



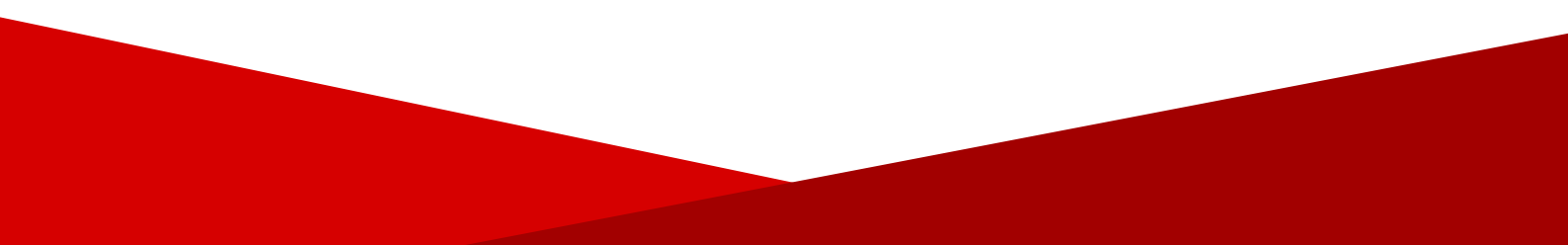
DOJA

THE ONLINE JUDGE

PROJECT REPORT



INDEX OF REPORT

- 1. Acknowledgement**
 - 2. Motivation Behind the Project**
 - 3. Analysis of Research Paper**
 - 4. Overall Project Design**
 - 5. Features of Project**
 - 6. Languages / Technologies used**
 - 7. Algorithm / Description of the work**
 - 8. Results**
 - 9. Conclusion**
 - 10. Declaration Form**
- 

MOTIVATION BEHIND THE PROJECT

To get the idea of the motivation behind this project first we need to know what is Competitive Programming?

Competitive Programming is a mental sport which enables you to code a given problem under provided constraints. Consider a programming contest as a game of cricket, metaphorically. Compile your code and submit.

There are many platforms for competitive programming like Codeforces, Codechef, Hackerrank, Atcoder, TopCoder etc.

There are contests that can be given individually or in a team of generally 3 people. ICPC – the Olympic of Competitive Programming, is a team contest which is held once every year and is one of the prestigious competitions in the world of competitive programming.

We observed that the available platforms do not have support for team contests. Participating in a team contest requires coordination and communication.

We aim to provide these features with the help of a **Real Time Collaborative Code Editor and a Chat Window on one page** so that team members can code together simultaneously in one code editor and have the feature to communicate within one window or platform thus reducing the overhead of using third party messaging platforms.

This not only helps the programmers code in a focused environment but also reduces the cases of cheating where programmers can change their tabs and windows for communicating or sharing code snippets.

So to solve this problem our team has developed a platform which we call **DOJA – The Online Judge**. Here teams can participate in contests with live code editor and chatting facility so that they don't have to go through the copying and pasting of code and sending to other members and communicating using any other platforms for discussion and problem solving. This will improve the efficiency of the team and will save time and improve their speed.

ANALYSIS OF RESEARCH PAPER

Research papers used:-

1) <https://hal.archives-ouvertes.fr/hal-00921633/document> - LSEQ

2) <https://hal.inria.fr/inria-00432368/document> - LOGOOT

Logoot and LSEQ adopt CRDT(Conflict Free Replica Data Types) approach to solve the problem of Peer to Peer collaborative code editors.

CRDT is a data structure for achieving consistent data between replicas of data stored on different peers(clients) without any kind of coordination (e.g. transformation) between the replicas and where it is always mathematically possible to resolve inconsistencies that might come up like commutativity and idempotency of data.

In CRDT modifications produced locally are re-executed on remote replicas.

Logoot Model uses an identifier which is unique for every character present on the editor and composed of (pos, hs) where “hs” is the logical clock/counter of the client and “pos” is a list of position identifiers;

Pos : p1 p2 p3 p4 pn which designate a path on a tree,

Where p1 = (position, site) , position is an integer which is chosen from 0 to a fixed base number like 32 or 64. Site is the unique site id of the client which generated the operation.

The Code Editor is basically a 2D array of characters and so the CRDT stores the document as a 2D array of unique identifiers for every character.

Insertion and deletions can be done in Logarithmic time using binary search.

Collaborative Code editor can be built using a central relaying server. But that has many disadvantages like :-

- a) **Single Point of failure** - If the server goes down , users can not make any more connections with other users.
- b) **High Costs to scale** - a single central server performing all requests of connections and message transmission is not a sustainable model.
- c) **High Latency between users located in the same geographic location** - Even if users might be sitting in the same room, they would still require to connect to the central server which might be located in another continent.

These problems were highlighted and aimed to solve in the Logoot paper.

Logoot describes a Peer to Peer based architecture where every user is connected to an upper bound of users, although the initial connection requires the use of a central server.

LSEQ improves on the LOGOOT approach by using an adaptive allocation strategy for sequence CRDT resulting in a sub-linear upper bound in its spatial complexity whatever is the editing behaviour.

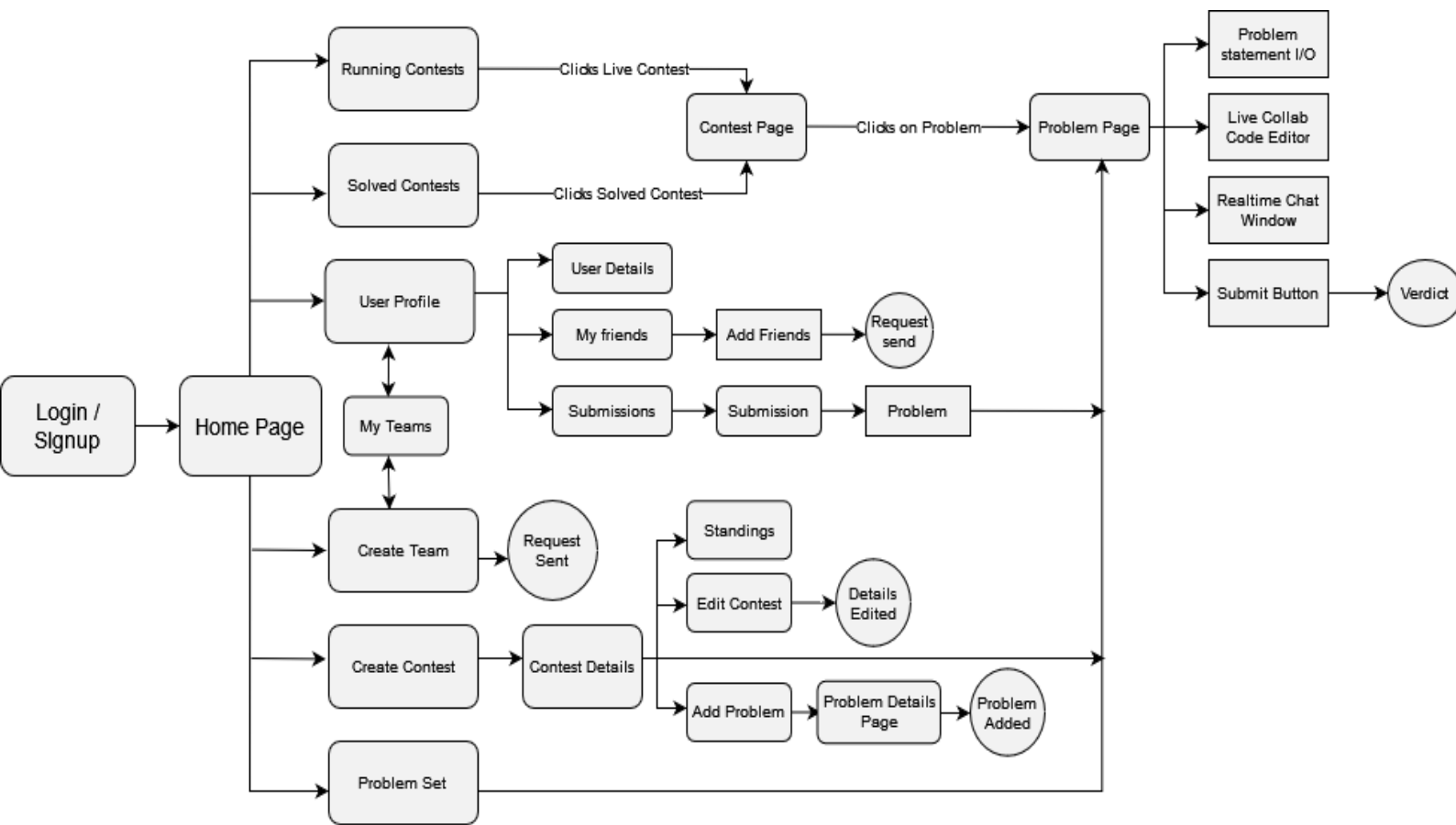
The Line positions on every level of the tree are doubled, so if the base level has 64 positions then the next level will have 128, and then next level will have 256 and so on.

Also the allocation strategy of choosing a position randomly in LOGOOT is modified by LSEQ by randomly choosing a position within boundary plus or boundary minus of left or right position. So if a position has to be chosen between p and q , and a fixed boundary, of 10 for example, has been defined, then if we randomly choose the left position then the possibilities would be $p+1, p+2$, till minimum of $(q, p+boundary)$ and same for randomly choosing right position.

Improvements we have been able to perform :-

- a) **Automatic Network Balancing** : We use an upper bound of Logarithmic complexity on the number of peers a peer or client can connect to.
- b) **Peer Discovery** - We maintain a list of peers or clients active on the network, so if a peer loses connection it can join the code editor again with the help of a peer list maintained on the network.
- c) **Logarithmic Insert and Delete using a 2D document model** - Since Code Editor is a 2D array of lines and characters, we also made the CRDT structure store a 2D document model of position identifiers for the characters.
- d) **Simple Code Highlighting** - Keyword highlighting of reserved words of programming languages just for better visual experience.

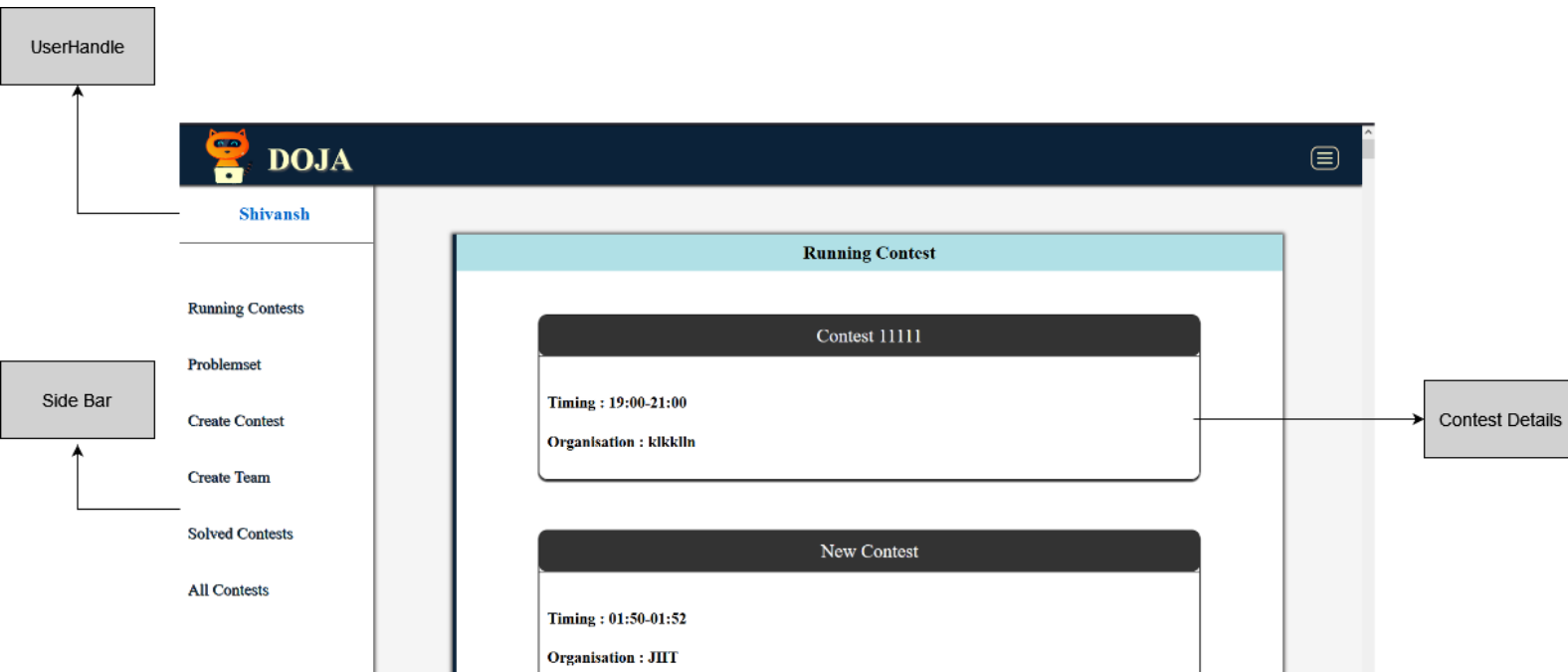
OVERALL PROJECT DESIGN



Project Design and Workflow of DOJA Online Judge

FEATURES OF PROJECT

1. **Running Contest Page** shows the current contests which are active/live right now. User can navigate from here to the contest he/she wants to attempt.

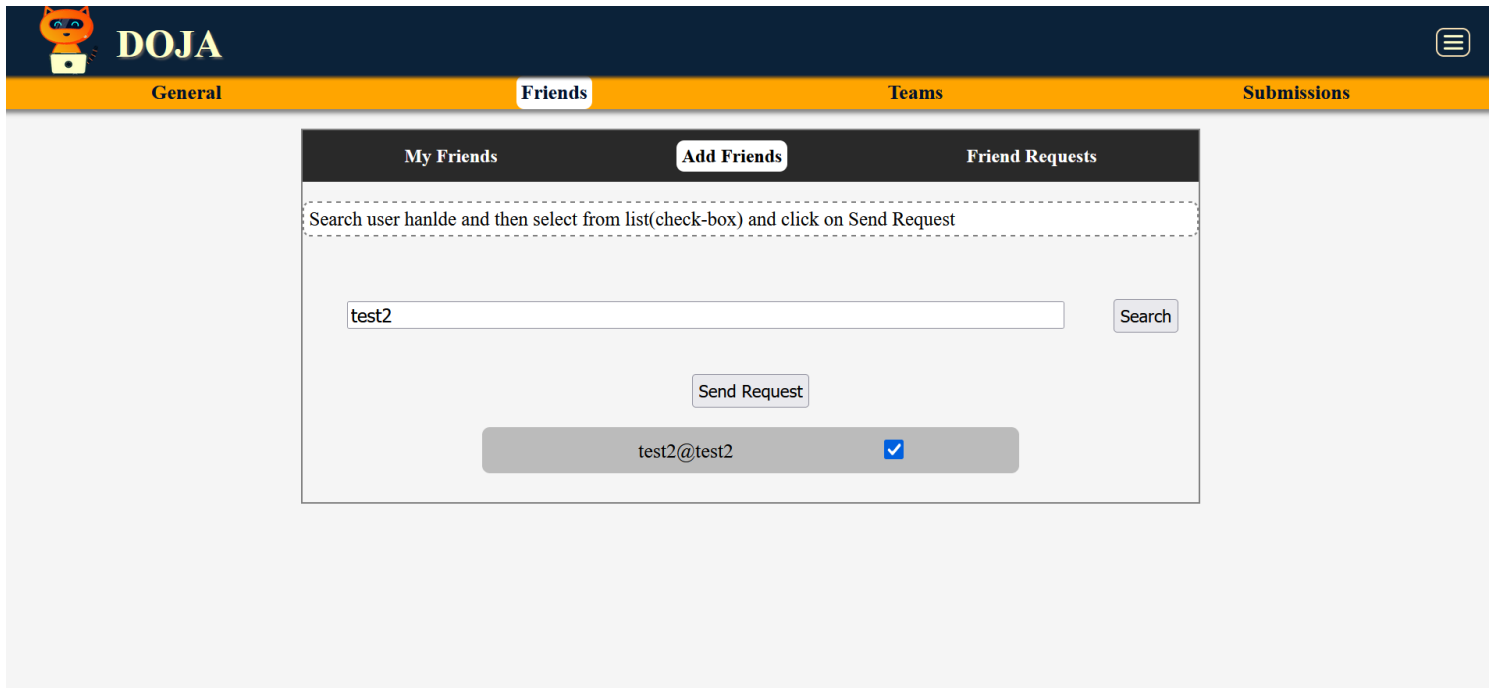


2. **Create Contest Page** is used to create the contest. User can add the Title of the contest, Description / about the contest, Date and Time of the contest, Organisation name, activate button and Prize details.

The screenshot shows the 'Create Contest' form in the DOJA application. The form is titled 'Create Contest' and includes the following fields and controls:

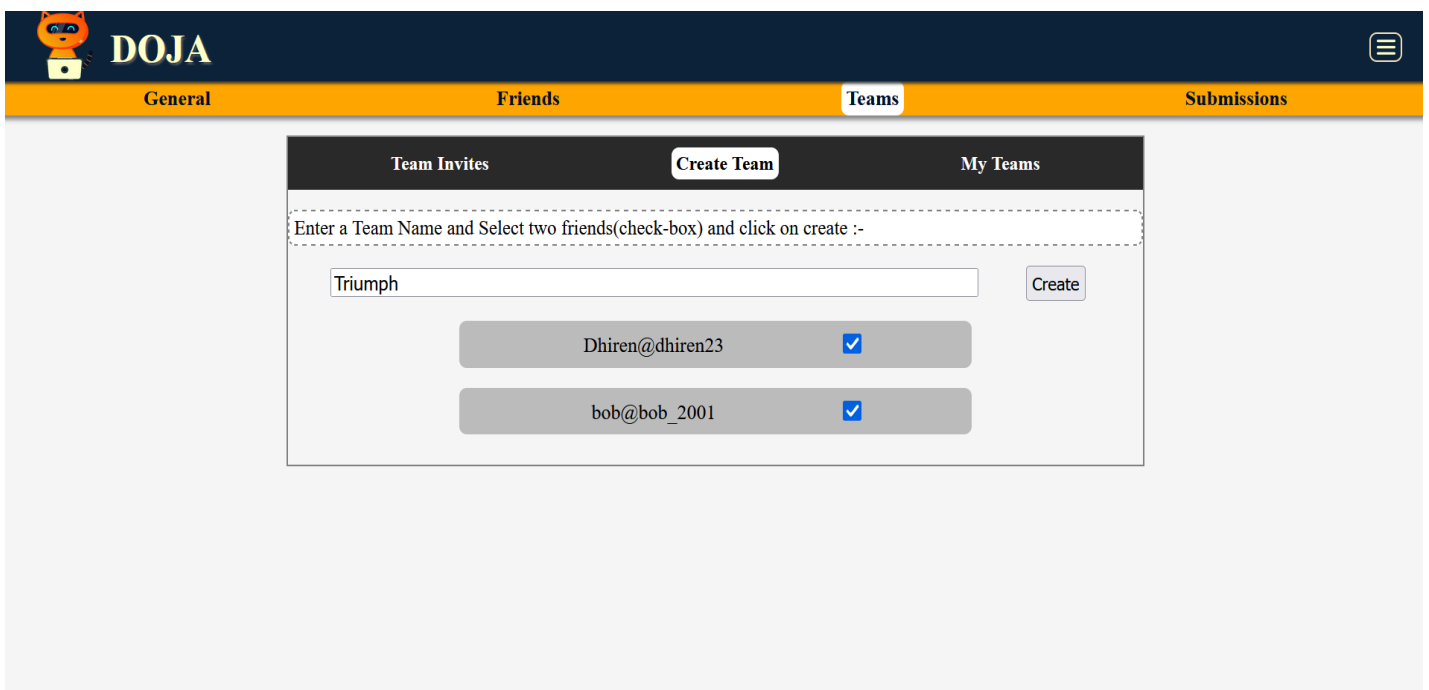
- Title:
- Description:
- Start Time:
- End Time:
- Date:
- Activate: ☐
- Organisation:

- 3. Add Friends Page** used to send friend request to the user handle that you type in the search box. If the user is not present in the database then request is not sent as there is no such user present on our platform.



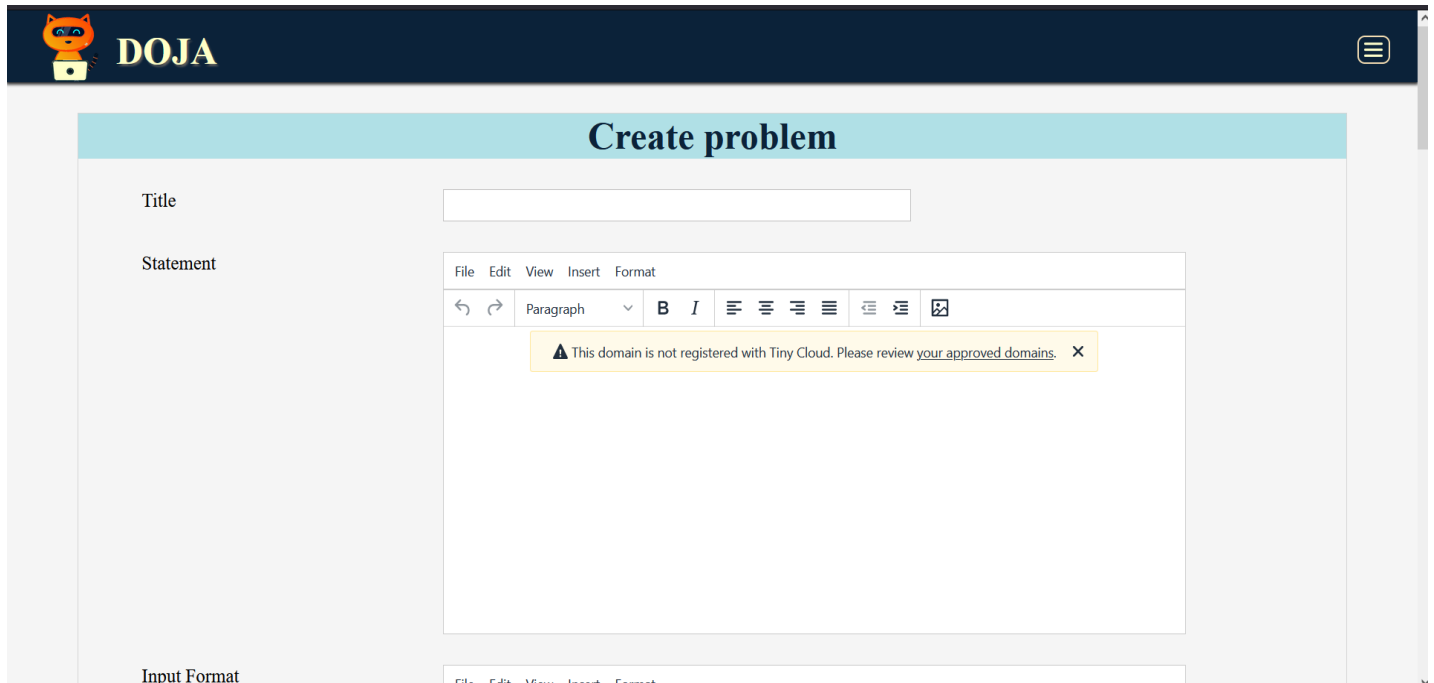
The screenshot shows the DOJA web application interface. The top navigation bar is dark blue with the DOJA logo and a hamburger menu icon. Below it, a yellow bar contains the main navigation tabs: General, Friends, Teams, and Submissions. The 'Friends' tab is active. The main content area has a dark header with three sub-tabs: My Friends, Add Friends, and Friend Requests. The 'Add Friends' sub-tab is selected. Below the sub-tabs, there is a dashed border containing the instruction: 'Search user handle and then select from list(check-box) and click on Send Request'. A search input field contains the text 'test2', and a 'Search' button is to its right. Below the search field is a 'Send Request' button. At the bottom, a list item shows the handle 'test2@test2' with a checked checkbox.

- 4. Create Team Page** used to create a team for participating in contests. Select two of your friends from your friend list and then click create team button. Your team will be created.



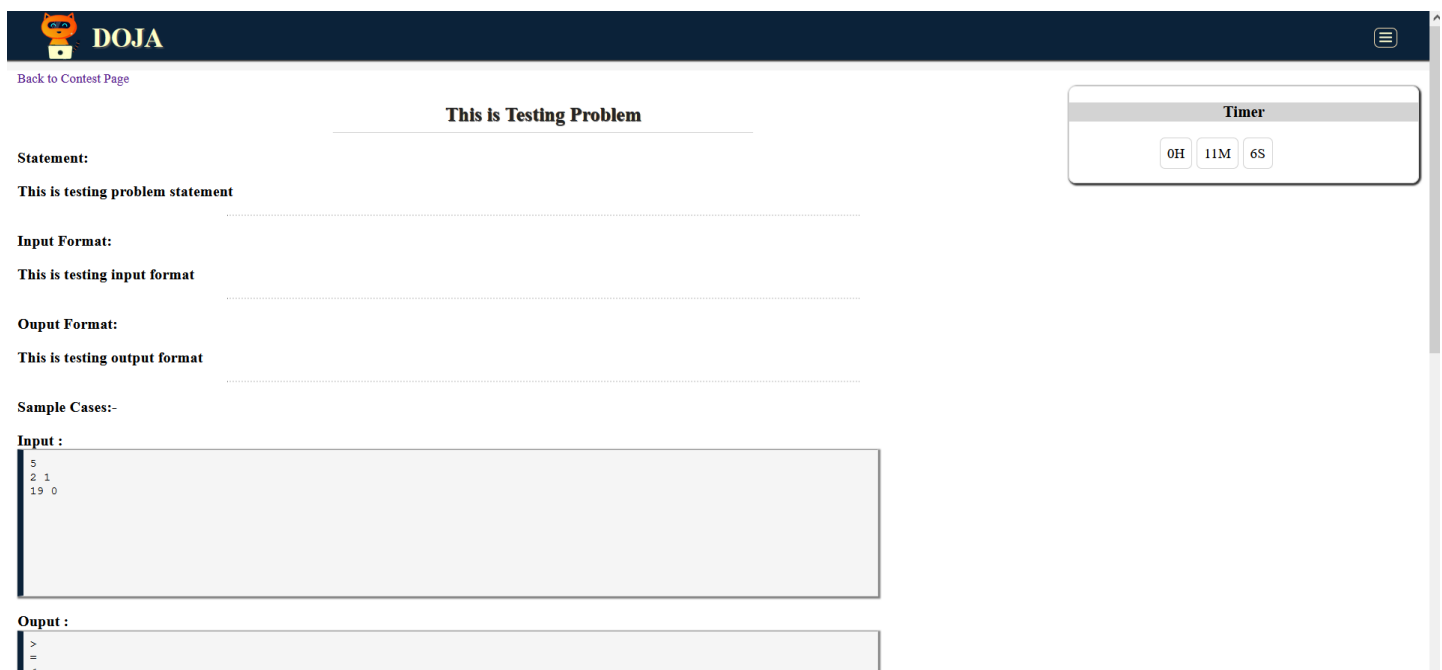
The screenshot shows the DOJA web application interface. The top navigation bar is dark blue with the DOJA logo and a hamburger menu icon. Below it, a yellow bar contains the main navigation tabs: General, Friends, Teams, and Submissions. The 'Teams' tab is active. The main content area has a dark header with three sub-tabs: Team Invites, Create Team, and My Teams. The 'Create Team' sub-tab is selected. Below the sub-tabs, there is a dashed border containing the instruction: 'Enter a Team Name and Select two friends(check-box) and click on create :-'. A text input field contains the team name 'Triumph', and a 'Create' button is to its right. Below the input field, there are two list items, each with a checked checkbox. The first list item shows the handle 'Dhiren@dhiren23' and the second shows 'bob@bob_2001'.

5. Add Problem Page used to add problems to the given contest. By providing necessary details like problem statement, Input format, Output format, Constraints, Sample test cases and Main test cases one can add problems to a particular contest.

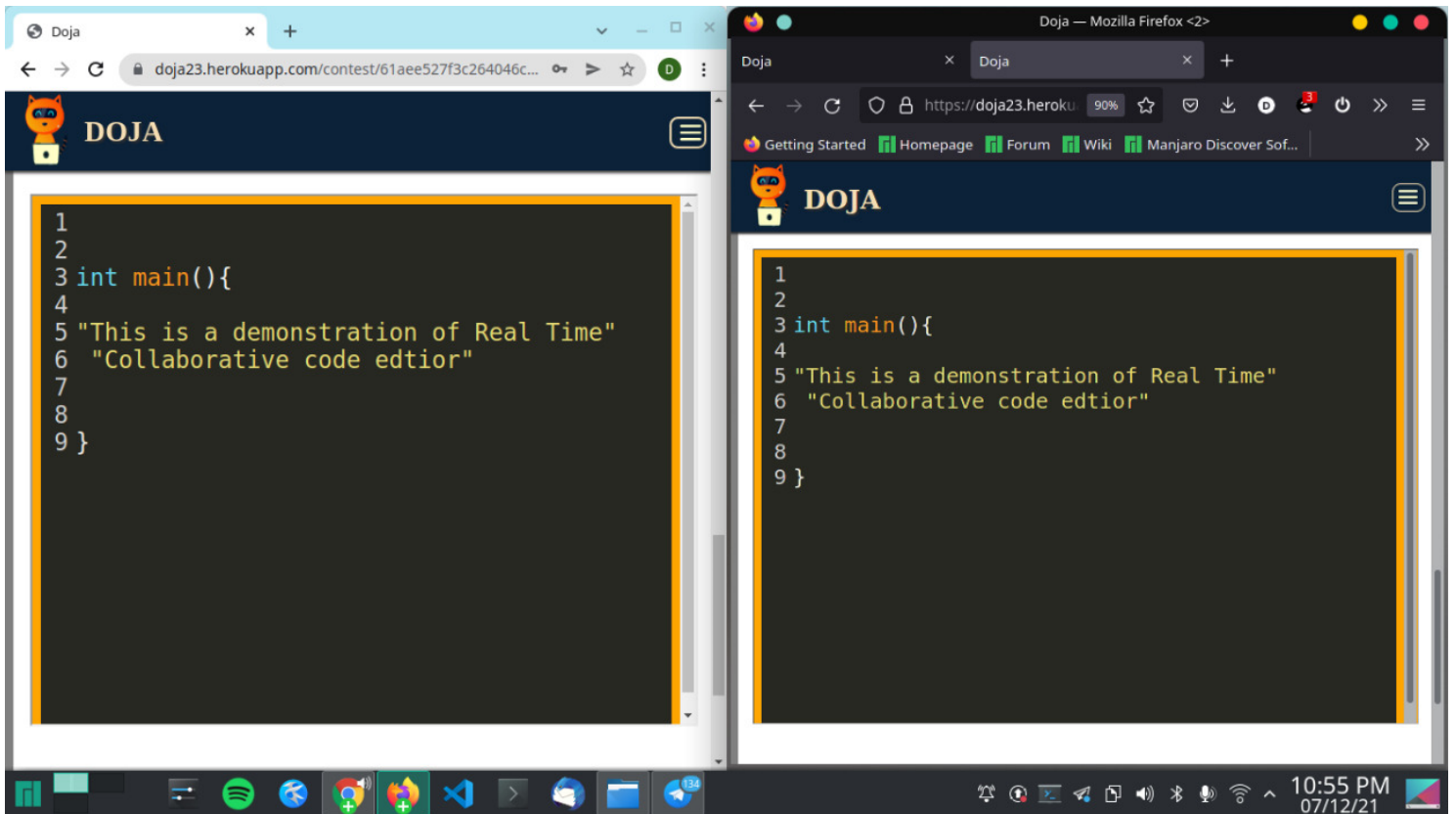


The screenshot shows the 'Create problem' page in the DOJA application. The page has a dark blue header with the DOJA logo and a menu icon. The main content area is light gray and contains a form with three sections: 'Title', 'Statement', and 'Input Format'. The 'Title' section has a text input field. The 'Statement' section has a rich text editor with a toolbar containing options for undo, redo, paragraph, bold, italic, bulleted list, numbered list, link, unlink, and image. A yellow warning box is displayed in the statement editor, stating: '⚠ This domain is not registered with Tiny Cloud. Please review your approved domains. ✕'. The 'Input Format' section has a text input field.

6. Problem Page- Individual or team can solve a particular problem using our live collaborative code editor and can communicate via chat window right on the same page. They can run the sample cases and submit the code from here only.



The screenshot shows the 'This is Testing Problem' page in the DOJA application. The page has a dark blue header with the DOJA logo and a menu icon. The main content area is light gray and contains a form with several sections: 'Statement', 'Input Format', 'Output Format', and 'Sample Cases:-'. The 'Statement' section has a text input field with the placeholder text 'This is testing problem statement'. The 'Input Format' section has a text input field with the placeholder text 'This is testing input format'. The 'Output Format' section has a text input field with the placeholder text 'This is testing output format'. The 'Sample Cases:-' section has two text input fields: 'Input :' and 'Output :'. The 'Input :' field contains the text '5', '2 1', and '19 0'. The 'Output :' field contains the text '>', '<', and '<'. In the top right corner, there is a 'Timer' box with three buttons: '0H', '11M', and '6S'. A 'Back to Contest Page' link is located in the top left corner.

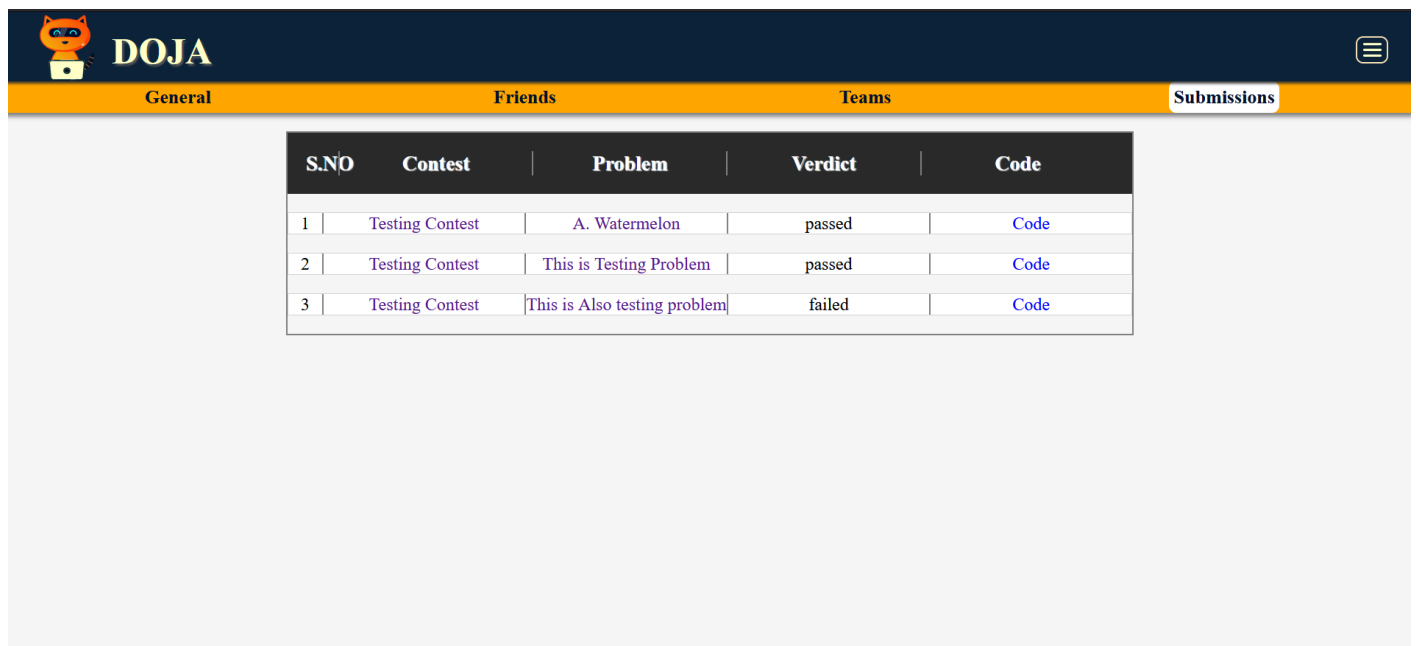


7. **User Profile Page** here a user can see his/her profile details like Handle, Name, Country, Gender, Organisation. From the same page the user can check and add Friends, Teams and can also see his submissions.

The image shows a screenshot of the Doja User Profile Page. The page has a dark blue header with the Doja logo and a navigation bar with tabs: General, Friends, Teams, and Submissions. The General tab is selected. The profile details are displayed in a form with the following fields:

DOJA	
Handle	bob_2001
Name	bob
Country	india
Gender	male
Organisation	jiit

8. Submissions Page here user can check all of his/her submissions made on DOJA. Here they can see the problem, the code that they submitted along with the verdict.



The screenshot shows the DOJA Submissions Page. At the top, there is a dark blue header with the DOJA logo (a cat with a laptop) and the text "DOJA". To the right of the header is a hamburger menu icon. Below the header is an orange navigation bar with four tabs: "General", "Friends", "Teams", and "Submissions". The "Submissions" tab is currently selected. Below the navigation bar is a table with five columns: "S.NO", "Contest", "Problem", "Verdict", and "Code". The table contains three rows of submission data.

S.NO	Contest	Problem	Verdict	Code
1	Testing Contest	A. Watermelon	passed	Code
2	Testing Contest	This is Testing Problem	passed	Code
3	Testing Contest	This is Also testing problem	failed	Code

LANGUAGES / TECHNOLOGIES USED

1. Front End

- HTML
- CSS
- Javascript
- EJS (Templating Engine)



2. Back End

- Javascript
- Nodejs
- Express.js
- Socket.io
- WebRTC



3. Database

- MongoDB
- Mongoose



4. Containerisation

- Docker



5. Version Control

- Git
- Github



ALGORITHM / DESCRIPTION OF THE WORK

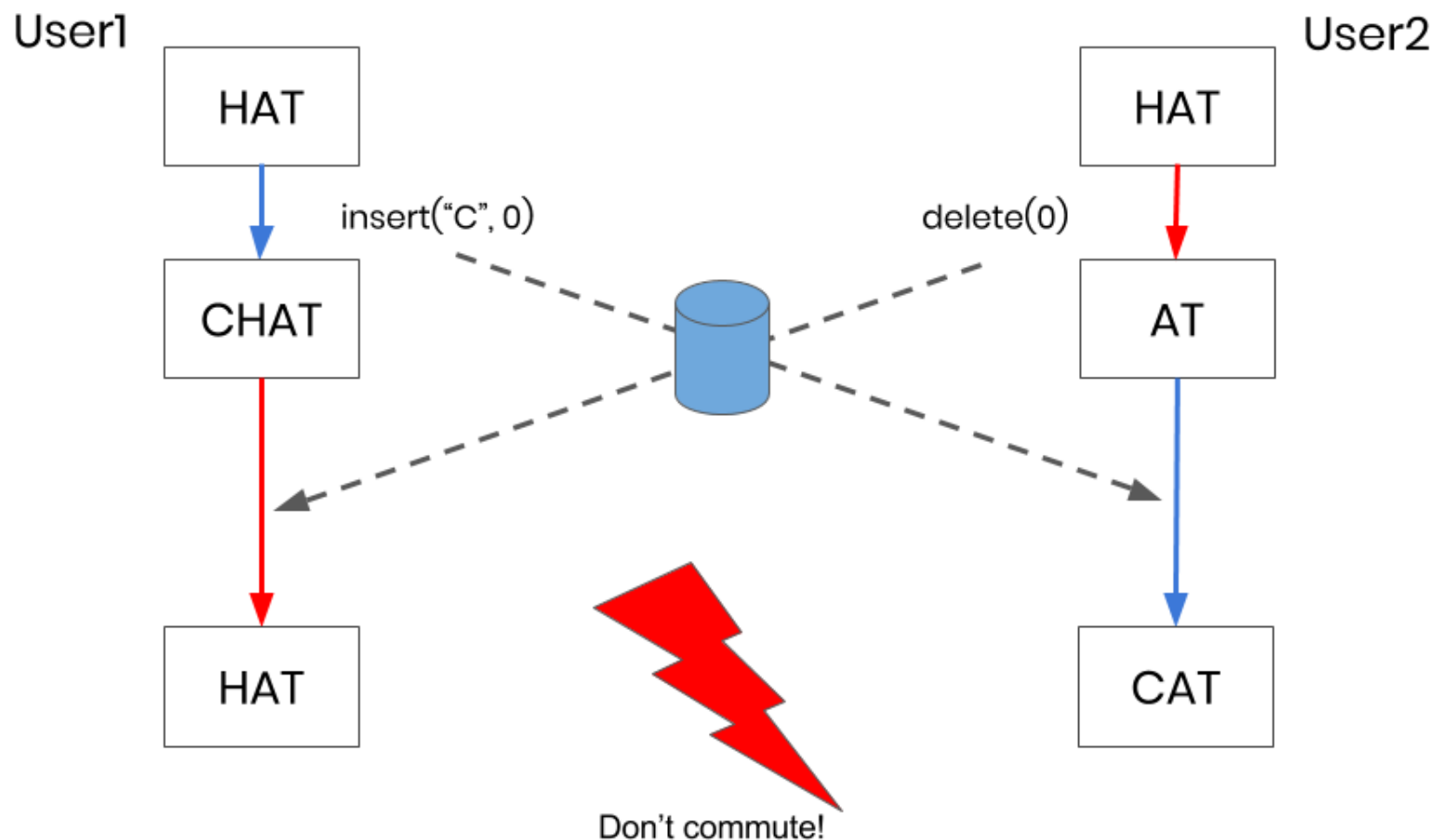
1. Live Code Editor -

A Real Time Collaborative code editor built with CRDTs (conflict free replica data type) and WebRTC which is a peer to peer based framework.

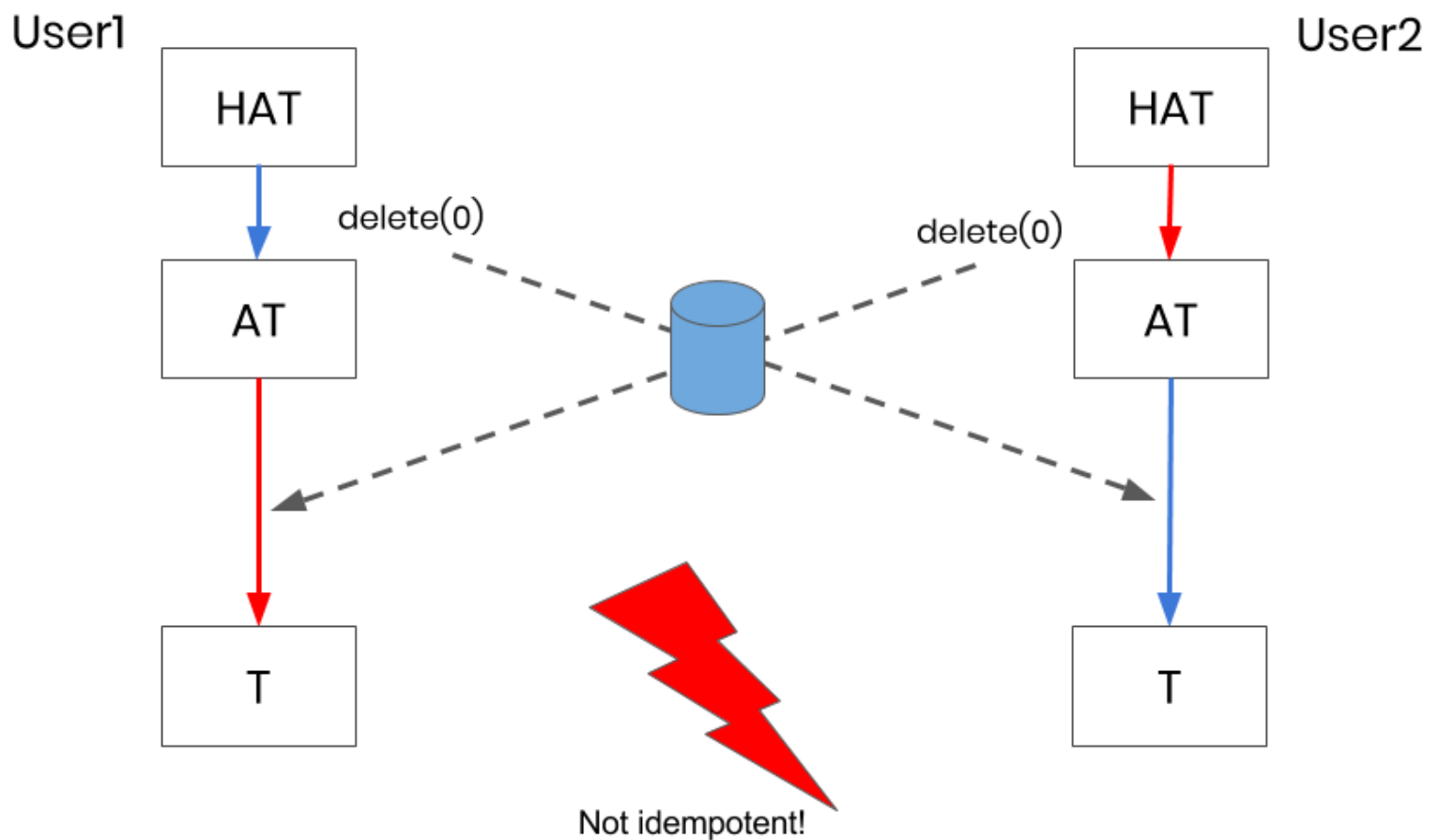
In a non-real time environment, users perform insert, delete and edit operations on a file either on **Lock** based approach or **Merge** based approach.

In **Lock** based approach, only one user can perform operations.

Whereas in **Merge** based approach, users have local copy of same files and they perform operations locally and then upload over a network or software which tries to look for conflicts and either merges those conflicts automatically or informs the user to take action.



Simultaneous insertion and deletion produce different results



Duplicate deletion operations are not idempotent

For collaborative code editors either of the approaches is not sustainable. So research was done and CRDT was born.

A CRDT is primarily a strategy for achieving consistent data between replicas of data without any kind of coordination (e.g. transformation) between the replicas. Since a text document requires the characters to be in a specific order, the type of CRDT that we used is usually categorized as a **Sequence CRDT**.

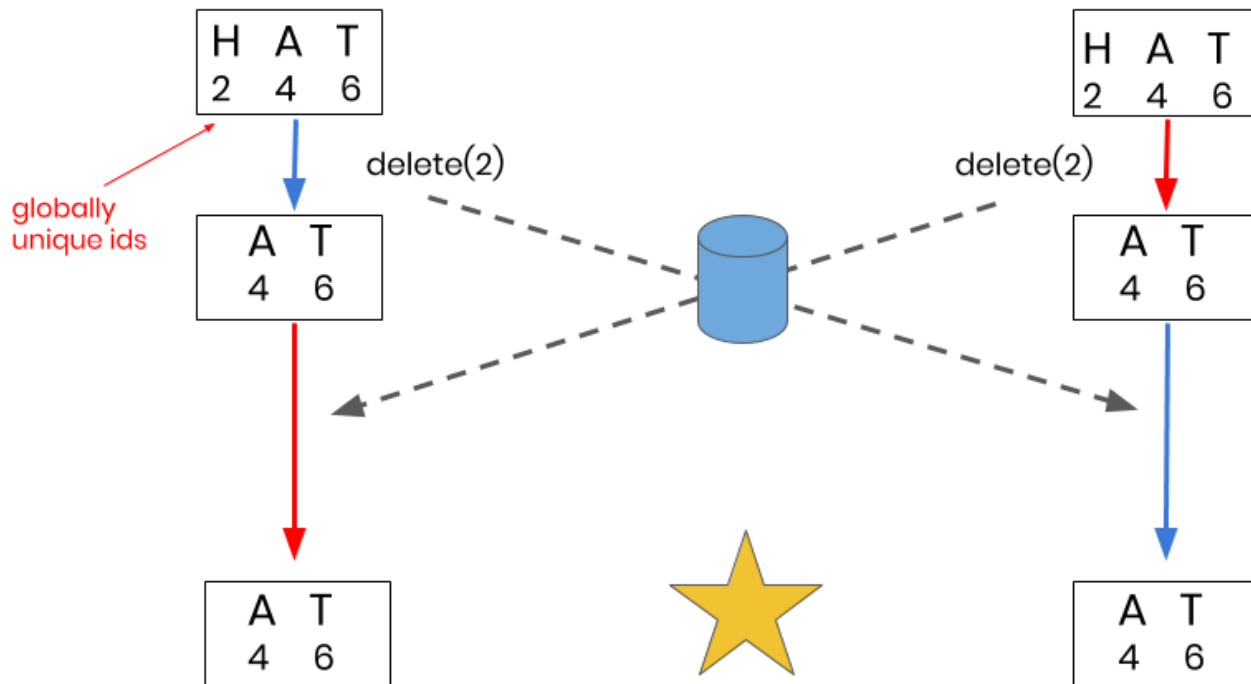
To use CRDTs specifically for a collaborative text editor, there are a couple critical requirements like :-

Globally Unique Characters

The 1st requirement is that each character object must be globally unique. This is achieved by assigning **Site ID** and **Site Counter** properties whenever a new character is inserted. Since the **Site Counter** at each site increments whenever inserting or deleting a character, we ensure the global uniqueness of all characters.

User1

User2

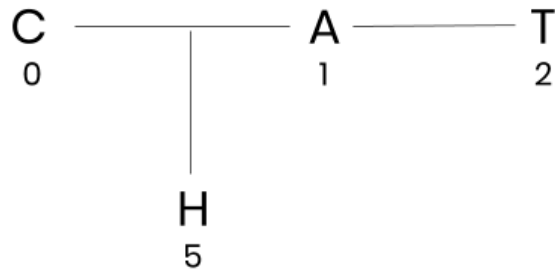
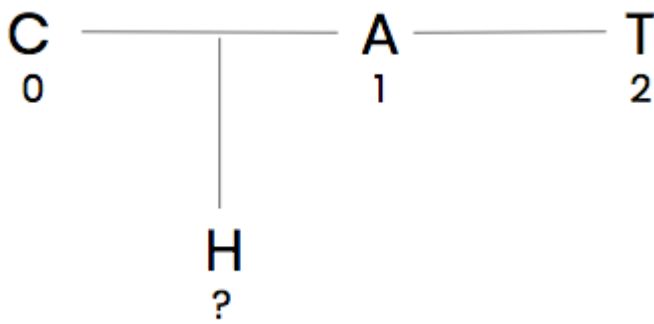


Duplicate deletion operations are idempotent with a CRDT

Globally Ordered Characters

The 2nd requirement for a collaborative text editor CRDT has to do with the positioning of characters. Since we're building a text editor, preserving the order of characters within a text document is required. But for a collaborative text editor where each user has their own copy of the document, we must go a step further. We need all the characters to be globally ordered and consistent. That means that when a user inserts a character, it will be placed in the same position on every user's copy of the shared document. The key is that by using fractional indices to insert characters, we never have to shift the positions of surrounding characters.

We can avoid this problem and ensure commutativity by using fractional indices as opposed to numerical indices. In the example below, when a user insert an "H" at position 1, they specifically intend to insert an "H" in between "C" and "A".



C	A	T
[0]	[1]	[2]

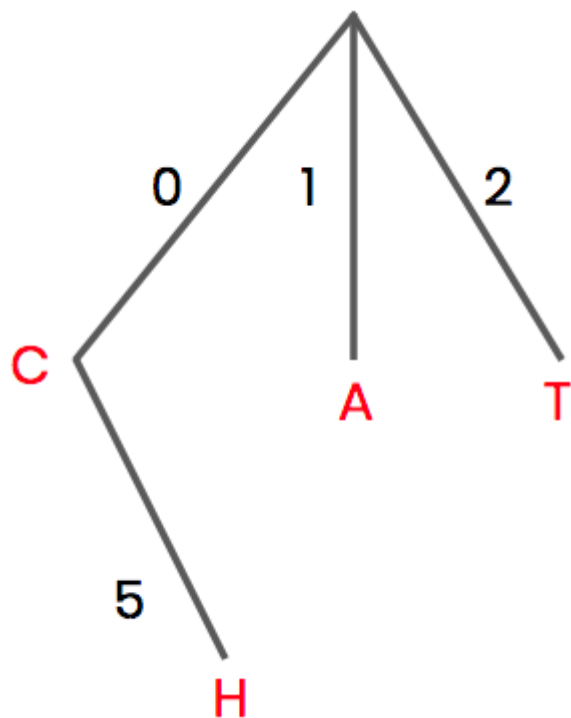


C	H	A	T
[0]	[0, 5]	[1]	[2]

Inserting a character at a position in between two existing characters.

Another way to imagine fractional indices is as a tree. As characters are inserted into the document, they can be inserted in between two existing position identifiers at one level of the tree. However, if there is no space between two existing character positions, as demonstrated below, we proceed to the next level of the tree and pick an available position value from there.

There are several academic papers dedicated to how best to “pick an available position”. We implemented an “adaptive allocation strategy for sequence CRDT” called [LSEQ](#).



User1

C	A	T
2	3	6

Insert("H", 2,4)

C	H	A	T
2	2,4	3	6

delete("A", 3)

C	H	T
2	2,4	6

User2

C	A	T
2	3	6

delete("A", 3)

C	T
2	6

Insert("H", 2,4)

C	H	T
2	2,4	6



CRDT

Insert and Delete operations commute using a CRDT

Coding CRDT required :-

a) **Web Text Editor** - We have used open-source [CodeMirror](#) text editor for it's ease of use and robust API.

b) **CRDT Structure :-**

```
class CRDT {  
    constructor(id) {  
        this.siteId = id;  
        this.struct = [];  
    }  
}
```

CRDT must handle 4 basic operations:

- **Local Insert:** User inserts character into their local text editor and sends the operation to all other users.
- **Local Delete:** User deletes character from their local editor and sends the operation to all other users.
- **Remote Insert:** User receives a insert operation from another user and inserts it to their local editor.
- **Remote Delete:** User receives a delete operation from another user and deletes it from their local editor.

c) **Peer to Peer Based framework** - We have used WebRTC. WebRTC is a protocol that was designed for real-time communication over peer-to-peer connections. It's primarily intended to support plugin-free audio or video calling but its simplicity makes it perfect for us even though we're really just sending text messages. While WebRTC enables our users to talk directly to one another, a small server is required to initiate those peer-to-peer connections in a process called "signaling". For signalling we are creating a Node.js server at the backend.

Time Complexity - We can achieve $O(\log(N))$ insert and delete operation complexity.

Space Complexity - A sub-linear complexity for maintaining the position identifiers for every character.

2. Realtime Chatting Window –

A WebSockets approach to make a real time chat system with the help of socket.io library

WebSockets allow to make a duplex continuous stream of data flow between the client and the server. Messages are transmitted on UDP protocol and are broadcasted to every user connected to the server.

Since users are participating in a team contest , they join a chat room with a unique id. Only team members can join a particular chat room and can send messages , check online status of their team members, share links , images and code snippets.

This saves the time of coders who otherwise would have to use third party messaging platforms for communication.

3. Online Judge:

We have implemented the Online Judge with the help of **docker containers and bash script.**

Whenever a submission is made, the submission route gets hit which gets the code, language of code and the problem ID.

With the help of problem ID, we get the details of the problem like the time limit, memory limit, sample test cases, main test cases along with their outputs.

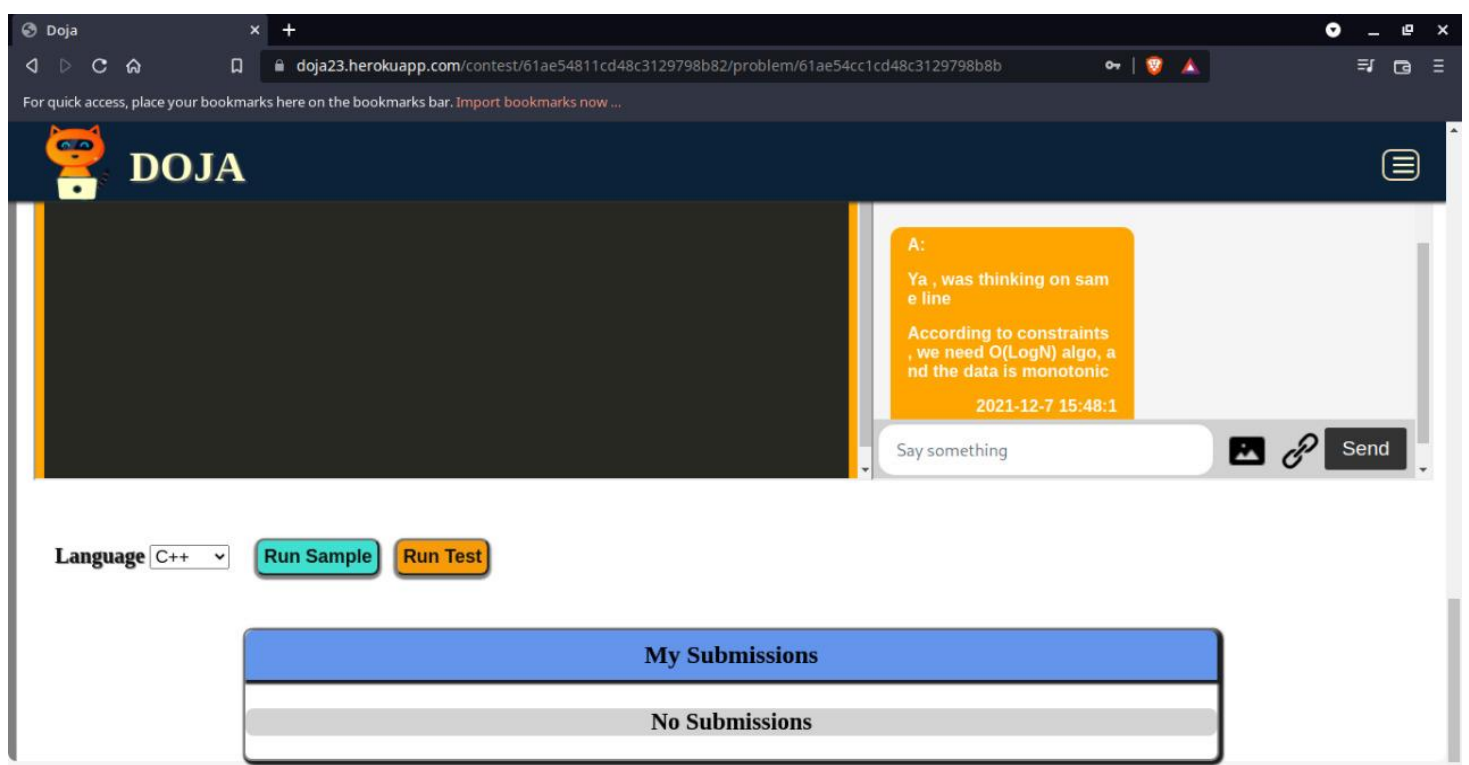
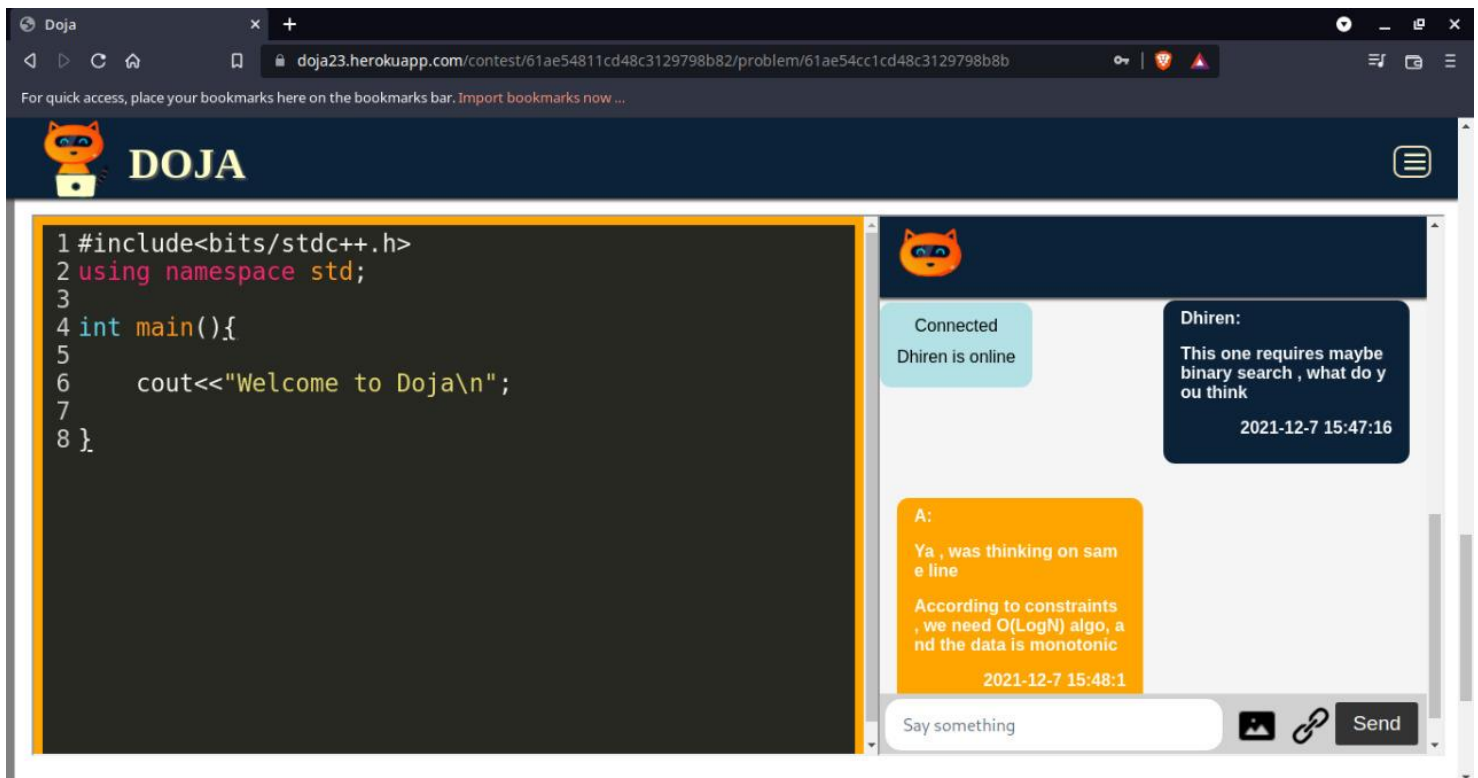
Then we run a bash script to run the program submitted by the user on the basis of language chosen in a docker container to prevent malicious code from running and blocking the main Node.js thread and preventing access to the database and private files

- If the output matches the given output of the test cases (actual output) then its **AC**(Accepted).
- If the output does not match with the actual output then the verdict is **WA**(Wrong Answer).
- If the code takes more time than the time limit then its **TLE**(Time Limit Exceeded).
- If the code takes more memory than the memory limit of the problem then its **MLE**(Memory limit exceeded).

The verdict of the code gets update in every 200ms and when the code is processed, the user is given the verdict for the submission.

RESULTS

As a result of this project, we have developed a platform where teams will get a feature of **Live code editor** along with **Realtime chatting** facility.





CONCLUSION

Now we have a platform for practicing in Team contests with Live Code Editor and Chat Window along with all features of an Online Judge like Judge, Submissions, creating contests , making Friends on the same platform and much more.

This will increase the team efficiency, their problem solving skills, provide practice for participation in team contests like ICPC, team work, time management skills and many more aspects.

This platform can be used by many organisations for conducting regular coding contests , team contests and coding competitions.

