 An-Najah Rank

Figure 97: Student submissions in student side

3.4.5 Sample of responsive design:

The figure consists of three side-by-side screenshots of a web application interface, likely a dashboard or communication platform for An-Najah Rank.

Figure 98: Conversation responsive (Left): This screenshot shows the "Conversations" section. It lists five recent messages from different users, each with a profile picture and timestamp. The users are Momen Odeh, Momen H. Odeh, NAMAS, Mohammad Muneer, and Mohee Qwareeq. The messages are: "Hello" (Momen Odeh), "How are you?" (Momen H. Odeh), "Hello Momen" (NAMAS), "I want to ask you about the linked list assignment How to take the size of linked list??" (Mohammad Muneer), and "you can take it by reading the first argument in the input stream." (Mohee Qwareeq). Below the list is a message input field with "Enter Message" placeholder and a "Send" button.

Figure 99: Chatting responsive (Middle): This screenshot shows a detailed view of a conversation between two users. The user on the left (Momen Odeh) has sent "Hello" and "How are you?". The user on the right (Momen H. Odeh) has responded with "Hello Momen". Momen H. Odeh then asks a question about a linked list assignment. Momen Odeh replies with an explanation. The conversation continues with "Ok, thank you." and "You're Welcome". At the bottom, there is a message input field and a "Send" button.

Figure 100: create course responsive (Right): This screenshot shows the "Create Course" form. It includes fields for "Course Number" (empty), "Course Name" (empty), and "Description" (empty). There are also fields for "Background Image" (choose file, no file chosen) and "Students Excel File" (choose file, no file chosen). A note at the bottom states "* should enter Students Excel File with .xlsx extension". At the bottom right are "Cancel Changes" and "Save Changes" buttons.

Figure 98: Conversation responsive

Figure 99: Chatting responsive

Figure 100: create course responsive

Momen Odeh
11923929
student
s11923929@stu.najah.edu

Edit Profile

Student Statistics

| Difficulty | Solved | Total | Rate |
|------------|--------|-------|---------|
| Easy | 3/3 | 3 | 100.00% |
| Medium | 0/1 | 1 | 0% |
| Hard | 0/0 | 0 | 0% |

Courses

- Computer Programming**
Momen H. Odeh
Noor Aldeen Abu Shehadeh
- Data Structure**
Momen H. Odeh
Noor Aldeen Abu Shehadeh

[Show all Courses](#)

</> Latest Challenges

factorial number

Difficulty: Easy Success Rate: 0%
Max Score: 25 [Solve Challenge](#)

Add Two Numbers ✓

Difficulty: Easy Success Rate: 100%
Max Score: 15 [Solve Challenge](#)

Print Linked List In Reverse ✓

Difficulty: Easy Success Rate: 100%
Max Score: 25 [Solve Challenge](#)

© 2024 An-Najah Rank

Notifications

- Running test case on contests selected with challenge id 47 finish successfully
1 hours ago
- Running test case on contests selected with challenge id 47 finish successfully
5 hours ago
- Running test case on contests selected with challenge id 47 finish successfully
5 hours ago
- Similarity data ready for submissions in Data Structure course
5 hours ago
- Your added as moderator in Computer Programming course
1 days ago
- Similarity data ready for submissions in Data Structure course
12 days ago
- Failed in calculate similarity for submissions in Data Structure course, please try again later
17 days ago
- Failed in calculate similarity for submissions in Data Structure course, please try again later
17 days ago
- Failed in calculate similarity for submissions in Data Structure course, please try again later
17 days ago
- Failed in calculate similarity for submissions in Data Structure course, please try again later
24 days ago
- Similarity data ready for submissions in Data Structure course
25 days ago
- Running test case on contests selected with challenge id 42 finish successfully
25 days ago
- Running test case on contests selected with challenge id 41 finish successfully
26 days ago
- Running test case on contests selected with challenge id 41 finish successfully
26 days ago

© 2024 An-Najah Rank

administration > challenges > create-challenge

Details

Challenge Difficulty
 Easy

Specify Language
 Java C C++ Python
 JavaScript Regex

Challenge Name

Description

Problem Statement

Input Format

Constraints

Output Format

Challenge Privacy
 make the challenge public.

Tags
 Add

[Cancel Changes](#) [Save Changes](#)

© 2024 An-Najah Rank

Figure 101: Profile responsive

Figure 102: Notification responsive

Figure 103: 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.

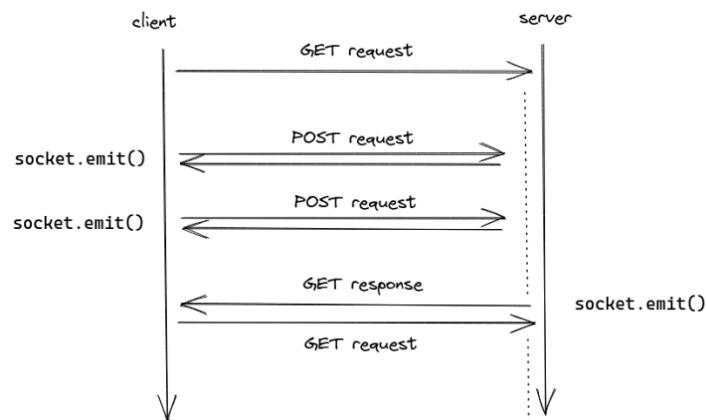


Figure 104: Socket IO

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.

| P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
|-----------------------------------|----|----|---|---|---|---|---|---|---|---|---|-----------------------|--|--|---|
| طباعة قائمة الحضور والغياب | | | | | | | | | | | | | | | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | اول 2024-2023 | اسم المدرس : سامر ناجح ناجي العزندى (10636456/1) | اسم المساق : أنظمة التشغيل الموزعة (1) | 1 |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | رقم المقاعة : 1111170 | الموعد الزمني : احد/ثلاثة/خميس 1-2 | 3 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | رقم الطالب | رمز الكتبة | 6 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | اسم الطالب | رقم الطالب | 1 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 7 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 8 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 9 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 10 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 11 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 12 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 13 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 14 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 15 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 16 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 17 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 18 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 19 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 20 | |

Figure 105: 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.

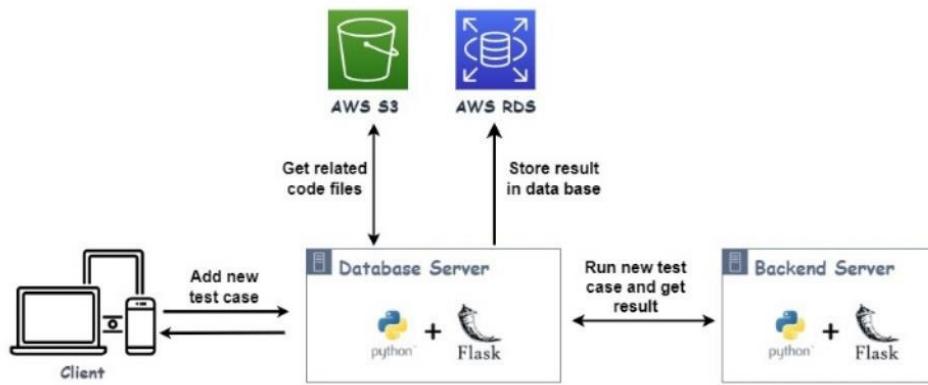


Figure 106: Add new test case diagram

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.

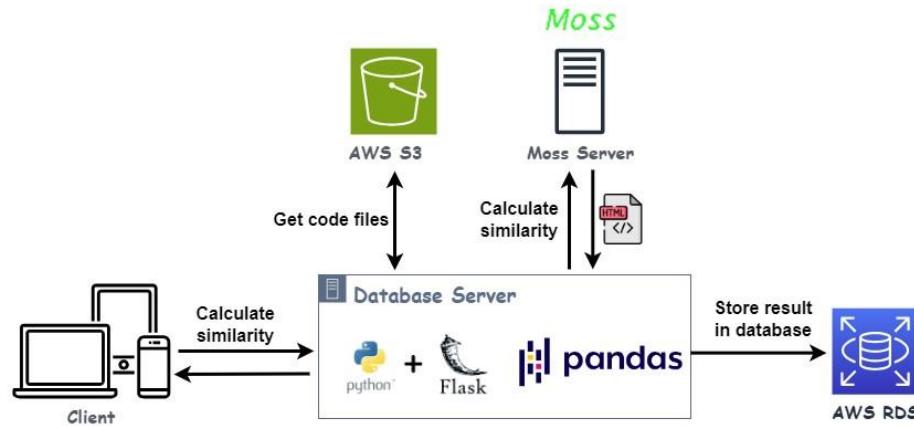


Figure 107: Calculate similarity operation

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:

| 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 |

Figure 108: Moss similarity result

```

FileSimilarity/contest81-challenge47/Mohammad_Muneer-11235499.c (73%)
4-12
21-36

FileSimilarity/contest81-challenge47/Noor_Aldeen_Muneer_Abu_Shehadeh-11923513.c (58%)
6-13
52-44

```

```

FileSimilarity/contest81-challenge47/Mohammad_Muneer-11235499.c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* insertNode(struct Node* head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    return newNode;
}

void printReverse(struct Node* head) {
    if (head == NULL) {
        return;
    }
    printf("%d ", head->data);
    printReverse(head->next);
}
}

FileSimilarity/contest81-challenge47/Noor_Aldeen_Muneer_Abu_Shehadeh-11923513.c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* insertNode(struct Node* head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    if (head == NULL) {
        return newNode;
    }
    struct Node* current = head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
    return head;
}

```

Figure 109: 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:

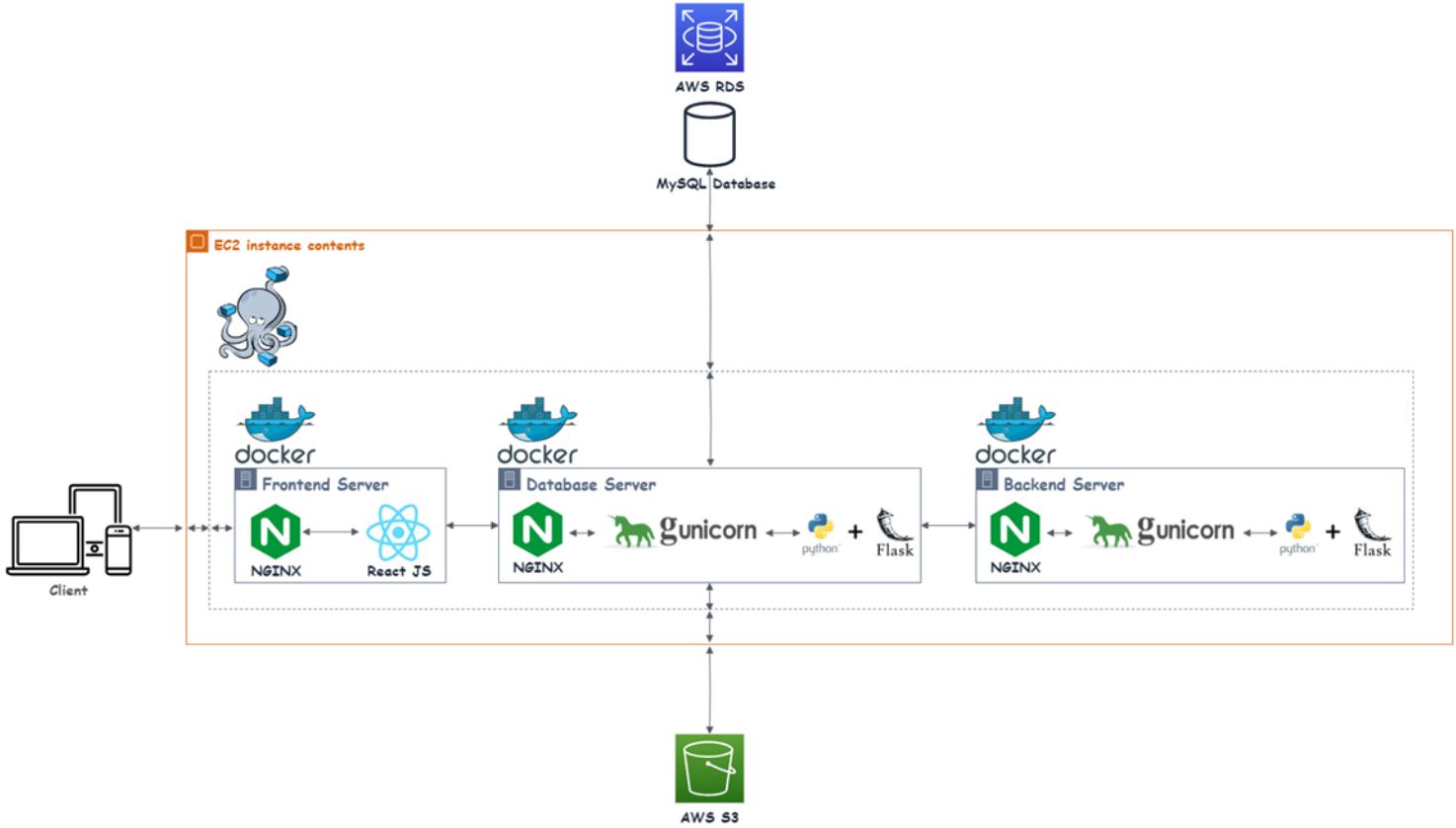


Figure 110: Deployment process

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 is not entirely within our control, and its occasional unavailability may disrupt our similarity calculation processes, potentially affecting the availability of the similarity feature system.

Chapter 4: Result and Analysis

CHAPTER

4

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

CHAPTER

5

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

CHAPTER

6

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 calculation for the submission code of the 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> .
- **Subprocess management.** Available :
<https://docs.python.org/3/library/subprocess.html> .