Major Project Report on

Coding Platform and Tool for Plagiarism Detection

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY
by

Shaulendra Kumar (201IT159)
Mayur Jinde (201IT135)
Ritik Kumar (201IT250)

under the guidance of

Prof. Ram Mohana Reddy Guddeti

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

April, 2024

DECLARATION

We hereby *declare* that the Major Project Work Report entitled ***"Coding Platform and Tool for Plagiarism Detection"*** , which is being submitted to the **National Institute of Technology Karnataka, Surathkal**, for the award of the Degree of Bachelor of Technology in Information Technology, is a *bonafide report of the work carried out by us.* The material contained in this Major Project Report has not been submitted to any University or Institution for the award of any degree.

S.No.	*Register No.*	*Name and Signature of the Student(s)*
(1)	201IT159	Shaulendra Kumar Shaulendra
(2)	201IT135	Mayur Jinde [illegible]
(3)	201IT250	Ritik Kumar Ritik

Department of Information Technology
Place : NITK, Surathkal
Date : 08/04/2024

CERTIFICATE

This is to *certify* that the Major Project Work Report entitled ***"Coding Platform and Tool for Plagiarism Detection"*** submitted by

S.No.	*Register No.*	*Name of the Student(s)*
(1)	201IT159	Shaulendra Kumar
(2)	201IT135	Mayur Jinde
(3)	201IT250	Ritik Kumar

as the record of the work carried out by them, is *accepted as the B.Tech.(IT) Major Project work report submission* in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Information Technology in the Department of Information Technology, NITK Surathkal.

[illegible] 09/04/2024
Prof. Ram Mohana Reddy Guddeti
Professor (HAG Scale)
Department of Information Technology
NITK Surathkal

[illegible] 9/4/24
Chairman, DUGC

CHAIRMAN - DUGC
Department of Information Technology
NITK Surathkal, Srinivasnagar P.O.
Mangaluru 575 025, INDIA

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards our project guide, Prof. Ram Mohana Reddy Guddeti for his invaluable guidance, support and mentorship throughout the duration of this project. Through our in-depth discussions with him, we have been able to develop robust solutions to the chosen problem and iteratively improve upon the work presented through this report. We are extremely grateful to our project guide, Prof. Ram Mohana Reddy Guddeti for taking the time to review and provide constructive feedback on our work, and for being a constant source of inspiration and support. Without his motivation and insights, the successful completion of this project would not have been possible.

ABSTRACT

Coding skills have become increasingly important in the digital age, and competitive coding platforms have emerged as valuable resources to hone these skills. This project focuses on developing a robust coding platform that collaborates with the popular and widely used coding platforms, namely: GitHub, Codeforces, Codechef, Leetcode, Hackerrank, Atcoder etc. This project not only facilitates the coding competition but also includes the improved methods of plagiarism detection. The main objective of this project is to develop an easy-to-use, multi-feature coding platform that meets the needs of beginners and experts Users may have a wide variety of coding challenges, problems with variety of practice, and competitions, in order to improve their coding skills, and problem solving skills. Further, opportunities to develop debugging and capabilities will be provided to the users. The coding platform will support programming languages (C, C++, Java, Python) and thus provides a collaborative coding environment to foster the learning and communication among the participants. Finally, the platform enables the instructor to detect the coding web-sources of plagiarism with similarity score and also checks with the similarity score of the code generated by AI tools like ChatGPT.

Keywords— AI Tools, ChatGPT, Coding Platform, Plagiarism Checker, Programming Module, Instructor Module.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview

The coding landscape is undergoing an unprecedented surge, fueled by the universal understanding that computers are integral to our collective future. Despite this exponential growth, the domain of competitive coding currently hinges on just two major platforms, serving a user base that surpasses the million mark. This glaring gap in the market presents a tremendous opportunity for innovation and expansion, precisely the mission behind our ambitious project.

Our initiative doesn't merely aim to replicate existing platforms; rather, it endeavors to redefine the coding experience. Going beyond the standard repertoire of problem-solving, we are introducing cutting-edge features tailored to capture the attention of both seasoned programmers and newcomers within academic circles. The system is intricately designed, encompassing distinct realms such as problem space, individual space, contest space, and personal development space.

At the core of our platform lies a robust plagiarism checker, a tool meticulously crafted to evaluate and score each individual based on the percentage of similarity in their code. This groundbreaking feature addresses a pervasive challenge—cheating within the coding community—by fostering a culture of originality and accountability.

By integrating this plagiarism detection mechanism, our platform aspires to create an environment where meritocracy thrives, and users are recognized and rewarded for their authentic contributions. We envision a community that not only excels in technical proficiency but also upholds the principles of integrity and fair play. Our holistic approach seeks to empower users, offering a multifaceted and enriching coding experience that transcends traditional boundaries. In essence, our project is a catalyst for transforming the coding landscape, shaping it into a vibrant, ethical, and inclusive ecosystem for all enthusiasts.

1.2 Popular Coding Platforms

GitHub is a leading platform in the coding community, renowned for its robust version control system using Git. It serves as a central hub for developers to store, manage, and collaborate on code, whether for open-source projects or private repositories. Its intuitive interface, extensive documentation, and integration with various development tools make it a preferred choice for software development teams.

LeetCode and HackerRank are essential resources for developers refining their coding skills and preparing for technical interviews. LeetCode hosts a vast repository of categorized coding problems by difficulty level, while HackerRank offers challenges, tutorials, and interview prep materials. Both platforms foster a supportive community and provide tools for developers to enhance their problem-solving abilities and coding proficiency. TopCoder and Codewars are renowned for their focus on competitive programming and algorithmic challenges. These platforms host coding competitions where developers test their skills against others globally, honing problem-solving abilities and programming proficiency.

CodeChef is a leading platform for competitive programming, offering monthly coding challenges, cook-offs, and long contests across various difficulty levels. With a strong emphasis on algorithmic problem-solving, CodeChef provides programmers with opportunities to enhance their coding skills and participate in global competitions. Similarly, Codeforces is renowned for its regular contests and extensive problem archive, challenging participants to devise efficient solutions within strict time constraints. Both platforms foster vibrant communities where programmers can engage in discussions, share insights, and learn from each other's experiences. GeeksforGeeks (GfG) stands out as a comprehensive resource for computer science topics, offering tutorials, coding practice, and interview preparation materials.

GitLab offers similar features to GitHub but stands out with its comprehensive DevOps capabilities. Alongside version control, it provides project management, CI/CD pipelines, and a container registry within a unified platform. This integration streamlines development workflows, making GitLab popular among organizations seeking end-to-end solutions for software development.

1.3 Impact of Generative AI

Generative AI has brought forth transformative changes to coding platforms and the coding culture, impacting various facets of software development in unique ways. Firstly, it has revolutionized the process of code generation and automation. With the advent of advanced generative models like GPT (Generative Pre-trained Transformer), developers now have access to tools capable of generating code snippets or even entire programs based on natural language descriptions. This automation streamlines repetitive coding tasks, accelerates development timelines, and allows programmers to focus more on high-level problem-solving and design aspects rather than mundane coding details.

Secondly, generative AI has facilitated code refactoring and optimization. By analyzing large codebases and learning from established coding practices, AI-powered tools can identify inefficiencies, detect code smells, and suggest improvements to enhance code readability, maintainability, and performance. This leads to cleaner, more efficient codebases and helps developers adhere to coding standards and best practices, ultimately resulting in higher-quality software products.

Moreover, generative AI has made significant contributions to code review and quality assurance processes. AI models can automatically identify potential bugs, vulnerabilities, or inconsistencies in code by analyzing patterns and dependencies within the codebase. This automated analysis speeds up the code review process, reduces manual effort, and helps mitigate the risk of introducing errors or security vulnerabilities into production systems, thus improving overall code quality and reliability.

Furthermore, in the realm of learning and education, generative AI-powered coding platforms have emerged as valuable educational resources for aspiring programmers and coding enthusiasts. By offering interactive coding exercises, personalized feedback, and adaptive learning experiences, these platforms make learning coding skills more engaging and effective. AI-driven recommendation systems tailor learning paths to individual preferences and learning styles, democratizing access to coding education and empowering learners to acquire skills at their own pace.

1.4 Motivation

This project serves a pivotal purpose—to furnish users with a dynamic platform for showcasing, refining, and mastering their problem-solving prowess. Navigating the complexities of learning Data Structures and Algorithms can be challenging, yet our endeavor, Codescript, is meticulously designed to render this journey both accessible and enjoyable. Consider it your one-stop solution, meticulously crafted to cater to all your coding needs.

Our motivation stems from a firsthand understanding of the coding landscape, having been active participants ourselves. Through this immersion, we identified a myriad of issues plaguing existing coding platforms. Recognizing the expansive potential within the coding market, characterized by continuous growth and an insatiable demand for more challenging problems, we embarked on a mission to redefine the coding experience.

As a team, we firmly believe in the philosophy of constant improvement. While existing platforms have paved the way, we see ample room for innovation and enhancement. Replicating successful models could be profitable, but our aim is loftier—to introduce novel challenges and features that resonate with the evolving needs of the coding community. Human psychology craves variety and choices; our platform seeks to satiate this inherent desire by offering a diverse array of problems and solutions.

The market's incessant expansion creates an opportune moment for us to inject fresh perspectives and tougher challenges. In recognizing this potential, we have positioned ourselves to seize this opportunity. Our proximity to the coding domain, coupled with a keen understanding of user preferences, propels us to create a platform that not only meets but surpasses the expectations of coding enthusiasts. In essence, Codescript is not just a platform; it is a dynamic ecosystem designed to evolve with the ever-changing landscape of coding, providing users with an unparalleled and enriching experience.

CHAPTER 2
LITERATURE REVIEW

2.1 Background and Related Works

Research Paper "Communication." coding platform for students" offers an interactive web-based coding platform for students, designed to make academic coding fun and effective. It offers personalized lessons, interactive exercises, real-time feedback, collaboration tools, games resources for.Support for multiple programming languages and cloud storage storage[1].
Another document "AST-based code." plagiarism detection algorithm" Provides a new method for detecting code plagiarism using Abstract Syntax Trees (ASTs), which represent the structure of a system, regardless of its type. This algorithm generates ASTs for code comparison and things like node-edge counts Considering their similarity calculates and labels.It can detect plagiarism, even with unambiguous code changes, and has been shown to work well with large data sets.Compared to traditional methods, it is robust, efficient, and functional in programming languages and plagiarism detection. [2].
Karnalim's paper "A New Approach to Checking Code Theft Based on Minimum Hierarchies" introduces a new approach to code theft by analyzing subsets of executable code represented as token sequences This approach this tokenizes rules and Jaccard index and Levenshtein distance to compare code structures Similarity measures are used. It provides advantages over traditional methods by providing robust capabilities for confusing ideas, being more efficient, and working across programming languages[3].
Mayank Agarwal and D.K. Sharma's paper "State of the Art on Source Code Plagiarism Detection" provides a comprehensive overview of the current state of source code plagiarism detection It includes methods and algorithms that happen as methods based on systematic analysis in programming languages emphasize effectiveness. The authors acknowledge the challenges in this area, such as the definition of plagiarism and data set diversity[4].

2.2 Outcome of Literature Review

The literature review highlights key findings in computer science and plagiarism detection. One paper describes a web-based coding platform that is engaging for students, offering personalized learning, interactive exercises, real-time feedback and collaboration tools, supporting multiple languages Another paper presents a novel approach using Abstract Syntax Trees (ASTs) for rule-plagiarism detection, which better identifies similarities even in ambiguous rules It is more robust and efficient than traditional methods, applied to language in various fields. Karnalim's paper introduces an alternative approach based on minimum rule sets, using similarity measures such as token sequences and Jaccard indexes etc. It is durable, efficient and versatile. Finally, a paper provides an overview of source code plagiarism detection techniques, highlights the effectiveness of structural analysis in languages, and acknowledges challenges such as plagiarism definitions and data types. Table 2.1 summarizes the outcome of literature review in terms of key limitations.

Table 2.1: Outcome of Literature Review

Paper	**Outcome**	**Limitation**
Interactive coding platform for students	The paper aims at enhancing engagement in coding tasks within educational settings.	Innovative approach has limitations: requires validation and faces usability and integration challenges in education.
An ast-based code plagiarism detection algorithm	The paper focuses on innovative approaches to identifying code similarities.	Challenges may arise from validating against diverse code repositories and addressing scalability issues when handling large-scale codebases.
A low-level structure-based approach for detecting source code plagiarism.	This work introduces a "low-level structure-based approach for detecting source code plagiarism," likely providing novel insights into combating code plagiarism through structural analysis.	Potential limitations may involve the need for additional empirical validation across diverse codebases, along with challenges regarding the algorithm's adaptability to different programming languages and styles..
A state of art on source code plagiarism detection	This paper titled "A state of art on source code plagiarism detection," likely provides a comprehensive review of the current landscape of source code plagiarism detection methods and technologies.	Potential limitations includes the need for deeper analysis of emerging trends and challenges in source code plagiarism detection, along with empirical validation against real-world code repositories.

2.3 Problem Statement

To develop a coding Platform where NITK students can practice coding problems (Problems can be added by course instructors or the problems will be listed along with the topic tag from different codeing platforms like codechef,codeforces,hackerrank,leetcode,etc) and evaluating the code with plaigarism checker which will check whether the code is taken from other coding platform.

2.4 Objectives and Features of the Project

To create a coding platform that supports the following objectives and the features

(1) Problem Space: A plethora of problems in different categories. Each issue would be having a problem statement (that includes constraints), test cases, tags, and rating.

(2) Coding Space: Users can either use the website's basic IDE to write code and submit or upload their solution file (C, C++, Python), which will be compiled, evaluated, and corresponding results will be shown to him/her.

(3) Personal Development Space: This space is completely dedicated to a user. Here, the website recommends problems randomly based on the filters decided by the user.

(4) Contest space: Users can create their own contest with the problems of their liking and invite their friends and battle it out. Contest problems will include user-defined test cases, language options, and duration.

(5) Problem list: Users can create a list of problems of their liking and share it with others using a shareable link.

(6) User Profile: Users rating, solved problems, problem lists, friends.

(7) Evaluating the plaigarism of the submitted answers by the users.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 Architecture of Integrated Coding Platform

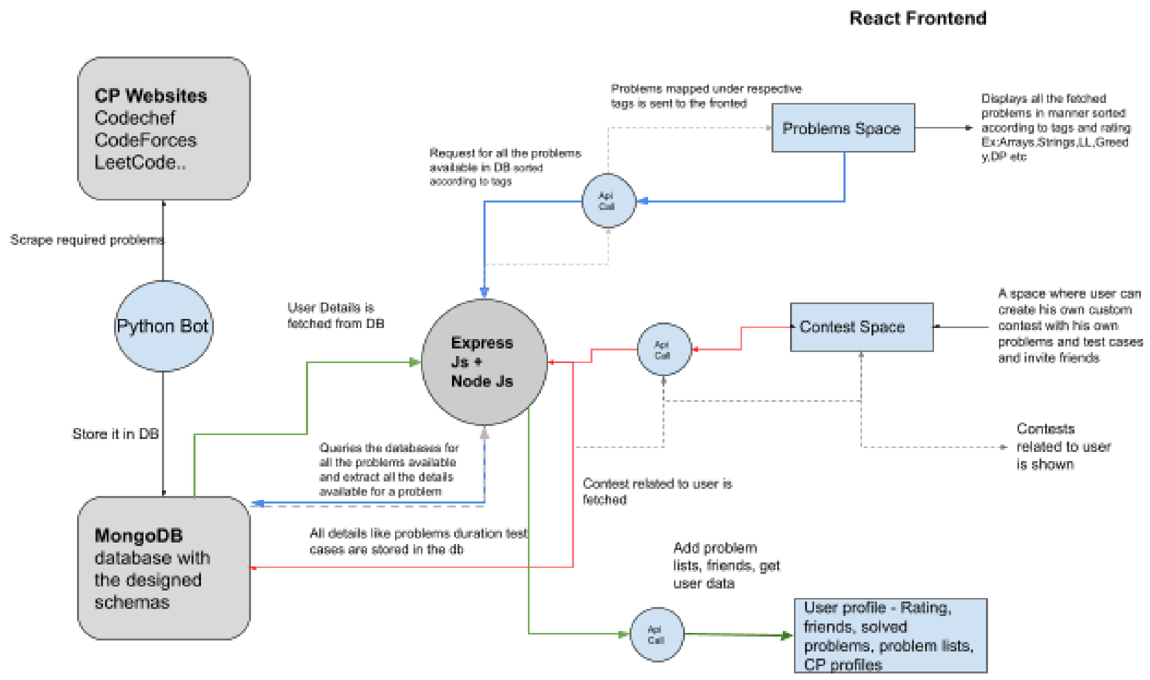

React Frontend
CP Websites
Codechef
CodeForces
LeetCode..
Scrape required problems
Python Bot
Store it in DB
MongoDB
database with
the designed
schemas
Request for all the problems available in DB sorted according to tags
Api Call
Problems mapped under respective tags is sent to the fronted
Problems Space
Displays all the fetched problems in manner sorted according to tags and rating Ex:Arrays,Strings,LL,Greedy,DP etc
User Details is fetched from DB
Express Js + Node Js
Api Call
Contest Space
A space where user can create his own custom contest with his own problems and test cases and invite friends
Contests related to user is shown
Queries the databases for all the problems available and extract all the details available for a problem
Contest related to user is fetched
All details like problems duration test cases are stored in the db
Add problem lists, friends, get user data
Api Call
User profile - Rating, friends, solved problems, problem lists, CP profiles

Figure 3.1: Architecture of the Integrated Coding Platform

Figure 3.1 shows the Architecture of Integrated Coding Platform. It is showing the interactions between backend (Express.js) and frontend (React). There are different spaces according to their features. A python bot is web scrapper which scraps the problems from different websites. Python bot scraps problems from different websites like leethcode, codeforces, codechef and hackerrank and save them into database (MongoDB). There are many spaces for user to interact in frontEnd (React). Problem Space shows all the problems available in system, whenever user interacts with space it makes api calls to backend where backend fetches all the problems from database and send them back to frontend.In contest space all the public and private contest are shown. Whenever an user enters in this space, frontend make api call to backend and backend application check database and send back the fresh data to frontend.There are other spaces also which we will discuss further.

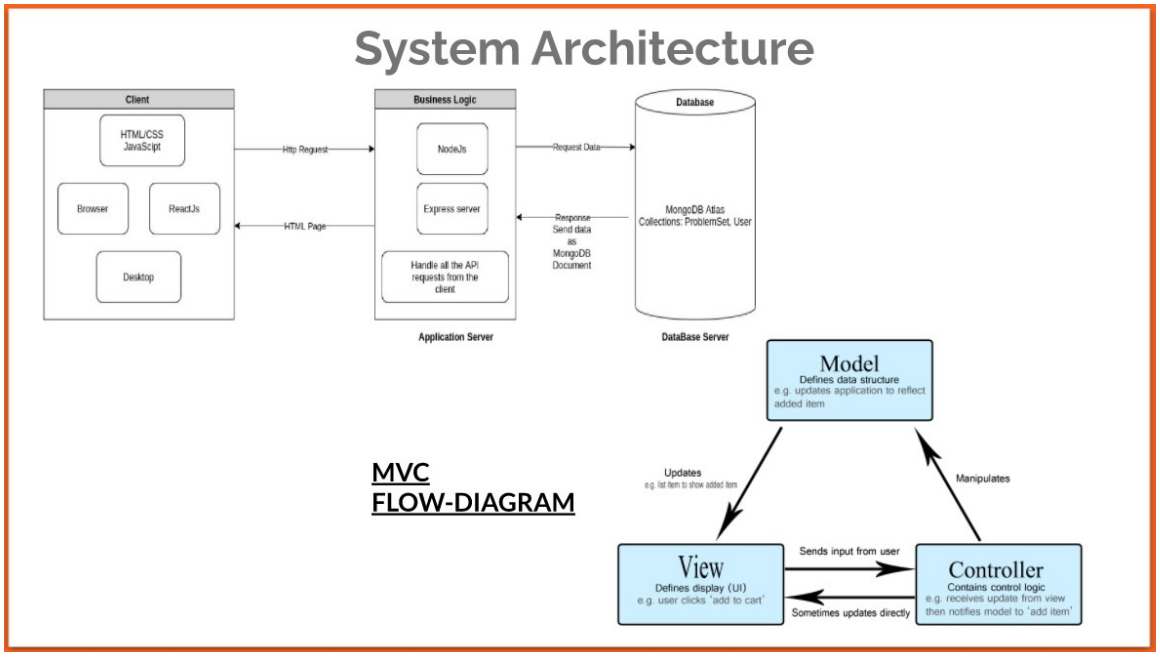

System Architecture
Client
HTML/CSS JavaScipt
Browser
ReactJs
Desktop
Http Request
HTML Page
Business Logic
NodeJs
Express server
Handle all the API requests from the client
Application Server
Request Data
Response Send data as MongoDB Document
Database
MongoDB Atlas Collections: ProblemSet, User
DataBase Server
MVC FLOW-DIAGRAM
Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Figure 3.2: MVC architecture

Figure 3.2 illustrates the Model-View-Controller (MVC) architecture, where the user interacts with the system through a desktop application or a web interface built with HTML, CSS, and JavaScript. This web interface, the view, translates user actions into signals for the controller. The controller, typically implemented using Node.js and Express.js, acts as the brains of the operation. It interprets user requests, validates them if necessary, and retrieves or updates data from the model. The model, often a MongoDB database in this case, encapsulates the core application data and business logic. Any changes to the data in the model trigger updates to the view, ensuring the user interface reflects the latest information. This separation of concerns promotes clean, maintainable, and reusable code. If a developer needs to modify the user interface, they only need to change the view components without affecting the controller or model logic. Likewise, changes to the data model can be implemented without impacting the presentation layer. This modularity makes the MVC architecture a popular choice for building complex and scalable software applications. This is the MVC architecture we have used inside our application to make it more scalable and enhanced.

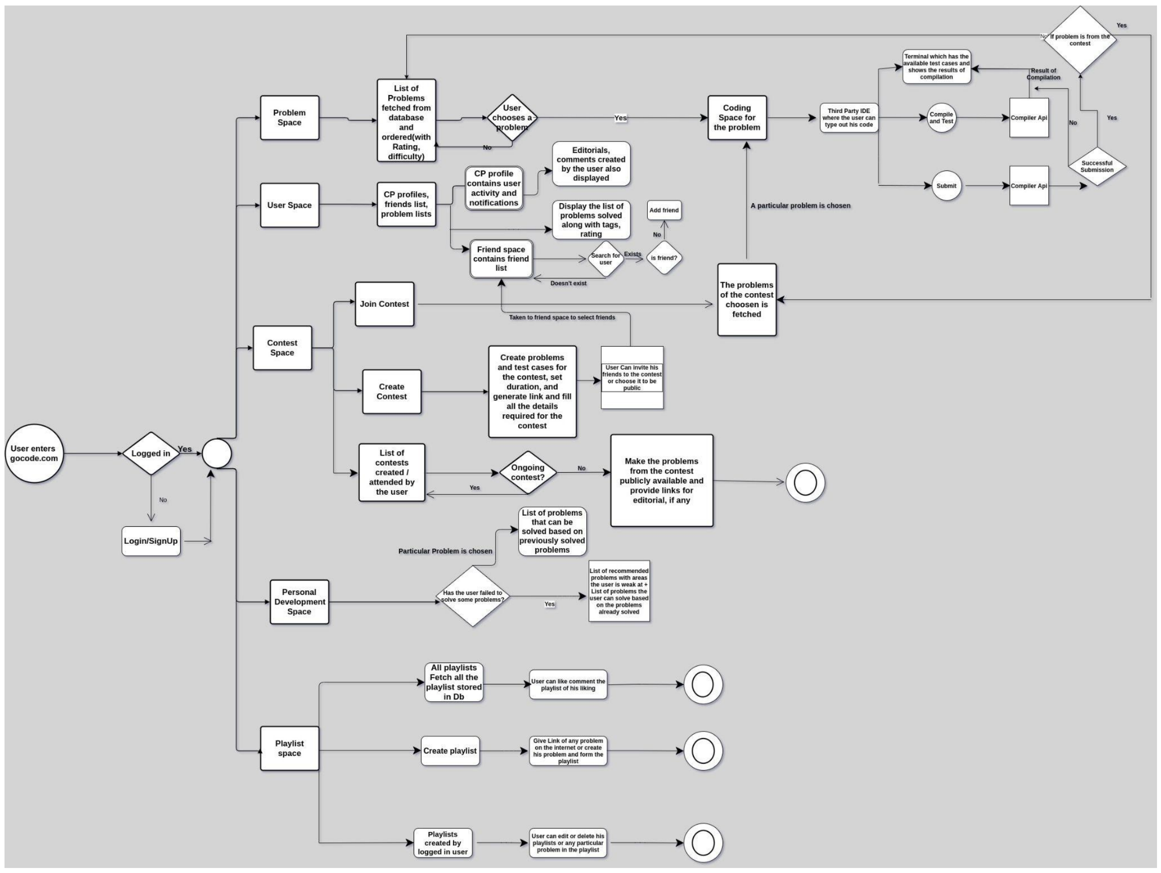

User enters gocode.com
Logged in
Yes
No
Login/SignUp
Problem Space
List of Problems fetched from database and ordered(with Rating, difficulty)
User chooses a problem
Yes
No
Coding Space for the problem
Third Party IDE where the user can type out his code
Terminal which has the available test cases and shows the results of compilation
Compile and Test
Compiler Api
Submit
Compiler Api
Result of Compilation
Successful Submission
No
Yes
If problem is from the contest
Yes
User Space
CP profiles, friends list, problem lists
CP profile contains user activity and notifications
Editorials, comments created by the user also displayed
Display the list of problems solved along with tags, rating
Friend space contains friend list
Search for user
Exists
Doesn't exist
Is friend?
No
Add friend
A particular problem is chosen
The problems of the contest choosen is fetched
Contest Space
Join Contest
Taken to friend space to select friends
Create Contest
Create problems and test cases for the contest, set duration, and generate link and fill all the details required for the contest
User Can invite his friends to the contest or choose it to be public
List of contests created / attended by the user
Ongoing contest?
No
Yes
Make the problems from the contest publicly available and provide links for editorial, if any
Personal Development Space
Has the user failed to solve some problems?
Particular Problem is chosen
List of problems that can be solved based on previously solved problems
Yes
List of recommended problems with areas the user is weak at + List of problems the user can solve based on the problems already solved
Playlist space
All playlists Fetch all the playlist stored in Db
User can like comment the playlist of his liking
Create playlist
Give Link of any problem on the internet or create his problem and form the playlist
Playlists created by logged in user
User can edit or delete his playlists or any particular problem in the playlist

Figure 3.3: Activity diagram

Figure 3.3 shows the activity diagram of different problem spaces considered in this coding platform. Further, it shows how a user interacts with the system. The user first goes through a login or sign-up process. Then, the user can see a list of problems they have solved, create playlists of problems, and view their contest space. The user can also view a list of all contests, create a contest, or choose a problem to participate in. The backend, which is built with Node.js and Express.js, handles all of the data processing and communication between the database and the frontend. The database stores information about the user, such as their profile, friends list, and problem lists. It also stores information about the contests, such as the problems, duration, and generated link. Figure 3.3 also shows the first two steps, namely: authentication and authorization.

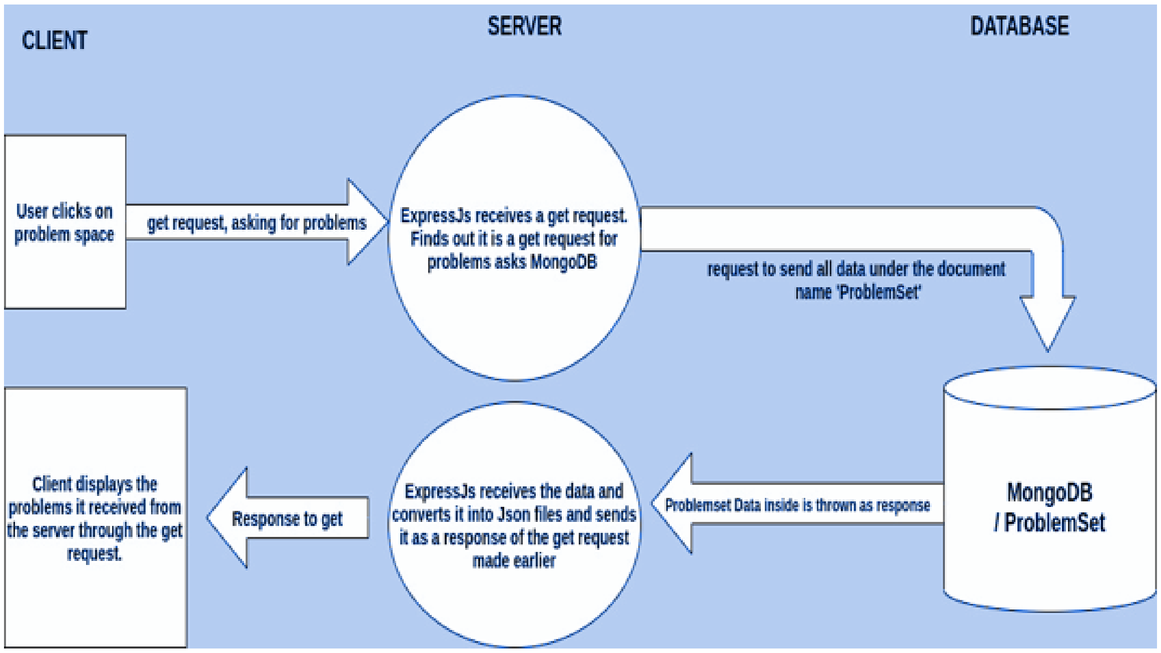

CLIENT
SERVER
DATABASE
User clicks on problem space
get request, asking for problems
ExpressJs receives a get request. Finds out it is a get request for problems asks MongoDB
request to send all data under the document name 'ProblemSet'
MongoDB / ProblemSet
Problemset Data inside is thrown as response
ExpressJs receives the data and converts it into Json files and sends it as a response of the get request made earlier
Response to get
Client displays the problems it received from the server through the get request.

Figure 3.4: Flow diagram of Problem space

Figure 3.4 explains the working flow of Problem Space. Whenever user enters into this space. FrontEnd makes api call to BackEnd then backend application comes into picture, backend makes connection with database (MongoDB) and fetch all the problems based on given condition and then send them back to FrontEnd. FrontEnd index them to user where user can select any problem and solve it. In this space problems are shown based upon topics which makes it easy for user to practice.

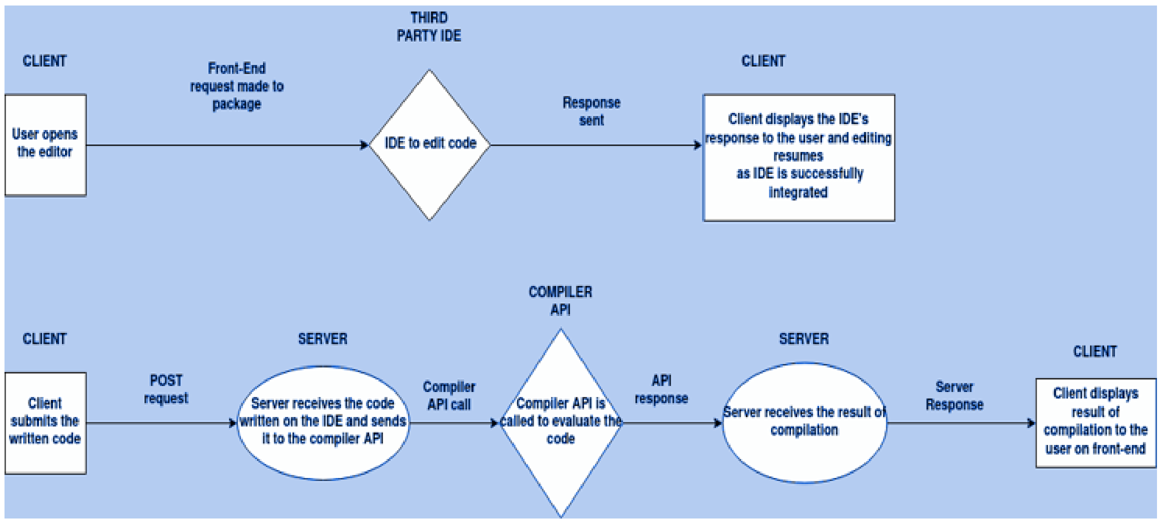

CLIENT
User opens the editor
Front-End request made to package
THIRD PARTY IDE
IDE to edit code
Response sent
CLIENT
Client displays the IDE's response to the user and editing resumes as IDE is successfully integrated
CLIENT
Client submits the written code
POST request
SERVER
Server receives the code written on the IDE and sends it to the compiler API
Compiler API call
COMPILER API
Compiler API is called to evaluate the code
API response
SERVER
Server receives the result of compilation
Server Response
CLIENT
Client displays result of compilation to the user on front-end

Figure 3.5: Flow diagram of Coding space

Figure 3.5 provides a comprehensive insight into the operational workflow within an Integrated Development Environment (IDE) and delineates the systematic process of problem submission for evaluation. Within the Coding Space environment, users are equipped with a versatile IDE, empowering them to efficiently compose and refine their code. The Testing feature within the coding space serves as a crucial tool, enabling users to input custom data sets and assess the output, thereby facilitating rigorous testing and debugging procedures. Upon completion of the coding task, users can seamlessly submit their code for evaluation, triggering a series of backend operations. As the code is submitted, it is seamlessly transmitted to the backend infrastructure through an Application Programming Interface (API), initiating a cascade of actions. The backend application orchestrates the interaction with the compiler API, which diligently processes the code and generates comprehensive results. These results are then meticulously compared with predefined test cases, enabling the backend to derive a precise evaluation score. Post-compilation, the code is securely stored within the database, ensuring traceability and archival of user submissions. Subsequently, the evaluation results are relayed back to the frontend interface, where they are elegantly presented to the user through an intuitive User Interface (UI). In the context of contest-based scenarios, the evaluation process extends to updating the user's score, thereby dynamically reflecting their progress and standing within the competition environment.

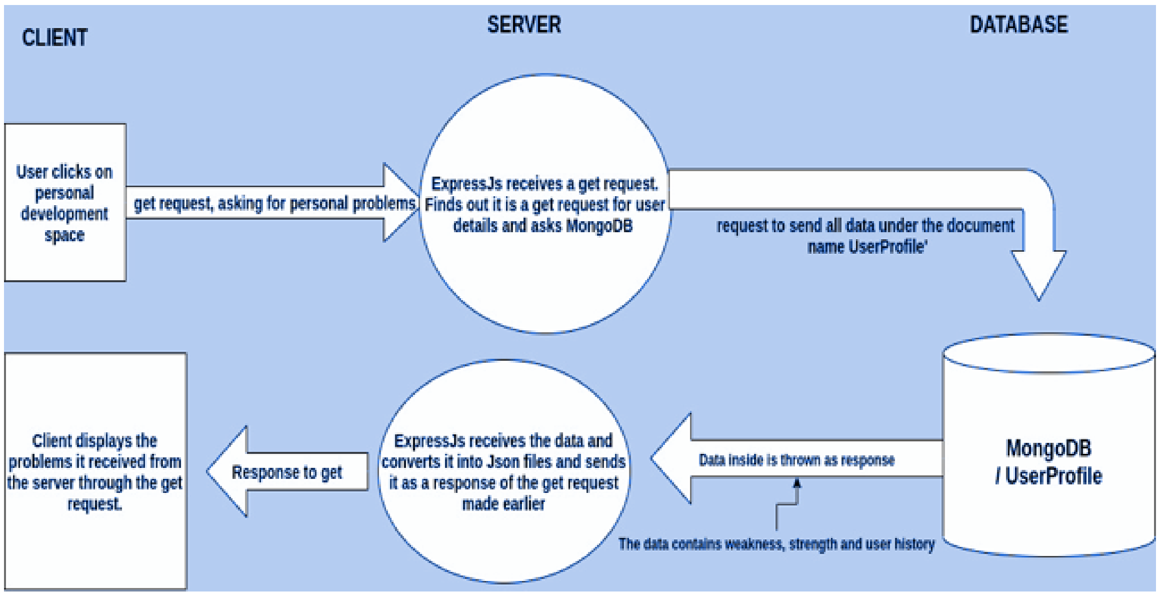

CLIENT
SERVER
DATABASE
User clicks on personal development space
get request, asking for personal problems
ExpressJs receives a get request. Finds out it is a get request for user details and asks MongoDB
request to send all data under the document name UserProfile'
MongoDB / UserProfile
Data inside is thrown as response
The data contains weakness, strength and user history
ExpressJs receives the data and converts it into Json files and sends it as a response of the get request made earlier
Response to get
Client displays the problems it received from the server through the get request.

Figure 3.6: Flow diagram of Personal Development Space

Figure 3.6 serves as a user-friendly guide to navigating the system and understanding how user data is handled. Upon a user's login, the frontend interface initiates a request to the backend system, specifying the user's details required for their personalized experience. The backend efficiently retrieves this information from the database, leveraging robust storage mechanisms like MongoDB to ensure data integrity and accessibility. Once retrieved, the backend undertakes a series of insightful analyses on the user's activity, including but not limited to, identifying patterns in problem-solving behaviors, assessing proficiency levels across various topics, and tracking progress over time. These analyses are crucial in providing users with meaningful insights into their learning journey and areas for improvement. Subsequently, the processed data is seamlessly transmitted back to the frontend, where it undergoes transformation into intuitive visualizations, such as dynamic graphs and interactive charts. These visual representations not only enhance user engagement but also facilitate a deeper understanding of their performance metrics. Moreover, the system may incorporate additional features such as personalized recommendations based on user behavior and performance trends, further enriching the user experience. Through this iterative process of data retrieval, analysis, and visualization, users are empowered to make informed decisions, track their progress effectively, and ultimately achieve their learning goals with confidence.

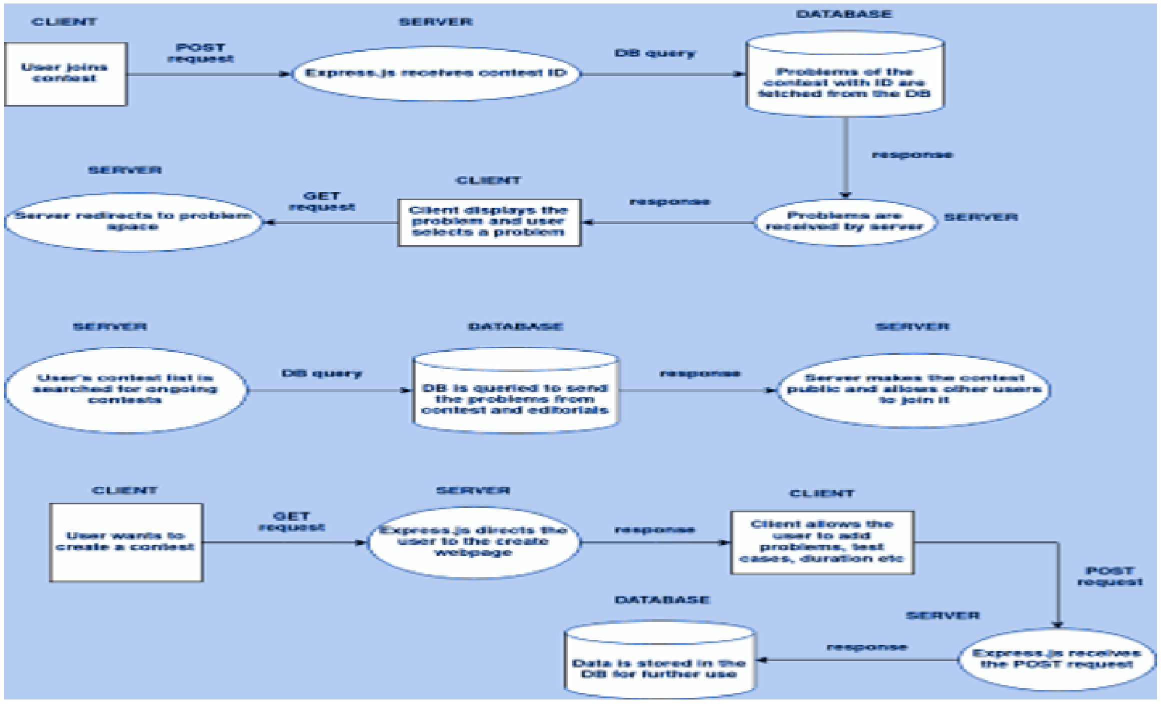

CLIENT
User joins contest
POST request
SERVER
Express.js receives contest ID
DB query
DATABASE
Problems of the contest with ID are fetched from the DB
response
SERVER
Problems are received by server
response
CLIENT
Client displays the problem and user selects a problem
GET request
SERVER
Server redirects to problem space
SERVER
User's contest list is searched for ongoing contests
DB query
DATABASE
DB is queried to send the problems from contest and editorials
response
SERVER
Server makes the contest public and allows other users to join it
CLIENT
User wants to create a contest
GET request
SERVER
Express.js directs the user to the create webpage
response
CLIENT
Client allows the user to add problems, test cases, duration etc
POST request
SERVER
Express.js receives the POST request
response
DATABASE
Data is stored in the DB for further use

Figure 3.7: Flow diagram of Contest space

Figure 3.7 illustrates the operational workflow within the contest space, offering insights into the process of creating and accessing contests. When a user initiates the creation of a contest, they do so by submitting a post request along with various parameters defining the contest details. These parameters typically include information such as the contest name, duration, rules, and any additional specifications. Upon receiving this request, the backend system processes the data and creates the contest, ensuring that all parameters are accurately recorded and stored in the database for future reference.

Similarly, when a user seeks to view all available contests, the frontend interface sends a get request to the backend system. In response, the backend diligently retrieves the contest data from the database, cross-referencing the user's permissions and any filtering criteria specified. This data retrieval process is essential for ensuring that users are presented with only the contests that are relevant and accessible to them. Once the appropriate contest data is retrieved, it is seamlessly transmitted back to the frontend, where it is dynamically rendered in the user interface (UI). This rendering process typically involves organizing the contest information into a visually appealing and user-friendly format, such as a list or grid layout.

Through this seamless interaction between the frontend and backend components, users can effortlessly create and explore contests within the system. Moreover, by leveraging robust data management techniques and efficient communication protocols, the system ensures that users are presented with accurate and up-to-date contest information, enhancing their overall experience and engagement with the platform.

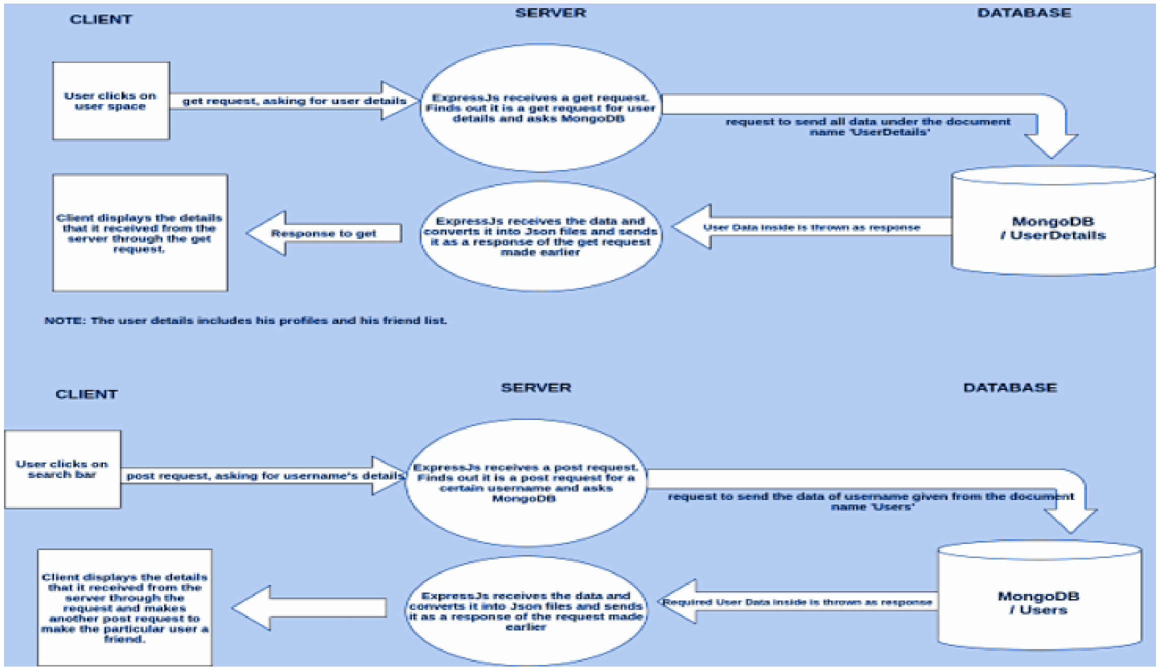

CLIENT
SERVER
DATABASE
User clicks on user space
get request, asking for user details
ExpressJs receives a get request. Finds out it is a get request for user details and asks MongoDB
request to send all data under the document name 'UserDetails'
MongoDB / UserDetails
User Data inside is thrown as response
ExpressJs receives the data and converts it into Json files and sends it as a response of the get request made earlier
Response to get
Client displays the details that it received from the server through the get request.
NOTE: The user details includes his profiles and his friend list.
CLIENT
SERVER
DATABASE
User clicks on search bar
post request, asking for username's details
ExpressJs receives a post request. Finds out it is a post request for a certain username and asks MongoDB
request to send the data of username given from the document name 'Users'
MongoDB / Users
Required User Data inside is thrown as response
ExpressJs receives the data and converts it into Json files and sends it as a response of the request made earlier
Client displays the details that it received from the server through the request and makes another post request to make the particular user a friend.

Figure 3.8: Flow diagram of User space

Figure 3.8 shows the user space diagram that illustrates how users interact within the system. When users enter this space, the FrontEnd asks the BackEnd for their data using an API. The BackEnd then searches for the user's information in the database, including solved problems, friends, name, and email. Once found, this data is sent back to the FrontEnd for display.

In addition to showing information, users can update their profile picture here. It's not just about data; the User Space also encourages interaction. Users can connect with friends, share achievements, and work on problems together. This fosters a sense of community and makes the platform more engaging for users.

Overall, the User Space aims to make things easy for users, providing access to their information and enabling social interactions in a straightforward manner.

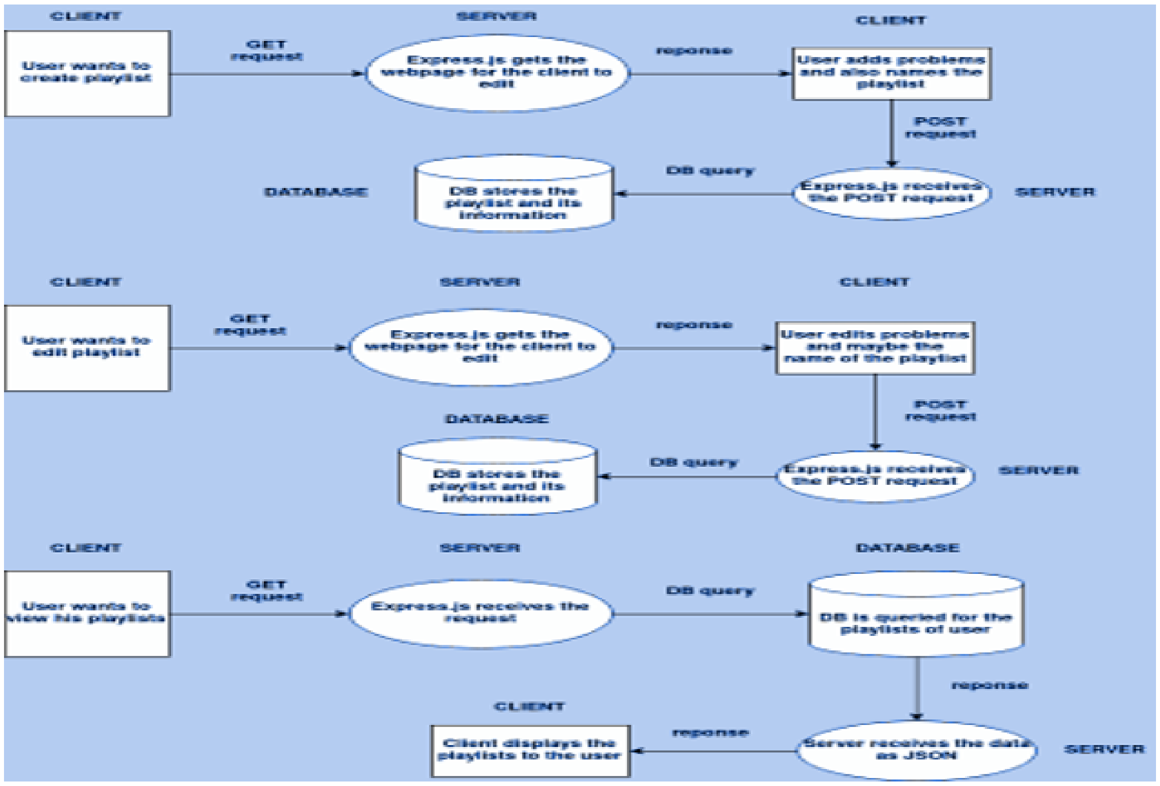

CLIENT
User wants to create playlist
GET request
SERVER
Express.js gets the webpage for the client to edit
reponse
CLIENT
User adds problems and also names the playlist
POST request
Express.js receives the POST request
SERVER
DB query
DATABASE
DB stores the playlist and its information
CLIENT
User wants to edit playlist
GET request
SERVER
Express.js gets the webpage for the client to edit
reponse
CLIENT
User edits problems and maybe the name of the playlist
POST request
Express.js receives the POST request
SERVER
DB query
DATABASE
DB stores the playlist and its information
CLIENT
User wants to view his playlists
GET request
SERVER
Express.js receives the request
DB query
DATABASE
DB is queried for the playlists of user
reponse
Server receives the data as JSON
SERVER
reponse
CLIENT
Client displays the playlists to the user

Figure 3.9: Flow diagram of Playlist space

Figure 3.9 shows the Playlist space where users can practice questions on specific themes or topics by creating customized playlists. Users can add problems to these playlists and choose whether to make them public or private. When a user creates a playlist, the FrontEnd sends the data to the BackEnd through a POST request, where it's stored in the database.

On the default page of the Playlist Space, users see a list of public playlists. This makes it easy to explore and use playlists created by others. These playlists cover a variety of topics, providing users with a diverse range of problems to solve.

This space is not only for learning but also for teaching. Educators can create playlists tailored to their curriculum, helping students focus on specific topics. Overall, the Playlist Space is a useful tool for users to discover, create, and share problem sets, promoting collaborative learning and skill development.

3.2 Architecture of Plagiarism Detection

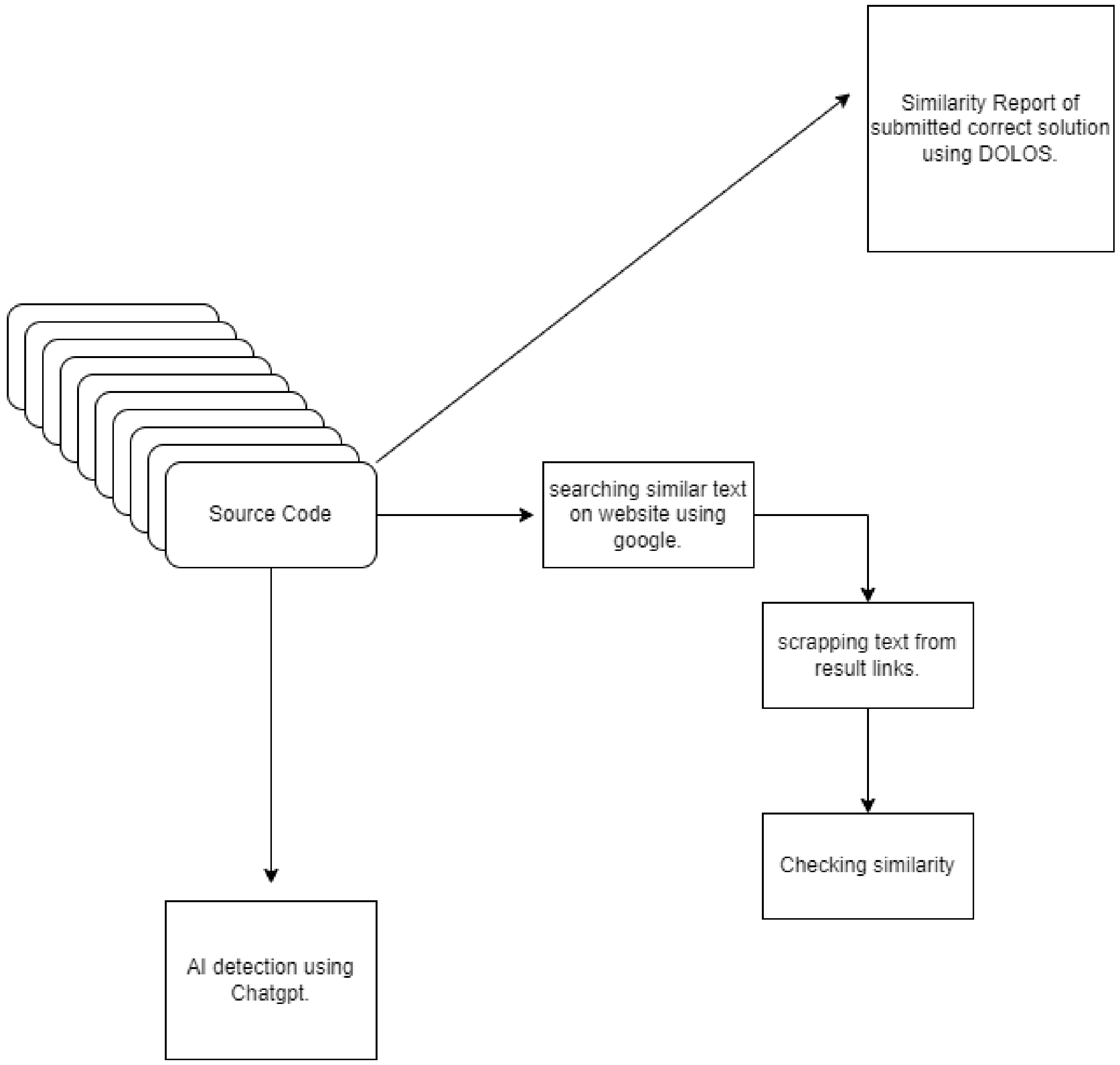

Similarity Report of submitted correct solution using DOLOS.
Source Code
searching similar text on website using google.
scrapping text from result links.
Checking similarity
AI detection using Chatgpt.

Figure 3.10: Plagiarism Checker Module

Figure 3.10 shows the Plagiarism Checker Module that initiates with the retrieval of source code from a designated database. Leveraging Dolos, an advanced plagiarism detection tool, the code undergoes scrutiny to detect any resemblances with existing codebases. Subsequently, utilizing the Google Custom Search API, an extensive search is conducted across the internet to uncover any parallels or matches with the original code. Upon discovering potential matches, a meticulous comparison ensues between the identified snippets and the original code to evaluate their level of similarity. In the final stage, an examination for AI-generated code segments is undertaken to ascertain their presence. ChatGPT, a state-of-the-art AI language model, is employed for this purpose, aiding in the identification of code portions potentially authored by automated programs. Through this rigorous and multi-faceted approach, the integrity and originality of the code are meticulously verified, mitigating the risk of inadvertent plagiarism and ensuring its authenticity.

3.2.1 Dolos

Dolos is a source code plagiarism checker that uses a variety of techniques to identify potential plagiarism. It can parse code from a wide range of programming languages and create unique fingerprints for each code fragment. It then uses these fingerprints to identify code that is similar to other code in its database. Dolos can also calculate three different plagiarism metrics to quantify the degree of similarity between code pairs. These metrics are the overlap metric, the containment metric, and the weighted metric. Finally, Dolos can generate interactive visualizations to help users see the results of their analysis. These visualizations can include a suspicious code map and a similarity matrix.

3.2.2 Plagiarism Check Against Internet Resources

To check the source code plagiarism against internet resources, we will be using google custom search API which will return the url links of the search results pages. After that we are scrapping the text from those website. For Similarity comparision, we are using 'en_core_web_sm' language model. The model is trained on a diverse and extensive corpus of English text data, including web pages, news articles, books, and other sources.The 'en_core_web_sm' model is built on a neural network architecture, typically using convolutional neural networks (CNNs) and recurrent neural networks (RNNs). It consists of multiple layers of neurons interconnected through weighted connections, enabling it to learn complex patterns and relationships in text data. Using this model we will be comparing our program (code) and the website text, and thus returning the similarity score (plagiarism).

3.2.3 Plagiarsim Against ChatGpt

The submitted solution may be generated by AI. We have done it for chatGpt only. To detect AI-generated code we are using Chatgpt, using OPENAI Api, we will be asking how much percentage of code written by ChatGpt.

3.3 System Requirements Specifications

3.3.1 Functional Requirements

(1) Scraping problems from different websites like Codeforces, HackerEarth, etc.

(2) Codeforces API for fetching problem lists, details, tags, and user details.

(3) Also, code compilation APIs for compiling the code.

(4) Integration of IDE for writing code.

3.3.2 Non-Functional Requirements

(1) User authentication and authorization.

(2) The React frontend code can be used anywhere on the website which provides reusability.

(3) MongoDB is the database used. Data is protected with pre-configured security features for authentication, authorization, encryption, and more.

CHAPTER 4

RESULTS AND ANALYSIS

Successfully designed the architecture of the system, MVC diagram. Created the flow diagram of the system. Designed the flow diagram of the modules such as user space ,contest space and coding space. Developed the activity diagram of the system.Ready with the functional and non functional requirements along with the system and software requirements. Evaluated the different modules which will be a part of the system along with their flow diagrams. On the basis of the different diagrams we analyzed that the system will be compatible and will have low latency and can be easily vertical scalable as MongoDB is being used as a database. The screenshots and results obtained from our Coding Platform are as follows.

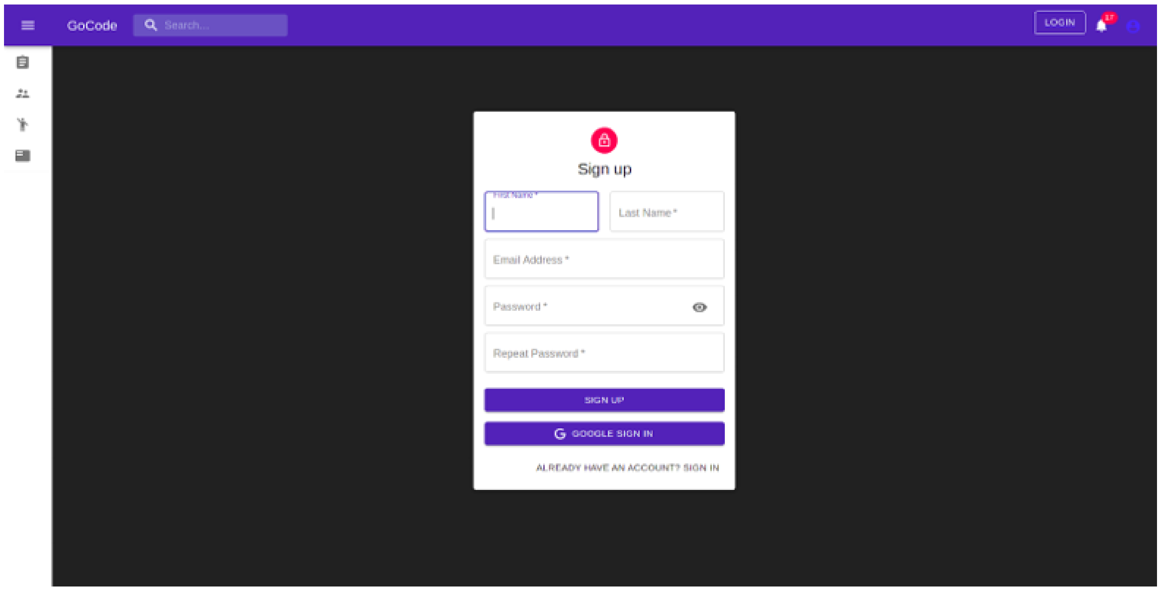


Figure 4.1: Sign Up page

Figure 4.1 shows the sign-up page for our platform provides a streamlined and user-friendly experience, prompting users to input their first name, last name, email address, and password to create a new account. Additionally, users have the option to sign up using their Google account for added convenience. Clear instructions guide users through the process, ensuring that they understand what information is required. For existing users, a sign-in link is conveniently located at the bottom of the page for quick access to their accounts. This thoughtful design enhances the user experience by simplifying the registration and sign-in processes, ultimately contributing to a positive user journey on the platform.

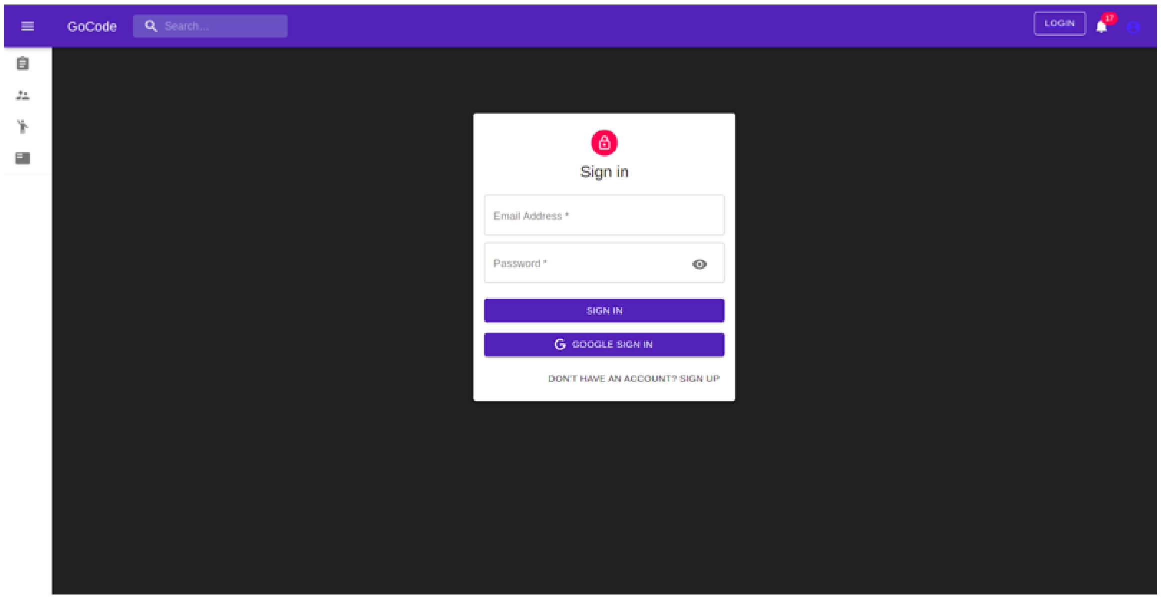
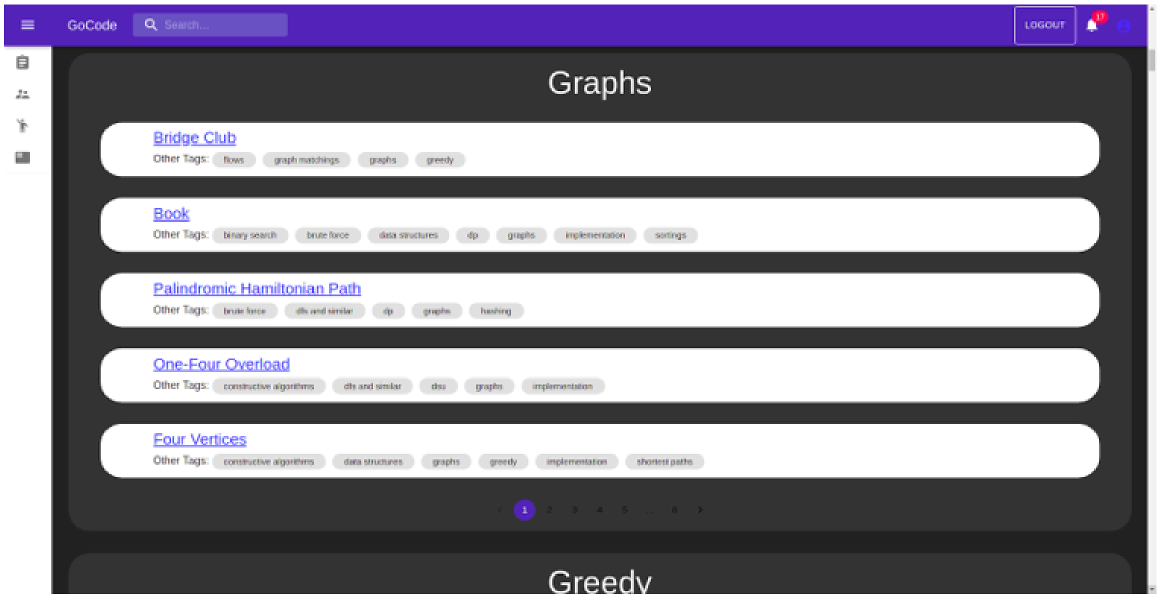


Figure 4.2: Sign in page

Figure 4.2 shows the login screen for our platform offers a seamless and convenient experience for users to access their accounts. In addition to traditional username and password fields, users have the option to sign in using their Google account, eliminating the need to remember and enter separate login credentials. This integration with Google authentication streamlines the log-in process, enhancing user accessibility and reducing friction. By providing multiple sign-in options, the platform caters to diverse user preferences and simplifies the authentication process, ultimately contributing to a smoother and more efficient user experience.



Graphs

Bridge Club
Other Tags: flows graph matchings graphs greedy

Book
Other Tags: binary search brute force data structures dp graphs implementation sortings

Palindromic Hamiltonian Path
Other Tags: brute force dfs and similar dp graphs hashing

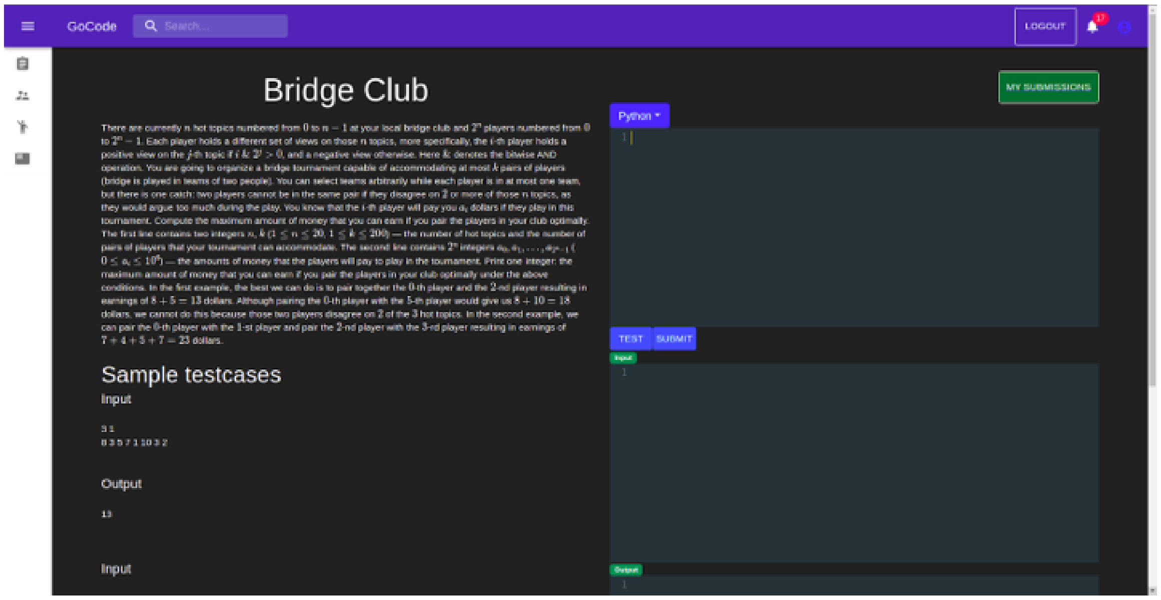
One-Four Overload
Other Tags: constructive algorithms dfs and similar dsu graphs implementation

Four Vertices
Other Tags: constructive algorithms data structures graphs greedy implementation shortest paths

Greedy

Figure 4.3: Problems Space

Figure 4.3 shows the screenshot that displays a section dedicated to "Graphs" on our platform, showcasing various problems categorized under this topic. Each problem is accompanied by relevant problem tags, providing users with insights into the specific concepts or algorithms covered. This organization allows users to easily navigate and explore a diverse range of graph-related problems, fostering a focused learning environment. By categorizing problems based on their topic and providing associated tags, the platform facilitates targeted practice and skill development for users interested in mastering graph algorithms. Additionally, the inclusion of problem tags enhances search-ability and enables users to efficiently find problems related to their areas of interest or study. Overall, the categorization of graph problems and the use of problem tags optimize the user experience, promoting effective learning and problem-solving on the platform.



Bridge Club

There are currently n hot topics numbered from 0 to $n-1$ at your local bridge club and 2^n players numbered from 0 to 2^n-1. Each player holds a different set of views on those n topics, more specifically, the i-th player holds a positive view on the j-th topic if $i \ \& \ 2^j > 0$, and a negative view otherwise. Here $\&$ denotes the bitwise AND operation. You are going to organize a bridge tournament capable of accommodating at most k pairs of players (bridge is played in teams of two people). You can select teams arbitrarily while each player is in at most one team, but there is one catch: two players cannot be in the same pair if they disagree on 2 or more of those n topics, as they would argue too much during the play. You know that the i-th player will pay you a_i dollars if they play in this tournament. Compute the maximum amount of money that you can earn if you pair the players in your club optimally. The first line contains two integers n, k ($1 \leq n \leq 20, 1 \leq k \leq 200$) — the number of hot topics and the number of pairs of players that your tournament can accommodate. The second line contains 2^n integers $a_0, a_1, \ldots, a_{2^n-1}$ ($0 \leq a_i \leq 10^6$) — the amounts of money that the players will pay to play in the tournament. Print one integer: the maximum amount of money that you can earn if you pair the players in your club optimally under the above conditions. In the first example, the best we can do is to pair together the 0-th player and the 2-nd player resulting in earnings of $8+5=13$ dollars. Although pairing the 0-th player with the 5-th player would give us $8+10=18$ dollars, we cannot do this because those two players disagree on 2 of the 3 hot topics. In the second example, we can pair the 0-th player with the 1-st player and pair the 2-nd player with the 3-rd player resulting in earnings of $7+4+5+7=23$ dollars.

Sample testcases

Input

```
3 1
8 3 5 7 1 10 3 2
```

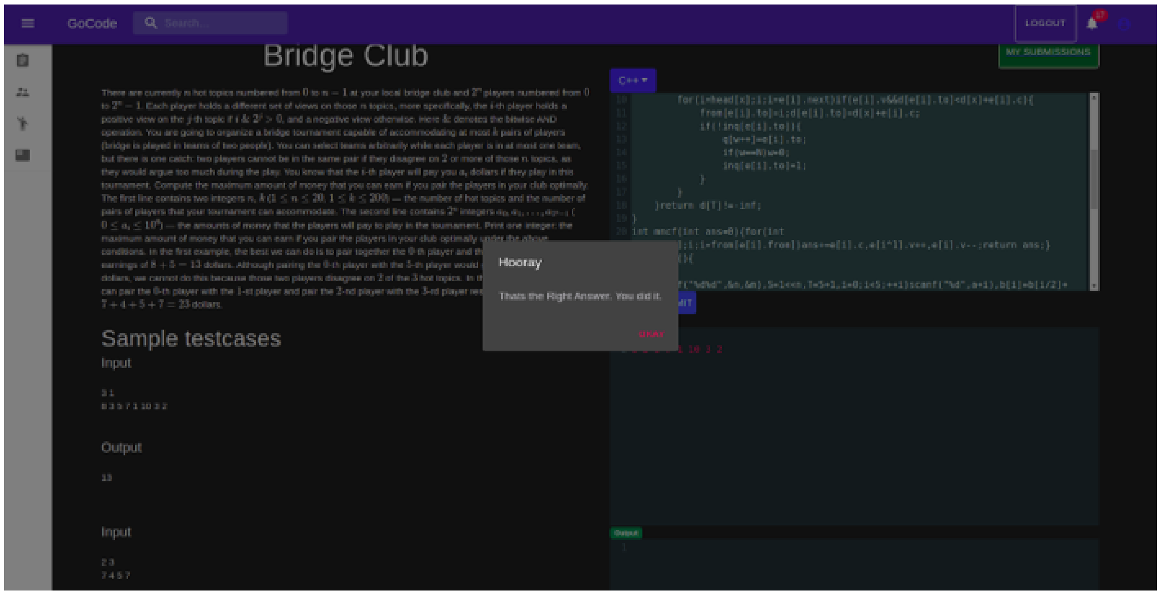
Output

```
13
```

Input

Figure 4.4: Problem Statement

Figure 4.4 shows the platform that presents users with a comprehensive coding environment which includes both the problem statement and an integrated code editor for compiling and executing the program (code). This setup allows users to view the problem statement alongside the code editor, facilitating a focused approach to problem-solving. Furthermore, users are provided with the flexibility to choose from four different programming languages for editing their code, catering to diverse language preferences and skill sets. This multi-language support enhances accessibility and accommodates users with varying levels of proficiency in different programming languages. By offering a seamless integration of problem statements and a versatile code editor supporting multiple languages, the platform empowers users to effectively tackle coding challenges and practice problem-solving skills in their preferred programming language.



Bridge Club

There are currently n hot topics numbered from 0 to $n-1$ at your local bridge club and 2^n players numbered from 0 to 2^n-1. Each player holds a different set of views on those n topics, more specifically, the i-th player holds a positive view on the j-th topic if $i \& 2^j > 0$, and a negative view otherwise. Here $\&$ denotes the bitwise AND operation. You are going to organize a bridge tournament capable of accommodating at most k pairs of players (bridge is played in teams of two people). You can select teams arbitrarily while each player is in at most one team, but there is one catch: two players cannot be in the same pair if they disagree on 2 or more of those n topics, as they would argue too much during the play. You know that the i-th player will pay you a_i dollars if they play in this tournament. Compute the maximum amount of money that you can earn if you pair the players in your club optimally. The first line contains two integers n, k ($1 \le n \le 20$, $1 \le k \le 200$) — the number of hot topics and the number of pairs of players that your tournament can accommodate. The second line contains 2^n integers $a_0, a_1, \ldots, a_{2^n-1}$ ($0 \le a_i \le 10^6$) — the amounts of money that the players will pay to play in the tournament. Print one integer: the maximum amount of money that you can earn if you pair the players in your club optimally under the above conditions. In the first example, the best we can do is to pair together the 0-th player and th[illegible] earnings of $8+5=13$ dollars. Although pairing the 0-th player with the 5-th player would [illegible] dollars, we cannot do this because those two players disagree on 2 of the 3 hot topics. In th[illegible] can pair the 0-th player with the 1-st player and pair the 2-nd player with the 3-rd player res[illegible] $7+4+5+7=23$ dollars.

Sample testcases

Input

```
3 1
8 3 5 7 1 10 3 2
```

Output

```
13
```

Input

```
2 3
7 4 5 7
```

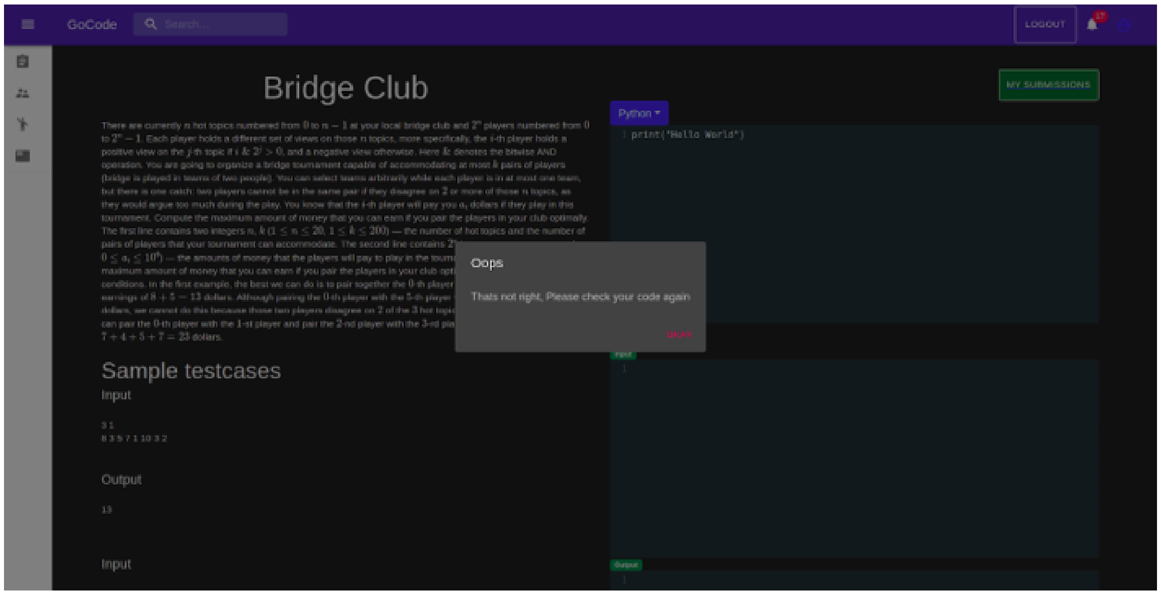
```
        for(i=head[x];i;i=e[i].next)if(e[i].v&&d[e[i].to]<d[x]+e[i].c){
            from[e[i].to]=i;d[e[i].to]=d[x]+e[i].c;
            if(!inq[e[i].to]){
                q[w++]=e[i].to;
                if(w==N)w=0;
                inq[e[i].to]=1;
            }
        }
    }return d[T]!=-inf;
}
int mncf(int ans=0){for(int
[illegible]];i;i=from[e[i].from])ans+=e[i].c,e[i^1].v++,e[i].v--;return ans;}
[illegible](){
[illegible]f("%d%d",&n,&m),S=1<<n,T=S+1,i=0;i<S;++i)scanf("%d",a+i),b[i]=b[i/2]+
```

Thats the Right Answer. You did it.

```
[illegible] 1 10 3 2
```

Figure 4.5: Successful Submission

Figure 4.5 shows the successful submission of a solution to a problem on our platform, users are greeted with a popup window confirming the successful submission. This notification serves as instant feedback, reassuring users that their submission has been received and processed. Additionally, the popup window may include relevant details such as the submission status, time of submission, and any additional information pertinent to the submission process. By providing immediate feedback in the form of a popup window, the platform enhances the user experience, promoting transparency and accountability throughout the problem-solving journey. This real-time feedback mechanism encourages users to stay engaged and motivated, fostering a positive learning and coding experience on the platform.



Bridge Club

There are currently n hot topics numbered from 0 to $n-1$ at your local bridge club and 2^n players numbered from 0 to 2^n-1. Each player holds a different set of views on those n topics, more specifically, the i-th player holds a positive view on the j-th topic if $i \,\&\, 2^j > 0$, and a negative view otherwise. Here $\&$ denotes the bitwise AND operation. You are going to organize a bridge tournament capable of accommodating at most k pairs of players (bridge is played in teams of two people). You can select teams arbitrarily while each player is in at most one team, but there is one catch: two players cannot be in the same pair if they disagree on 2 or more of those n topics, as they would argue too much during the play. You know that the i-th player will pay you a_i dollars if they play in this tournament. Compute the maximum amount of money that you can earn if you pair the players in your club optimally. The first line contains two integers n, k ($1 \le n \le 20$, $1 \le k \le 200$) — the number of hot topics and the number of pairs of players that your tournament can accommodate. The second line contains 2^n [illegible]
$0 \le a_i \le 10^6$) — the amounts of money that the players will pay to play in the tourn[illegible]
maximum amount of money that you can earn if you pair the players in your club opt[illegible]
conditions. In the first example, the best we can do is to pair together the 0-th player [illegible]
earnings of $8+5=13$ dollars. Although pairing the 0-th player with the 5-th player [illegible]
dollars, we cannot do this because those two players disagree on 2 of the 3 hot topi[illegible]
can pair the 0-th player with the 1-st player and pair the 2-nd player with the 3-rd pla[illegible]
$7+4+5+7=23$ dollars.

Sample testcases

Input

```
3 1
8 3 5 7 1 10 3 2
```

Output

```
13
```

Input

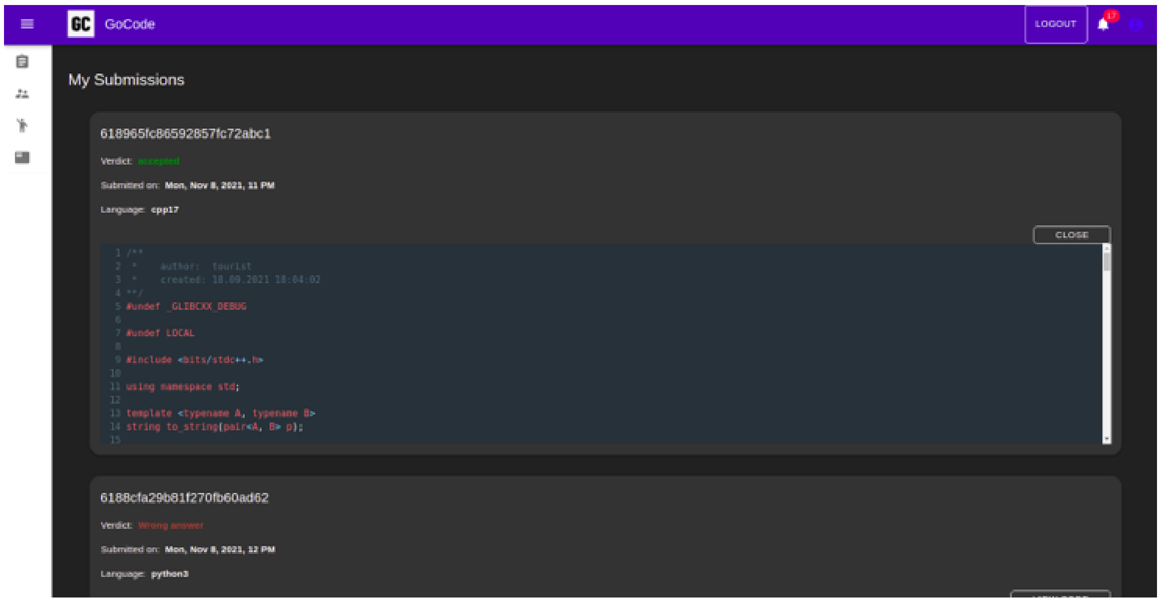
```
print("Hello World")
```

Oops

Thats not right, Please check your code again

Figure 4.6: Wrong Submission

Figure 4.6 shows the wrong submission When a user makes an incorrect submission on the platform, a popup window appears to notify them of the error. This popup window serves as instant feedback, informing the user that their submission was unsuccessful. Additionally, the platform records the incorrect submission in the submissions module for the specific problem. This feature allows users to track their submission history and review their mistakes, facilitating a learning-oriented approach to problem-solving. By providing real-time feedback and recording incorrect submissions, the platform encourages users to analyze their errors, refine their strategies, and ultimately improve their problem-solving skills over time.



My Submissions

618965fc86592857fc72abc1

Verdict: accepted

Submitted on: Mon, Nov 8, 2021, 11 PM

Language: cpp17

```
/**
 *    author:  tourist
 *    created: 18.09.2021 18:04:02
**/
#undef _GLIBCXX_DEBUG

#undef LOCAL

#include <bits/stdc++.h>

using namespace std;

template <typename A, typename B>
string to_string(pair<A, B> p);

```

6188cfa29b81f270fb60ad62

Verdict: Wrong answer

Submitted on: Mon, Nov 8, 2021, 12 PM

Language: python3

Figure 4.7: All Submissions

Figure 4.7 displays a comprehensive overview of all submissions made for a specific problem, offering users insights into various details such as the time of submission and other relevant information. This feature allows users to track their progress and compare their solutions with those of other users, fostering a collaborative learning environment. By providing transparency and visibility into submission details, including timestamps and possibly additional metadata, the platform promotes accountability and encourages users to learn from their peers' approaches. Additionally, this feature enables users to analyze their submission history, identify patterns, and iterate on their solutions, ultimately enhancing their problem-solving skills and contributing to a richer learning experience on the platform.

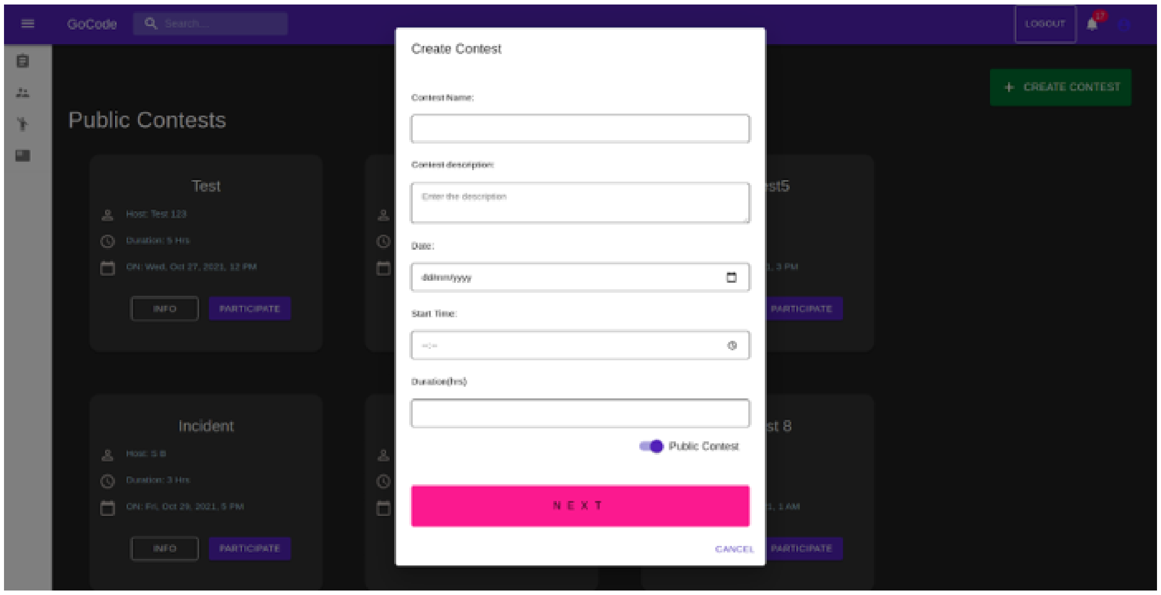
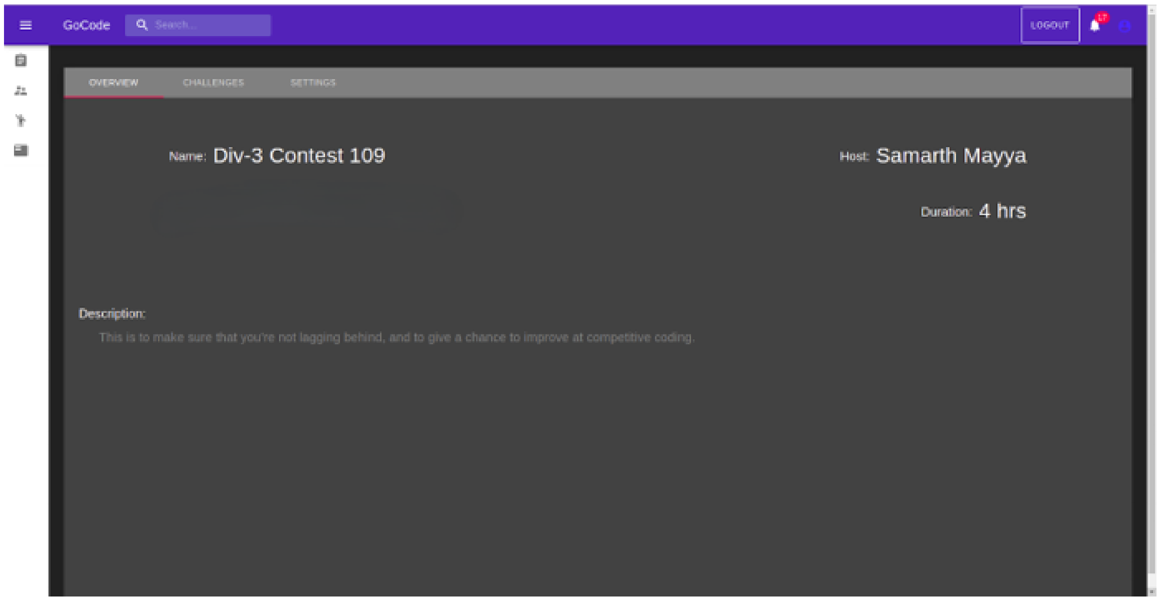


Figure 4.8: Create Contest

Figure 4.8 shows the platform that features a popup window that enables users to create contests with various parameters, offering a customizable and user-friendly experience. Within this window, users can specify parameters such as the contest name, duration, start time, and other relevant settings. Additionally, users may have the option to define contest rules, select problem sets, and invite participants. This intuitive interface streamlines the contest creation process, empowering users to organize and host contests tailored to their preferences and objectives. By providing a popup window dedicated to contest creation, the platform facilitates seamless navigation and ensures that users can efficiently set up and manage contests with ease, ultimately fostering engagement and participation within the coding community.



Name: Div-3 Contest 109

Host: Samarth Mayya

[redacted]

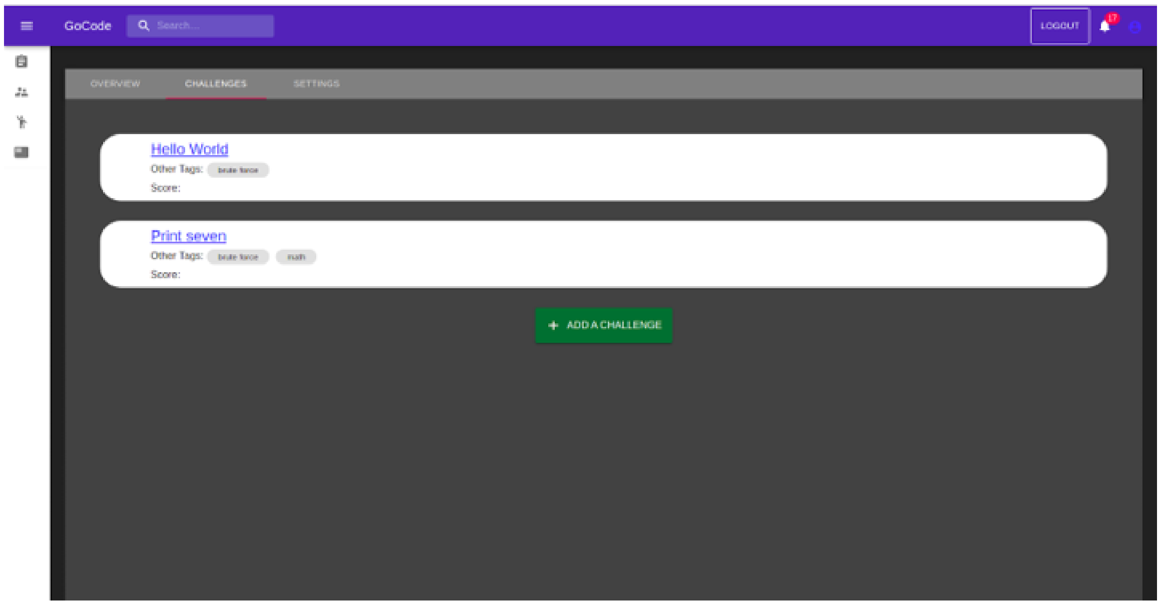
Duration: 4 hrs

Description:

This is to make sure that you're not lagging behind, and to give a chance to improve at competitive coding.

Figure 4.9: Contest Overview

Figure 4.9 shows the overview for the contest created by the user, consisting of various fields such as name of the contest, owner of the contest, duration etc.



Hello World
Other Tags: brute force
Score:

Print seven
Other Tags: brute force math
Score:

Figure 4.10: Created Problems

Figure 4.10 shows all the problems under the create contest module with various problem tags along with it.

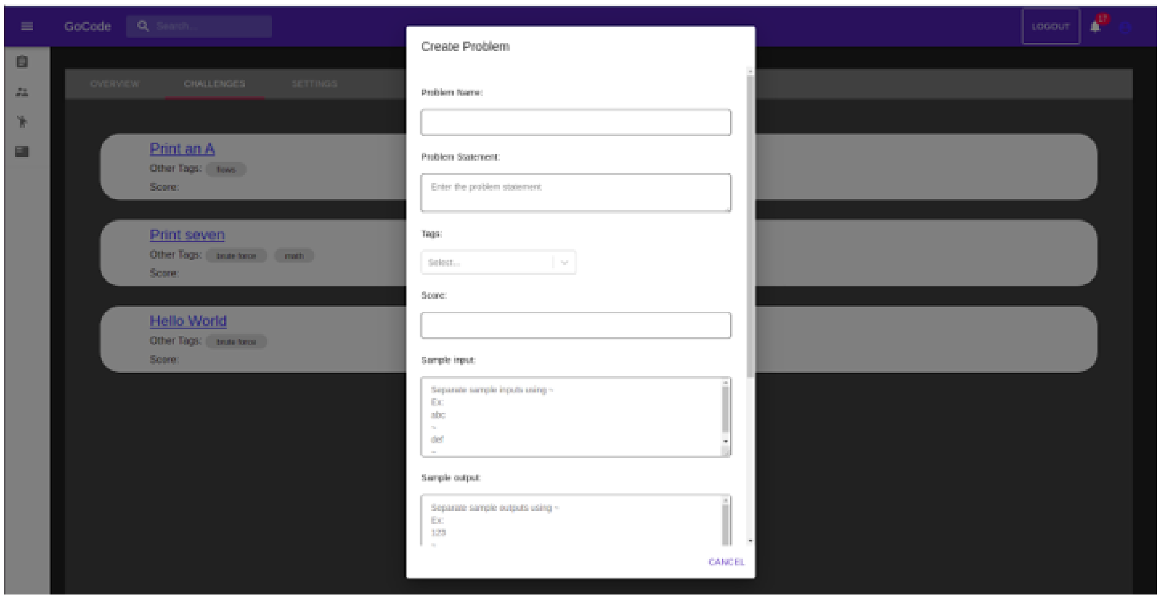


Figure 4.11: Create Problems

Figure 4.11 shows the created contest space wherein the users are presented with a set of fields to create a new problem. These fields typically include essential details such as the problem title, description, input/output specifications, constraints, and sample test cases. Additionally, users may have the option to specify tags or categories to classify the problem, making it easier for participants to search and identify relevant challenges. By providing these fields, the platform enables contest organizers to define clear and comprehensive problem statements that adhere to standard problem-solving conventions. This streamlined process ensures that problems are well-defined, facilitating a smooth and engaging contest experience for participants.

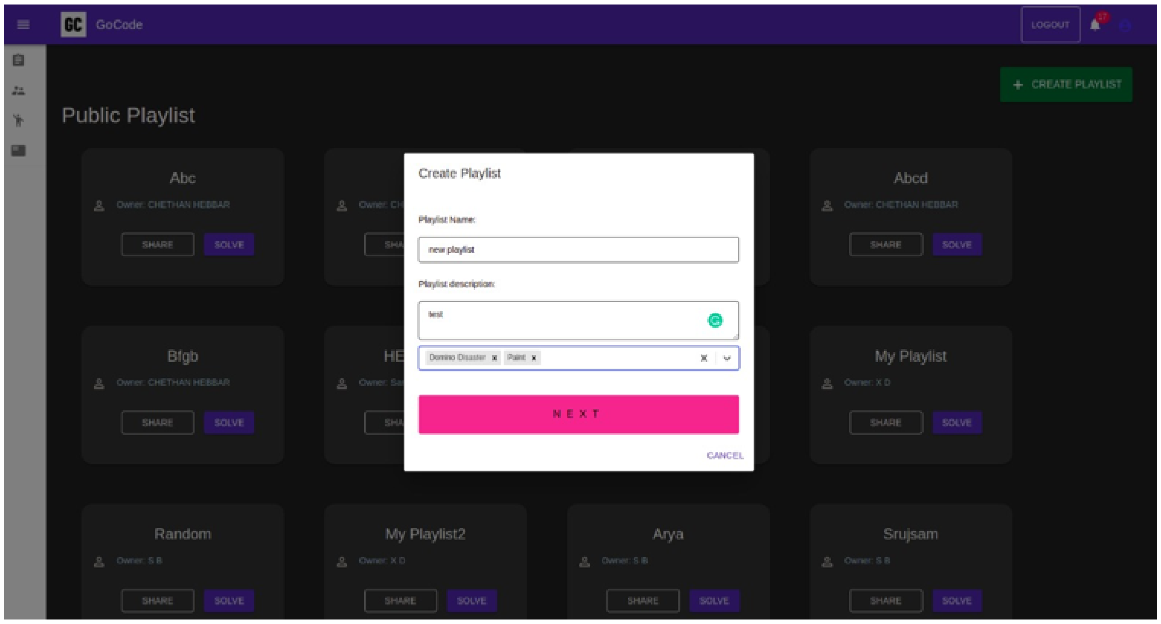
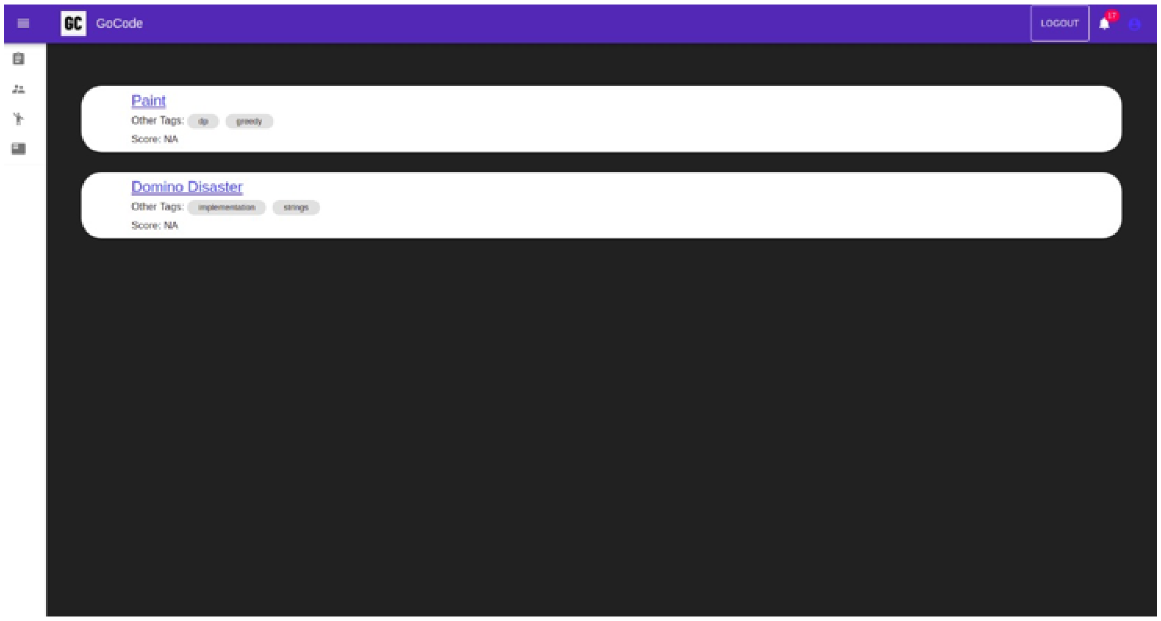


Figure 4.12: Create Playlist

Figure 4.12 shows the platform that offers users a popup window to create playlists consisting of a diverse variety of problems. Within this popup, users can specify the playlist name and add problems from different categories, topics, or difficulty levels. Additionally, users may have the option to customize the playlist by arranging the order of problems or adding tags for better organization. This feature allows users to curate personalized collections of problems tailored to their learning goals or preferences. By providing a convenient interface for playlist creation, the platform empowers users to create structured study materials or practice sets, enhancing their learning experience and facilitating focused skill development.



Paint
Other Tags: dp greedy
Score: NA

Domino Disaster
Other Tags: implementation strings
Score: NA

Figure 4.13: Playlist Problems

Figure 4.13 shows the platform's Playlists module that displays all problems organized within playlists, accompanied by various problem tags. This feature allows users to access curated collections of problems tailored to specific topics, categories, or learning objectives. Each problem is tagged with relevant keywords or labels, providing users with insights into the problem's content, difficulty level, or related concepts. By offering organized access to problems and incorporating problem tags, the platform enhances user navigation and facilitates targeted practice and skill development. This comprehensive approach to problem organization promotes the efficient learning and problem-solving strategies, thus contributing to a more enriching user experience on the coding platform.

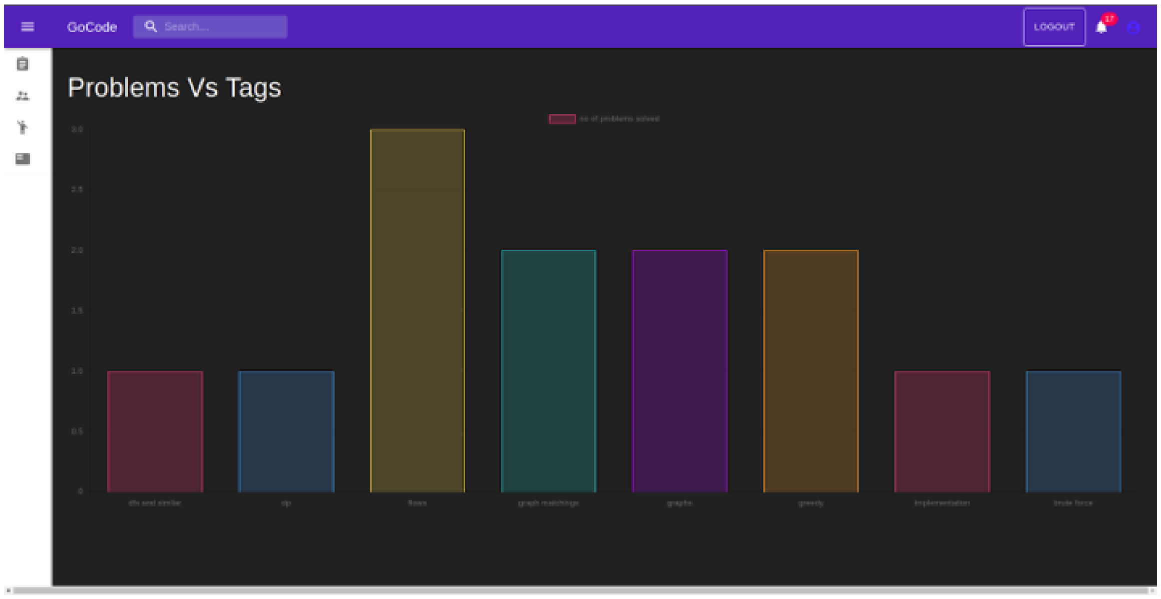
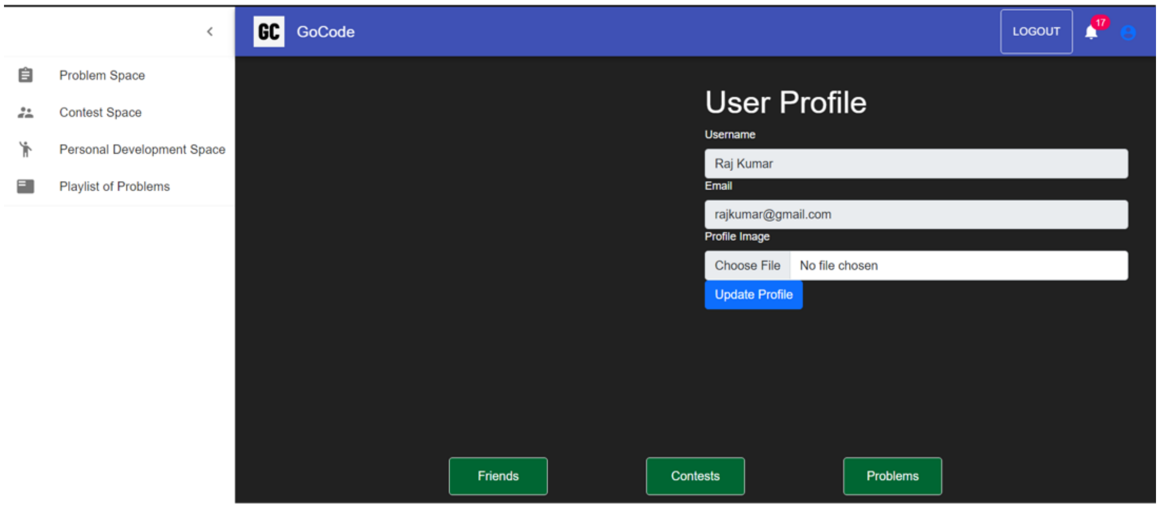


Figure 4.14: Problems v/s Tags

Figure 4.14 shows the platform that displays the number of problems solved under various categories or tags, providing users with insights into their progress and proficiency across different problem types. This feature enables users to track their performance and identify areas of strength or areas needing improvement. By presenting statistics on problems solved by category or tag, the platform encourages users to engage with a diverse range of problems and fosters a well-rounded skill development approach. Additionally, this information facilitates the goal setting and helps the users focus their efforts on specific areas of interest or challenge, ultimately contributing to a more targeted and effective learning experience on the coding platform.



User Profile

Username

Raj Kumar

Email

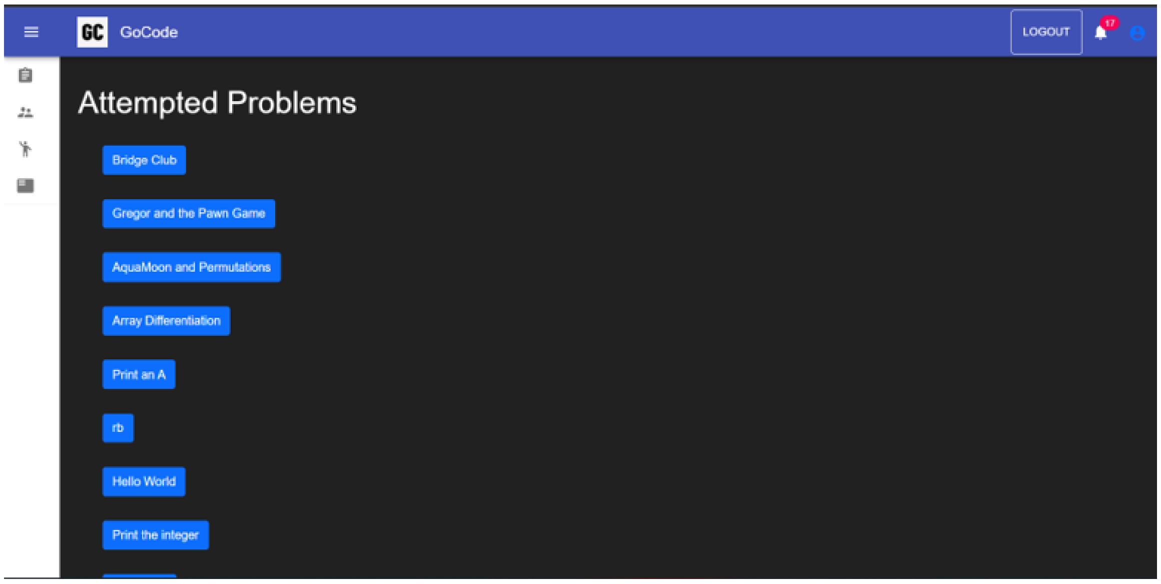
rajkumar@gmail.com

Profile Image

Choose File No file chosen

Update Profile

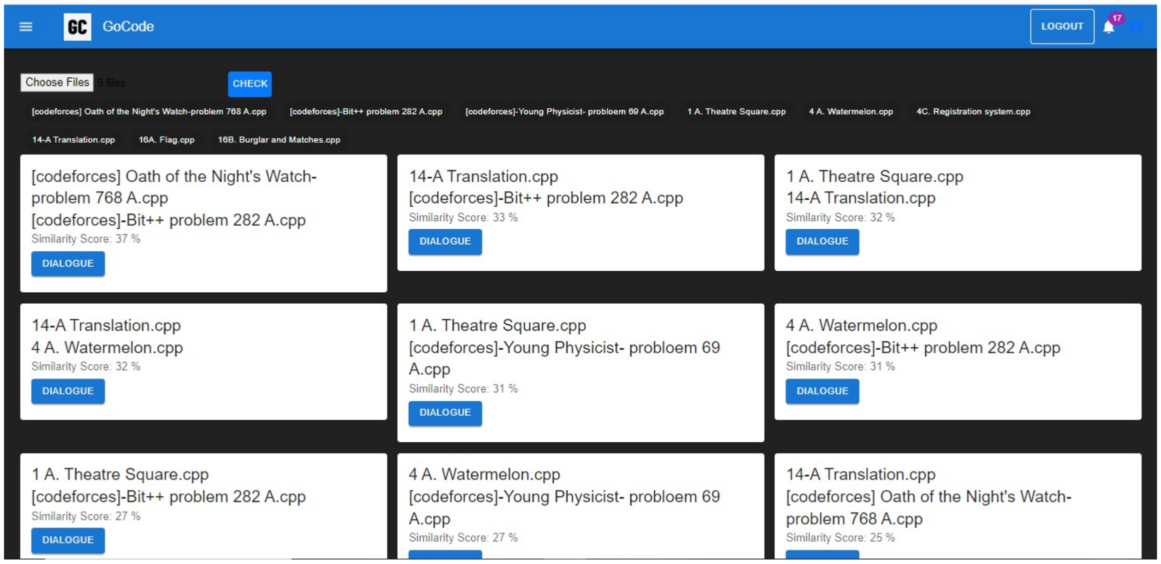
Figure 4.15: User Profile
Figure 4.15 shows the user profile page consisting of username, email, password field. The password of the user may be updated if required.



Attempted Problems

Bridge Club
Gregor and the Pawn Game
AquaMoon and Permutations
Array Differentiation
Print an A
rb
Hello World
Print the integer

Figure 4.16: Attempted Problems
Figure 4.16 shows all the total attempted problems on the platform by the user.



[codeforces] Oath of the Night's Watch-problem 768 A.cpp
[codeforces]-Bit++ problem 282 A.cpp
[codeforces]-Young Physicist- probloem 69 A.cpp
1 A. Theatre Square.cpp
4 A. Watermelon.cpp
4C. Registration system.cpp
14-A Translation.cpp
16A. Flag.cpp
16B. Burglar and Matches.cpp

[codeforces] Oath of the Night's Watch-problem 768 A.cpp
[codeforces]-Bit++ problem 282 A.cpp
Similarity Score: 37 %
DIALOGUE

14-A Translation.cpp
[codeforces]-Bit++ problem 282 A.cpp
Similarity Score: 33 %
DIALOGUE

1 A. Theatre Square.cpp
14-A Translation.cpp
Similarity Score: 32 %
DIALOGUE

14-A Translation.cpp
4 A. Watermelon.cpp
Similarity Score: 32 %
DIALOGUE

1 A. Theatre Square.cpp
[codeforces]-Young Physicist- probloem 69 A.cpp
Similarity Score: 31 %
DIALOGUE

4 A. Watermelon.cpp
[codeforces]-Bit++ problem 282 A.cpp
Similarity Score: 31 %
DIALOGUE

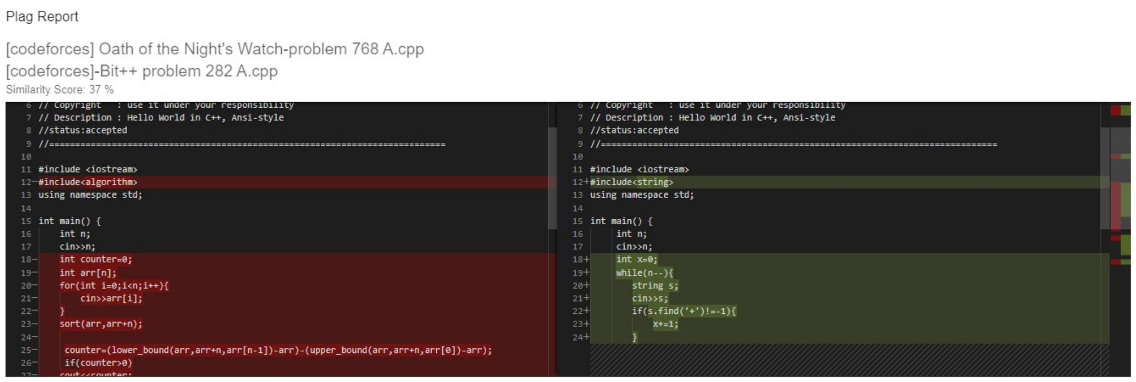
1 A. Theatre Square.cpp
[codeforces]-Bit++ problem 282 A.cpp
Similarity Score: 27 %
DIALOGUE

4 A. Watermelon.cpp
[codeforces]-Young Physicist- probloem 69 A.cpp
Similarity Score: 27 %

14-A Translation.cpp
[codeforces] Oath of the Night's Watch-problem 768 A.cpp
Similarity Score: 25 %

Figure 4.17: Similarity Score

Figure 4.17 shows the platform that features a page demonstrating the DOLOS framework, showcasing the similarity score among chosen source code files. This functionality allows users to assess the similarity between different code snippets or files, providing insights into code reuse, duplication, or plagiarism detection. By presenting similarity scores, the platform facilitates code analysis and promotes best practices in code quality and integrity. This feature is particularly valuable for developers collaborating on projects or reviewing code submissions, helping ensure consistency and adherence to coding standards. Overall, the similarity score functionality enhances the platform's capabilities for code analysis and fosters a culture of transparency and accountability in software development practices..



Plag Report

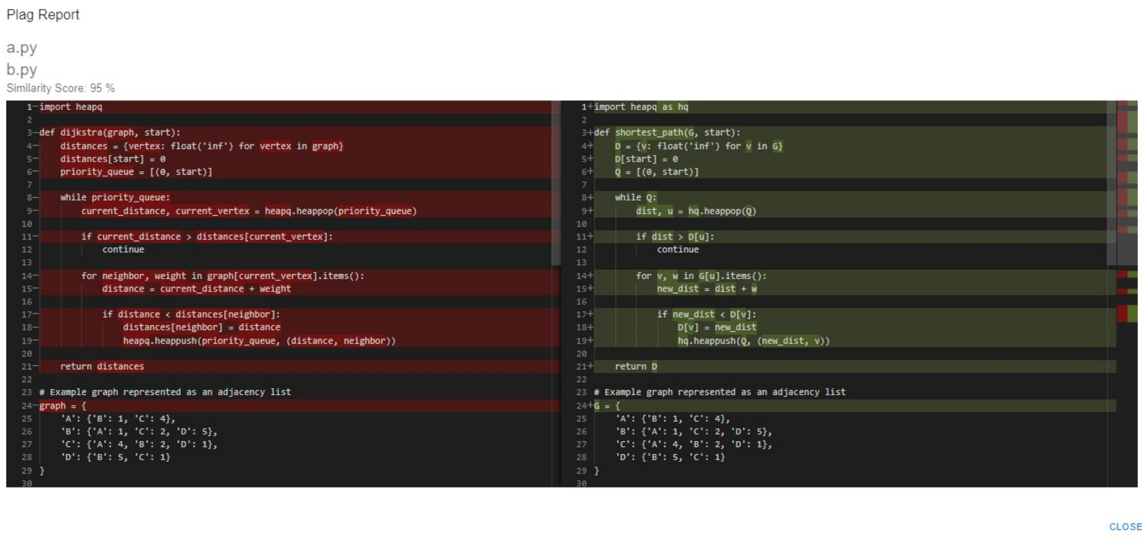
[codeforces] Oath of the Night's Watch-problem 768 A.cpp
[codeforces]-Bit++ problem 282 A.cpp
Similarity Score: 37 %

```
// Copyright   : use it under your responsibility
// Description : Hello World in C++, Ansi-style
//status:accepted
//==========================================================================

#include <iostream>
-#include<algorithm>
using namespace std;

int main() {
    int n;
    cin>>n;
-   int counter=0;
-   int arr[n];
-   for(int i=0;i<n;i++){
-       cin>>arr[i];
-   }
-   sort(arr,arr+n);
-
-    counter=(lower_bound(arr,arr+n,arr[n-1])-arr)-(upper_bound(arr,arr+n,arr[0])-arr);
-    if(counter>0)
-   cout<<counter;
```

```
// Copyright   : use it under your responsibility
// Description : Hello World in C++, Ansi-style
//status:accepted
//==========================================================================

#include <iostream>
+#include<string>
using namespace std;

int main() {
    int n;
    cin>>n;
+   int x=0;
+   while(n--){
+      string s;
+      cin>>s;
+      if(s.find('+')!=-1){
+          x+=1;
+      }
```

Figure 4.18: Plagiarism Report

Figure 4.18 shows the platform that offers a visual representation of the changes made in the second code by modifying variables, providing users with a clear and intuitive way to track code modifications. This feature visually highlights differences between the original and modified code, such as variable name changes, additions, or deletions, enabling users to quickly identify and understand the impact of their edits. By presenting these changes visually, the platform enhances user comprehension and facilitates code review and collaboration. This visual representation fosters a more efficient and effective coding process, thus promoting collaboration and ensures the code quality on the coding platform.



Plag Report

a.py

b.py

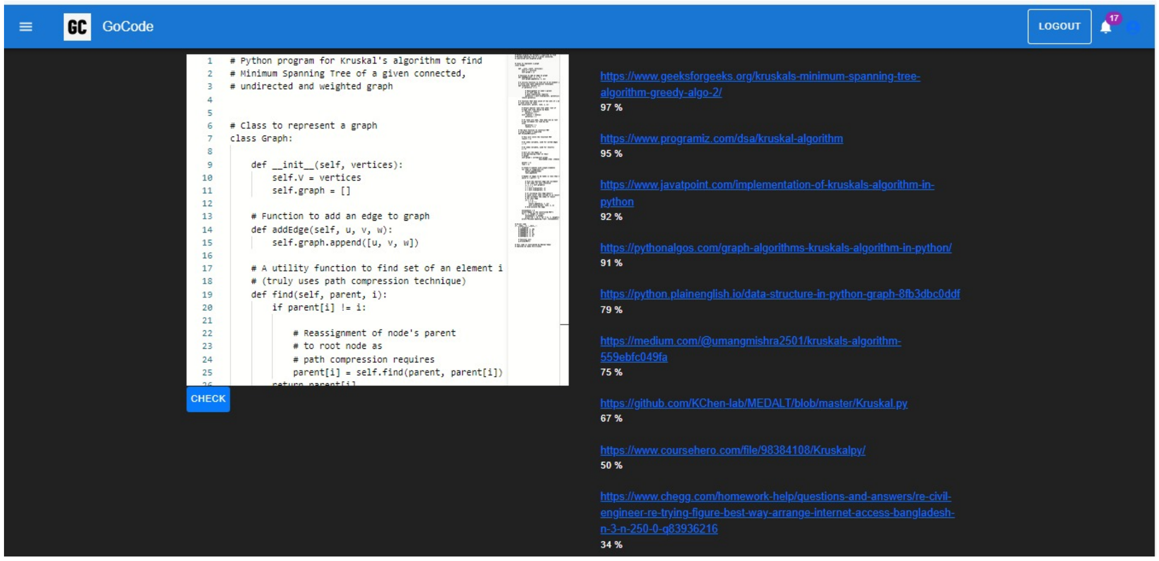
Similarity Score: 95 %

```
import heapq

def dijkstra(graph, start):
    distances = {vertex: float('inf') for vertex in graph}
    distances[start] = 0
    priority_queue = [(0, start)]

    while priority_queue:
        current_distance, current_vertex = heapq.heappop(priority_queue)

        if current_distance > distances[current_vertex]:
            continue

        for neighbor, weight in graph[current_vertex].items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance, neighbor))

    return distances

# Example graph represented as an adjacency list
graph = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5},
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1}
}
```

```
import heapq as hq

def shortest_path(G, start):
    D = {v: float('inf') for v in G}
    D[start] = 0
    Q = [(0, start)]

    while Q:
        dist, u = hq.heappop(Q)

        if dist > D[u]:
            continue

        for v, w in G[u].items():
            new_dist = dist + w

            if new_dist < D[v]:
                D[v] = new_dist
                hq.heappush(Q, (new_dist, v))

    return D

# Example graph represented as an adjacency list
G = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5},
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1}
}
```

Figure 4.19: Plagiarism Report

Figure 4.19 shows the platform that generates a plagiarism report for changed variable names, highlighting similarities and spotting potential instances of code reuse or plagiarism. This report offers users insights into the extent of similarity between code segments, helping to identify any potential violations of academic integrity or intellectual property rights. By flagging similarities in variable names, the platform assists users in ensuring the originality and integrity of their code submissions. This plagiarism detection feature reinforces ethical coding practices and promotes a culture of academic honesty and integrity within the coding community.



```
# Python program for Kruskal's algorithm to find
# Minimum Spanning Tree of a given connected,
# undirected and weighted graph


# Class to represent a graph
class Graph:

    def __init__(self, vertices):
        self.V = vertices
        self.graph = []

    # Function to add an edge to graph
    def addEdge(self, u, v, w):
        self.graph.append([u, v, w])

    # A utility function to find set of an element i
    # (truly uses path compression technique)
    def find(self, parent, i):
        if parent[i] != i:

            # Reassignment of node's parent
            # to root node as
            # path compression requires
            parent[i] = self.find(parent, parent[i])
```

https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/
97 %

https://www.programiz.com/dsa/kruskal-algorithm
95 %

https://www.javatpoint.com/implementation-of-kruskals-algorithm-in-python
92 %

https://pythonalgos.com/graph-algorithms-kruskals-algorithm-in-python/
91 %

https://python.plainenglish.io/data-structure-in-python-graph-8fb3dbc0ddf
79 %

https://medium.com/@umangmishra2501/kruskals-algorithm-559ebfc049fa
75 %

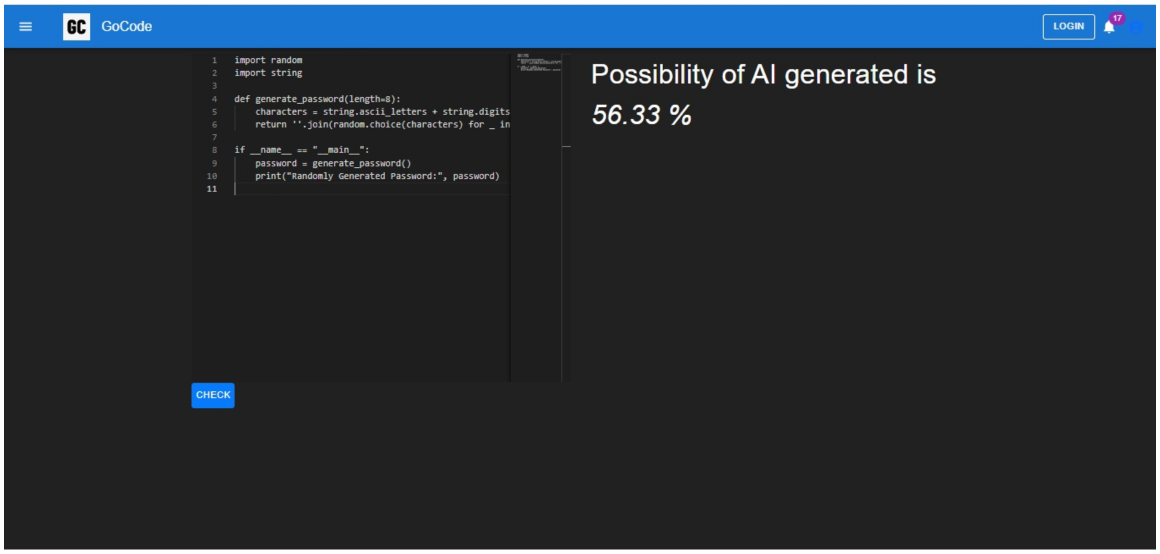
https://github.com/KChen-lab/MEDALT/blob/master/Kruskal.py
67 %

https://www.coursehero.com/file/98384108/Kruskalpy/
50 %

https://www.chegg.com/homework-help/questions-and-answers/re-civil-engineer-re-trying-figure-best-way-arrange-internet-access-bangladesh-n-3-n-250-0-q83936216
34 %

Figure 4.20: Source Code similarity across Web

Figure 4.20 shows the platform that provides a similarity score for the source code compared with solutions from different websites such as Codeforces, GeeksforGeeks, GitHub, and others. This feature enables users to assess the uniqueness of their code and identify any similarities with existing solutions available on external platforms. By presenting the similarity score, the platform aids users in ensuring the originality of their code submissions and avoiding unintentional plagiarism. This functionality promotes ethical coding practices and encourages users to develop solutions independently, fostering a culture of integrity and authenticity within the coding community. Additionally, it helps users gauge the effectiveness of their problem-solving approaches and gain insights into common coding patterns across different platforms.



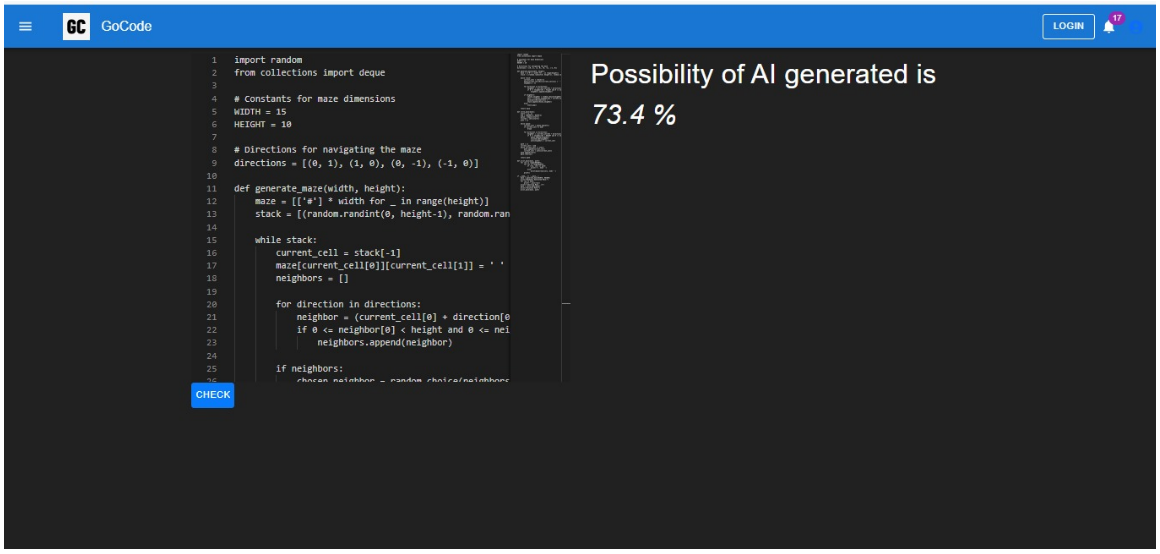
```python
import random
import string

def generate_password(length=8):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in

if __name__ == "__main__":
    password = generate_password()
    print("Randomly Generated Password:", password)

```

Possibility of AI generated is

56.33 %

Figure 4.21: Possibility of Similarity from AI tools

Figure 4.21 illustrates the potential for similarity between source code generated by AI-powered tools like ChatGPT and the existing codebases. This depiction highlights the risk of unintentional plagiarism or code similarity when using AI-generated code in software development projects. While AI tools can streamline coding processes and assist developers, it's crucial to be mindful of the possibility of generating code that closely resembles existing solutions. By acknowledging this possibility, developers can take proactive measures to ensure the originality and integrity of their code, such as incorporating rigorous code review processes and using plagiarism detection tools. Additionally, fostering a culture of ethical coding practices and encouraging proper attribution of sources can help mitigate the risk of unintentional code similarity in AI-generated code.



```
import random
from collections import deque

# Constants for maze dimensions
WIDTH = 15
HEIGHT = 10

# Directions for navigating the maze
directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]

def generate_maze(width, height):
    maze = [['#'] * width for _ in range(height)]
    stack = [(random.randint(0, height-1), random.ran

    while stack:
        current_cell = stack[-1]
        maze[current_cell[0]][current_cell[1]] = ' '
        neighbors = []

        for direction in directions:
            neighbor = (current_cell[0] + direction[0
            if 0 <= neighbor[0] < height and 0 <= nei
                neighbors.append(neighbor)

        if neighbors:
            chosen_neighbor = random.choice(neighbors
```

Possibility of AI generated is

73.4 %

Figure 4.22: Possibility of Similarity from AI tools

Figure 4.22 illustrates the potential for similarity between source code generated by AI-powered tools and existing codebases. This depiction highlights the risk of unintentional plagiarism or code similarity when using AI-generated code in software development projects. While Generative AI (GenAI) tools like ChatGPT can streamline coding processes and assist developers, it's crucial to be mindful of the possibility of generating code that closely resembles existing solutions. By acknowledging this possibility, developers can take proactive measures to ensure the originality and integrity of their code, such as incorporating rigorous code review processes and using plagiarism detection tools. Additionally, fostering a culture of ethical coding practices and encouraging proper attribution of sources can help mitigate the risk of unintentional code similarity in AI-generated code.

Table 4.1: Contest Result

UserName	**SimilarityScore**	**Rating**
Ritik	63.4	80
Mayur	41.6	60
Shaurya	33.9	90

Following the conclusion of the coding contest, the contest results were unveiled, unveiling participants' standings in the competition. Each participant's username was displayed alongside their corresponding similarity score for submitted code and the rating bestowed for solving coding questions. These metrics offered valuable insights into participants' coding abilities, with higher similarity scores signifying originality in code solutions and ratings reflecting the level of proficiency and accuracy in tackling contest questions. The contest results served as a comprehensive summary of participants' achievements, shedding light on their coding prowess and performance throughout the contest. Table 4.1 shows the results of coding contest.

Table 4.2: Plagiarism Report

UserName	**Website-Plagiarism**	**AI-Generative-Score**
Ritik	38.4	19.6
Mayur	29.1	15.3
Shaurya	22.9	11.5

Each participant's username was accompanied by a website plagiarism report for their submitted code and an assessment of AI-generated code similarity for the solved coding questions. These reports provided insights into participants' coding integrity, with website plagiarism reports indicating adherence to original code creation and AI-generated plagiarism assessments offering insights into the uniqueness of code solutions. The contest results served as a comprehensive overview of participants' achievements, showcasing their coding abilities and ethical practices throughout the contest. Table 4.2 shows the results of plagiarism score from Website and GenAI.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Our proposed coding platform can be very beneficial for the student's coding skills. It will promote the coding culture at the college level. The instructor of the course can use it to evaluate the codes given in the assignment. As we will implement the plagiarism checker, the evaluation will be fair without any favoritism of TA's towards any students. We will be checking in our system whether code is generated by any generative AI or not , ultimately it will help students to sharpen their coding skills.This system will help students prepare during the internship and placement season as students can directly practise the coding questions,Data structure and algorithm from this platform only since this platform includes all the variety of problems available all over the different coding sites like codeforce,codechef,leetcode,topcoder,etc.

5.2 Future Work

There are quite a good number of future improvements which we intend to make on the GUI part in our project, such as providing the user with even more problems, adding a timer for every problem the user starts, providing much more compile language options, and adding different themes to the code editor. We also intend to incorporate the plagiarism if the code is copied from various emerging AI platforms like ChatGPT-4, AI Devin, Gemini, etc. We aim to keep improving our website to the fullest benefits of the end user.Along with the coding platform we will try that whichever company visits the college their coding test will be held on our platform only.There is no need for the company to hold the test on other platform.

REFERENCES

[1] S. Karthikeyan K. Saranya T. K. Chandru, M. Dinesh Kumar. Interactive coding platform for students. *International Journal of Recent Technology and Engineering (IJRTE)*, 2018.

[2] Yilun Fu Baojiang Cui Jingling Zhao, Kunfeng Xia. An ast-based code plagiarism detection algorithm. *10th International Conference on Broadband and Wireless Computing, Communication and Applications*, 2015.

[3] Oscar Karnalim. A low-level structure-based approach for detecting source code plagiarism. *IAENG International Journal of Computer Science*, 44(4), 2017.

[4] Mayank Agrawal and Dilip Kumar Sharma. A state of art on source code plagiarism detection. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 236–241. IEEE, 2016.

Appendix A

BIODATA

Person 1

- **Name:** Ritik Kumar
- **Roll No:** 201IT250
- **Email:** ritik2245@gmail.com
- **Phone No:** 7351500348
- **Address:** Street No.- A/3, Adarsh Nagar Najibabad, Uttar Pradesh.

Person 2

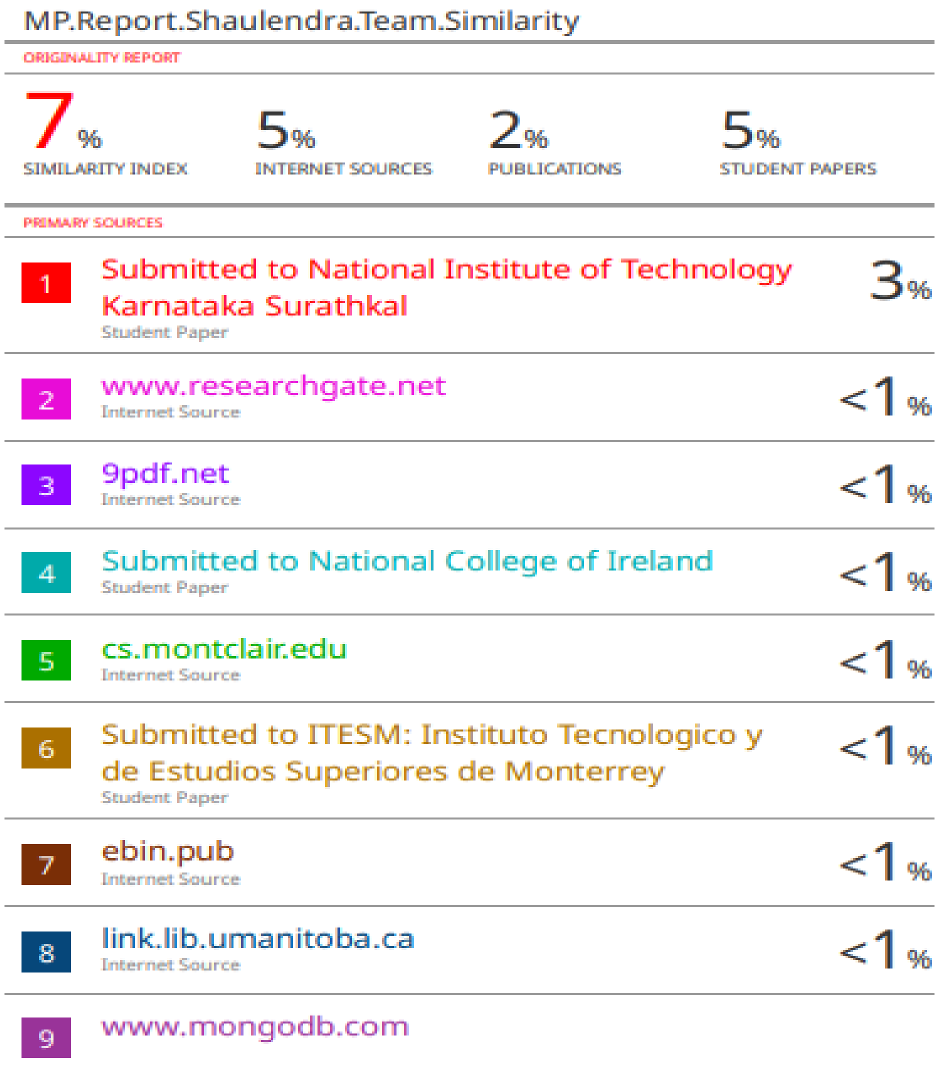
- **Name:** Shaulendra Kumar
- **Roll No:** 201IT159
- **Email:** shaulendra1@gmail.com
- **Phone No:** 8618179908
- **Address:** Flat No 30, Sri Venkateshwar Residency, Kattengalli, Bangalore.

Person 3

- **Name:** Mayur Jinde
- **Roll No:** 201IT135
- **Email:** jindemayur20@gmail.com
- **Phone No:** 8867289169
- **Address:** House No 30/4, Akkamahadevi Colony, Gulbarga, Karnataka.

Appendix B

SIMILARITY SCORE



MP.Report.Shaulendra.Team.Similarity

ORIGINALITY REPORT

7%	5%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to National Institute of Technology Karnataka Surathkal Student Paper	3%
2	www.researchgate.net Internet Source	<1%
3	9pdf.net Internet Source	<1%
4	Submitted to National College of Ireland Student Paper	<1%
5	cs.montclair.edu Internet Source	<1%
6	Submitted to ITESM: Instituto Tecnologico y de Estudios Superiores de Monterrey Student Paper	<1%
7	ebin.pub Internet Source	<1%
8	link.lib.umanitoba.ca Internet Source	<1%
9	www.mongodb.com	

Figure B.1: Originality Report