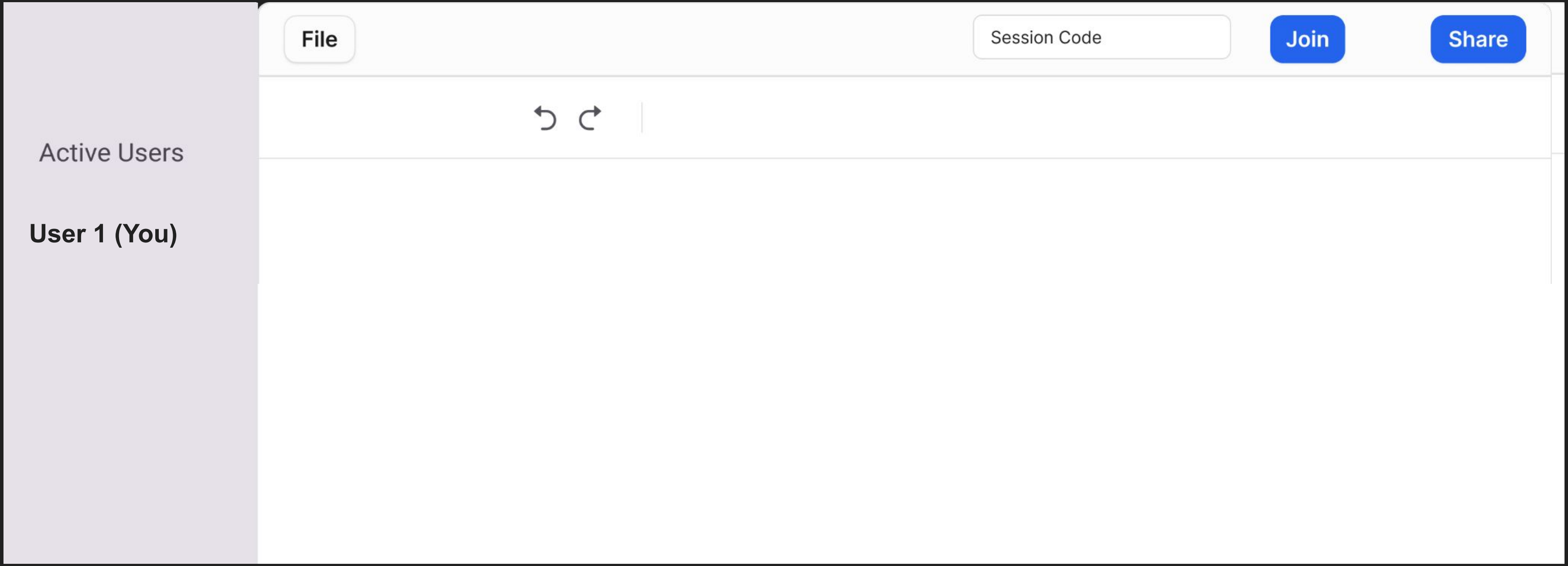
Collaborative Text Editing App

Outline

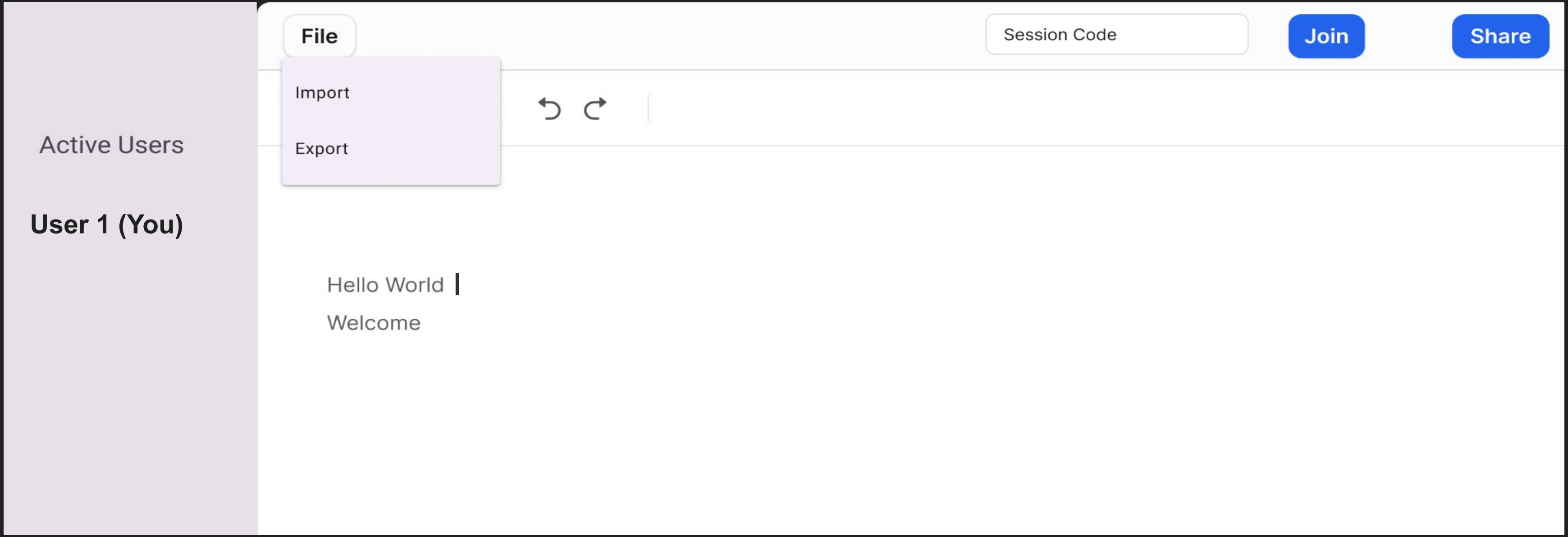
Overview

- Start with a blank document



Overview: Import/Export

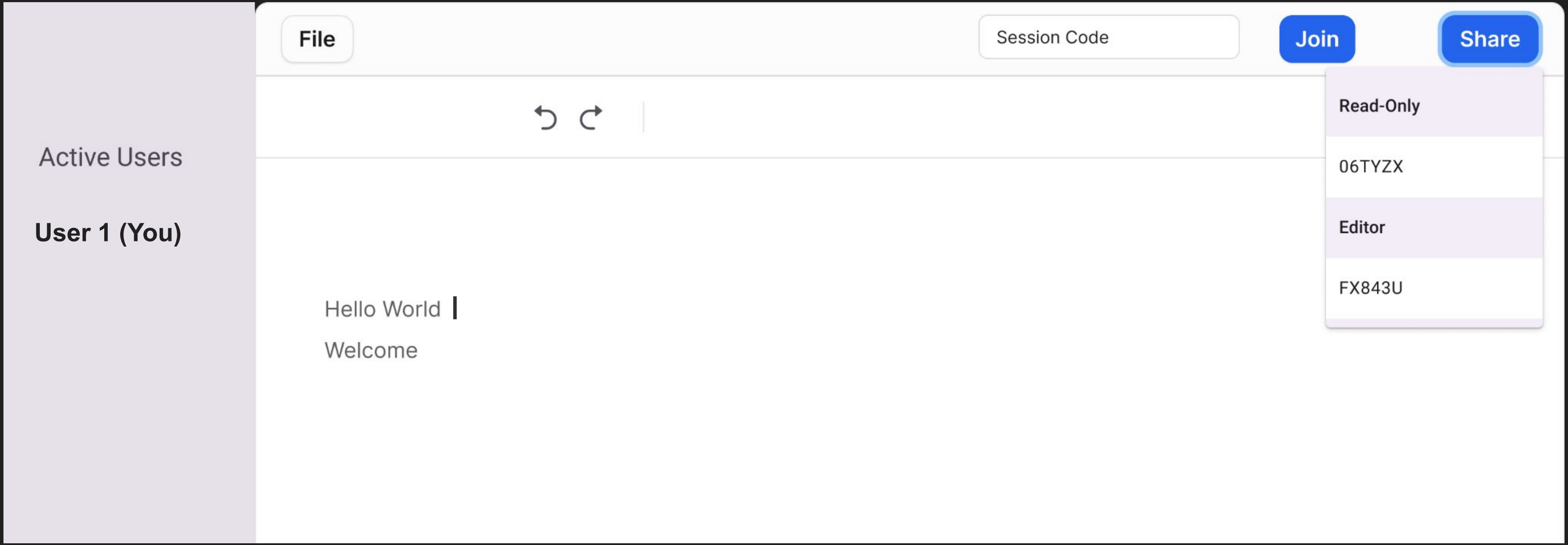
- Import/Export from/to the user's filesystem
- Support .txt files only
- Preserve line breaks



Hello World
Welcome

Overview: Sharing

- Request sharable code
- Different codes for viewers and editors



Read-Only

06TYZX

Editor

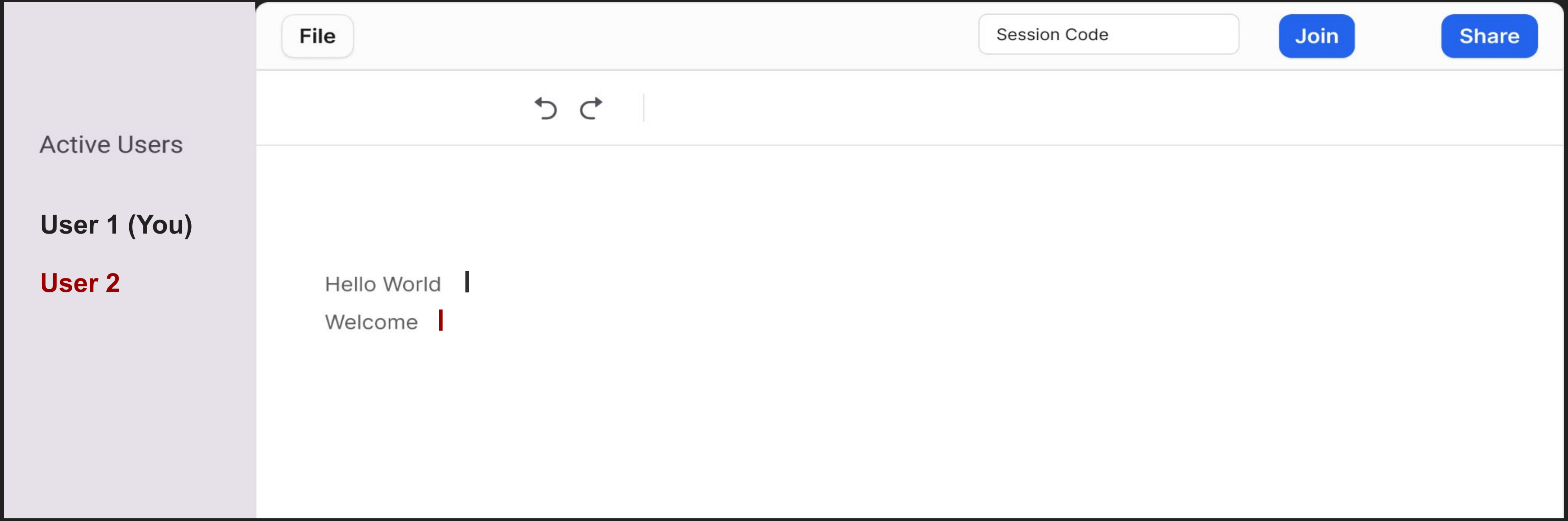
FX843U

Hello World

Welcome

Overview: Collaboration

- Join a collaboration session
- Track cursors of other users
- View names of users in the session

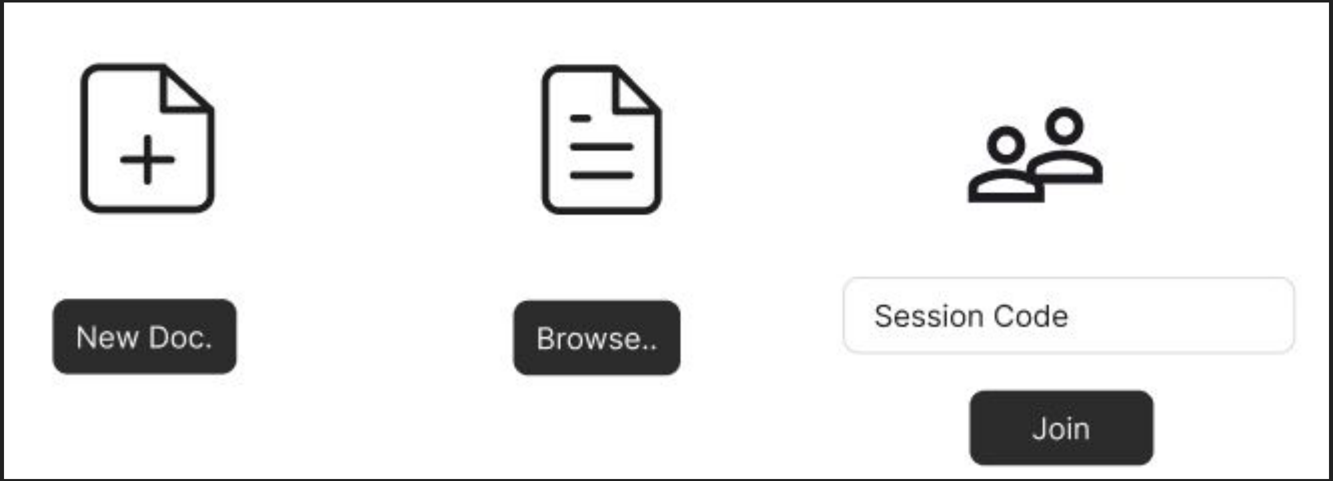


Hello World

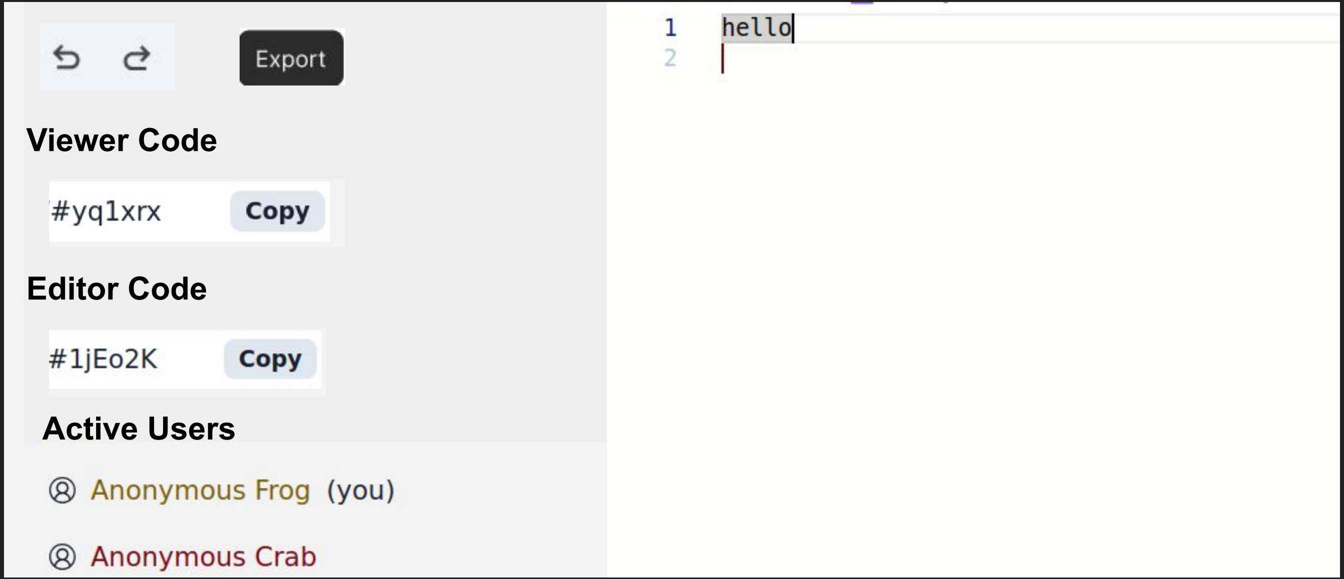
Welcome

Alternative UI Design (Recommended)

- Start with giving the user the option to
  - Join a session
  - Open a new blank document
  - Import a .txt file



Alternative UI Design (Recommended)



```
hello
```

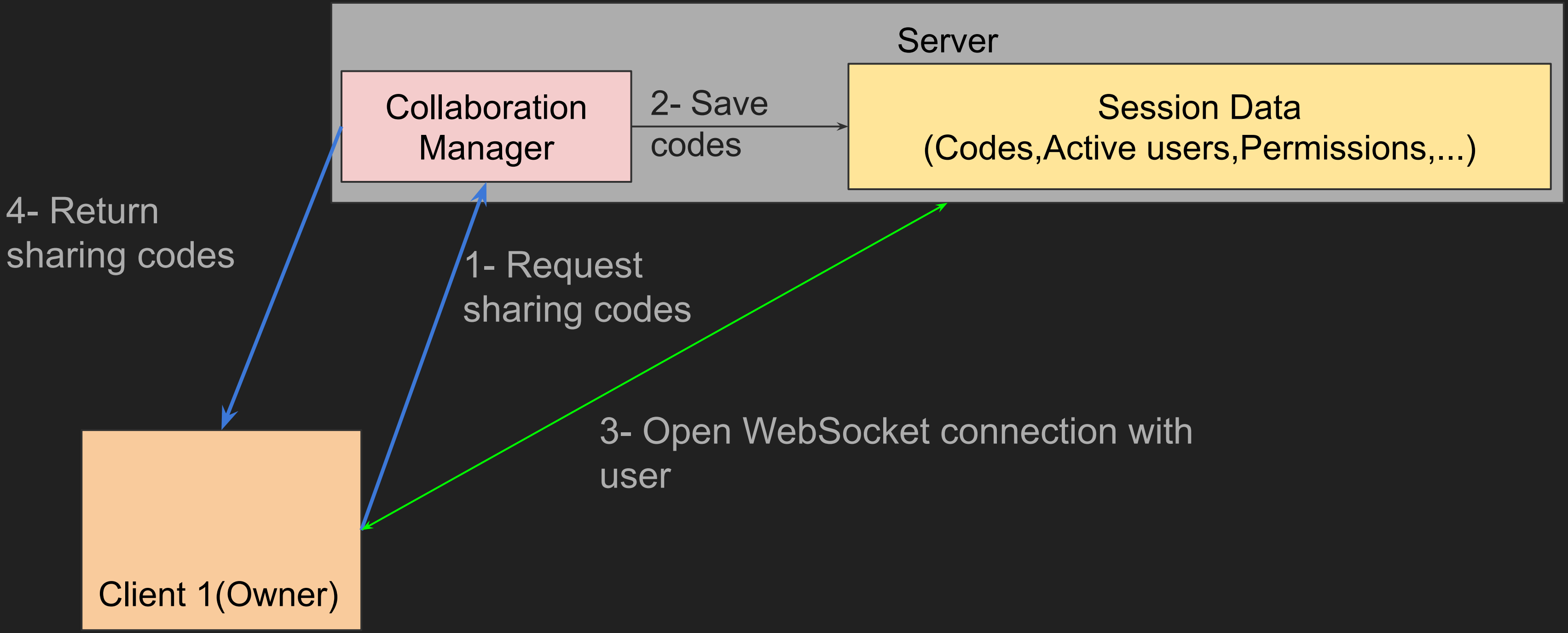
Overview: Collaboration

- Textarea editing disabled for viewers
- Viewers cannot see editor sharable code

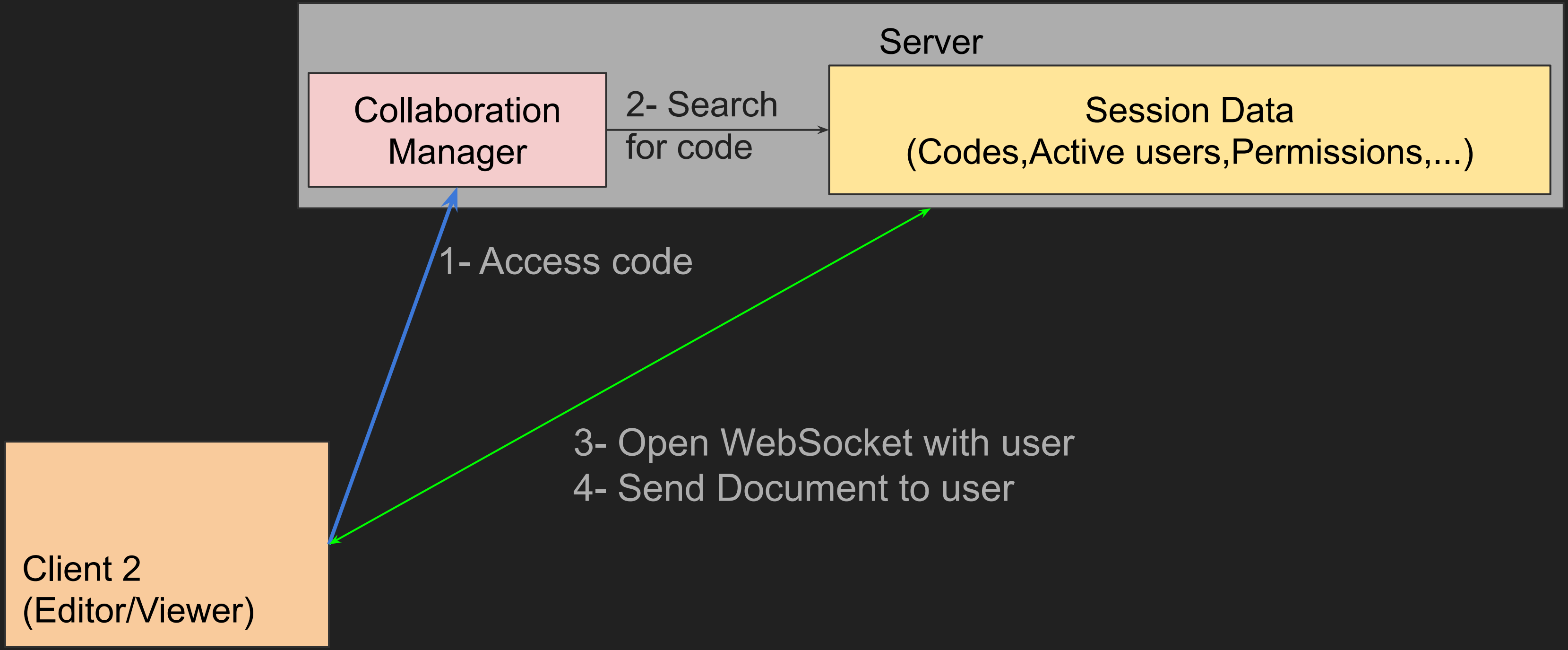
Overview: Editor Features

- Insert/Delete characters one-by-one
- Pasted text treated as sequential insertions
- Support ASCII characters only
- Undo/Redo the user’s own edits

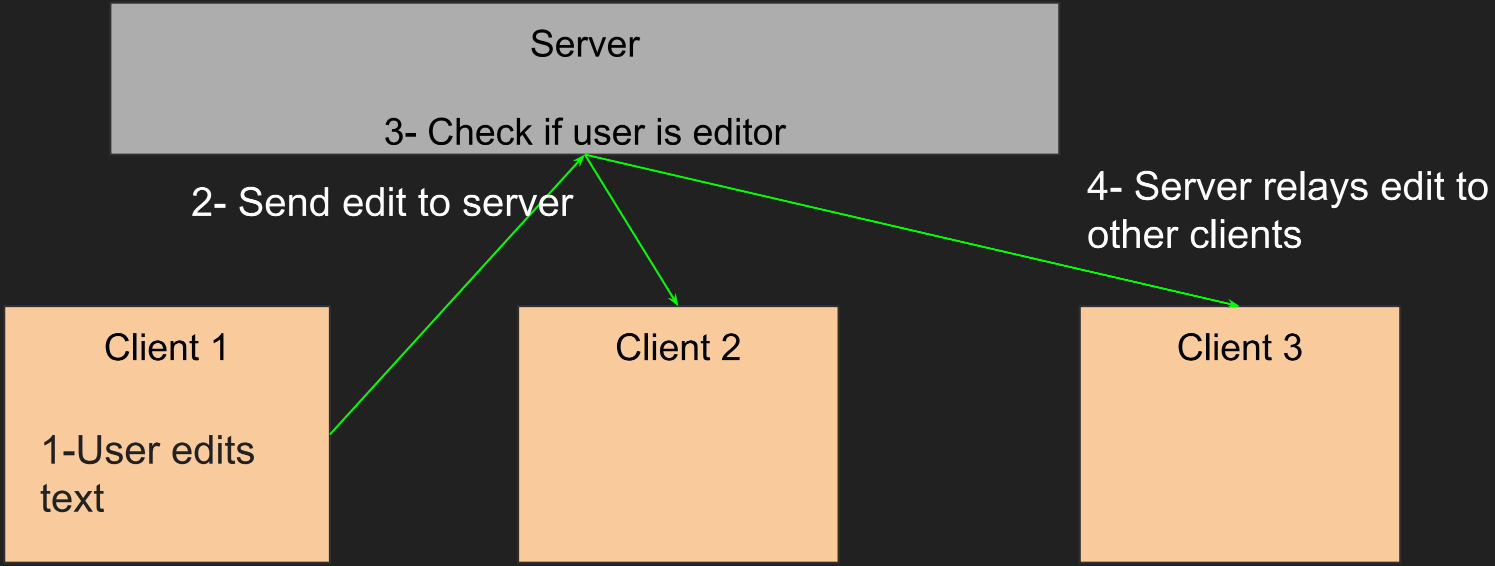
System Architecture: Document Sharing



System Architecture: Document Sharing

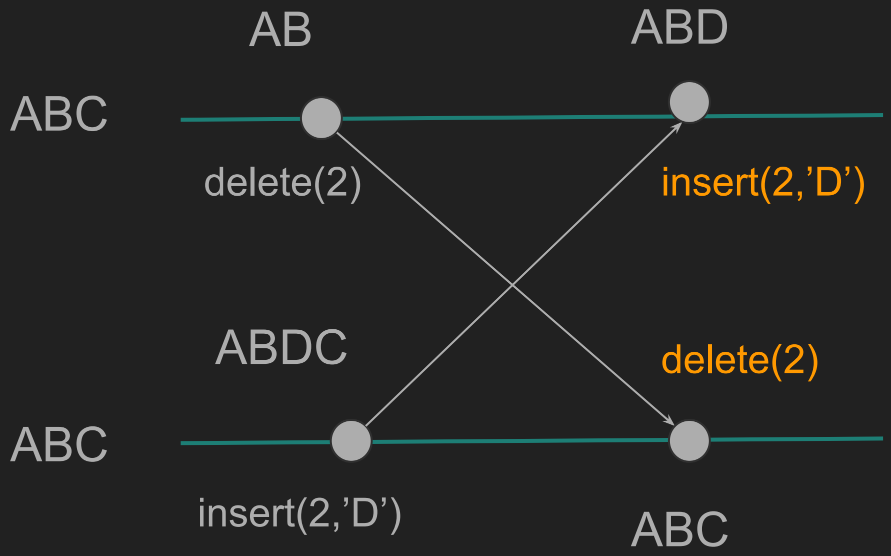


System Architecture: Collaborative Text Editing



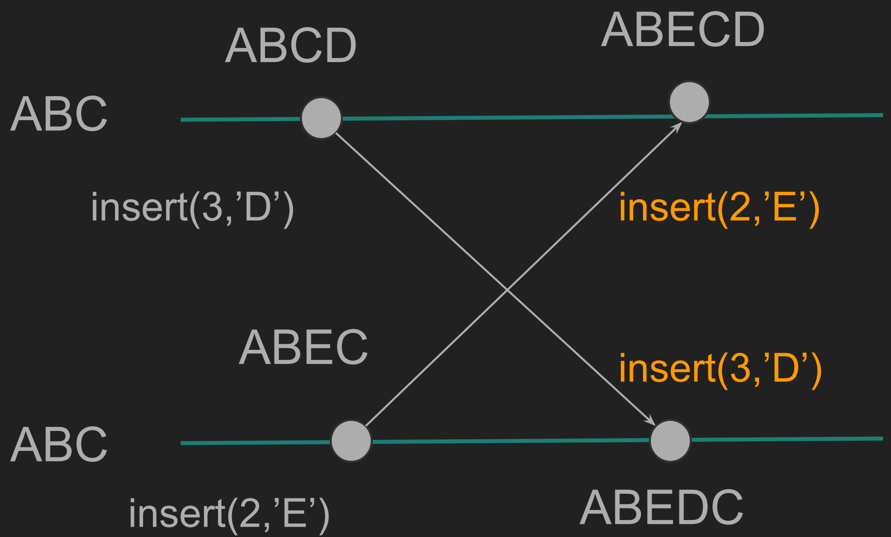
Concurrency Issues with Editing

- Main challenge of all parallel and distributed systems
- In text editing, concurrency problems lead to different document state at each user



Concurrency Issues with Editing

- Main challenge of all parallel and distributed systems
- In text editing, concurrency problems lead to different document state at each user



Concurrency Control

Locking

- Used in many databases and filesystems
- Collaborative editing: Bad user experience

Concurrency Control

- Translate the operations between users (operational transformation)
- Develop an algorithm for updating that avoids conflicts
  - Global identifiers for characters
  - Algorithm to achieve the same character order at all users

Concurrency Control

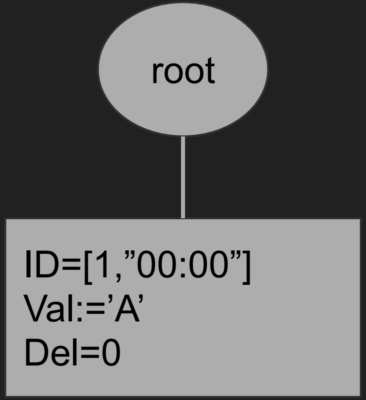
Conflict-Free Replicated Data Types (CRDTs)

- Smart data structure for representing of text
- Text converges to the same state regardless of operation order

CRDT: Tree-based

- ID of a character is user ID + clock(timestamp)
- Parent is the preceding character

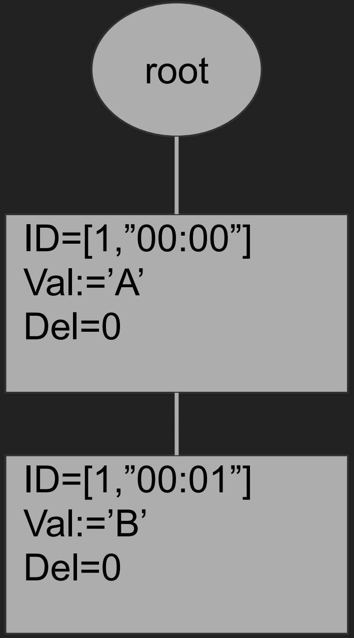
```
{
“Op”:”insert”
“UID”:1
“Clock”:”00:00”
“Value”:’A’
“Parent”:null
}
```



CRDT: Tree-based

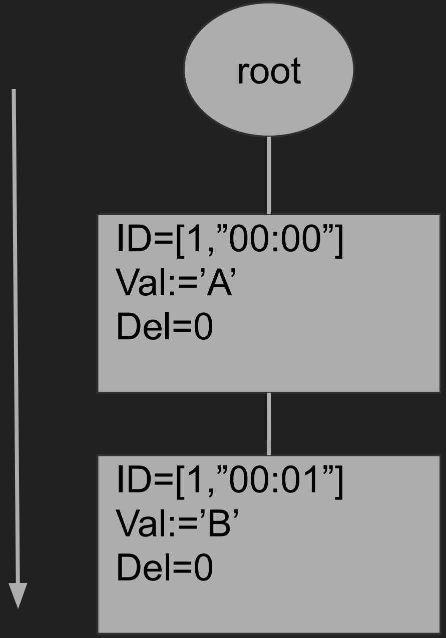
- ID of a character is user ID + clock(timestamp)
- Parent is the preceding character

```
{
"Op":"insert"
"UID":1
"Clock": "00:01"
"Value":'B'
"Parent":
[1,"00:00"]
}
```



CRDT: Ordering

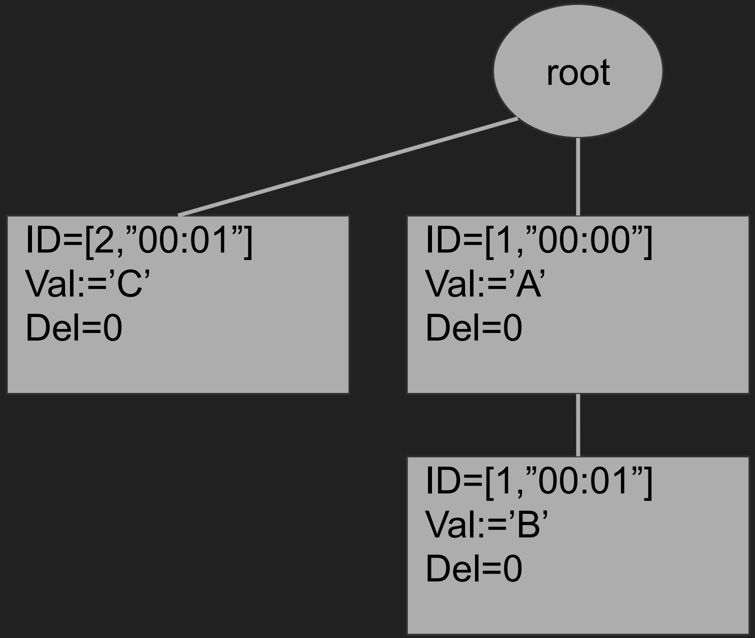
- Traverse the tree depth-first
- Document: AB



CRDT: Ordering

- What if we want to insert before A?
- Two letters will have same parent

```
{
“Op”:”insert”
“UID”:2
“Clock”: “00:01”
“Value”:’C’
“Parent”:null
}
```

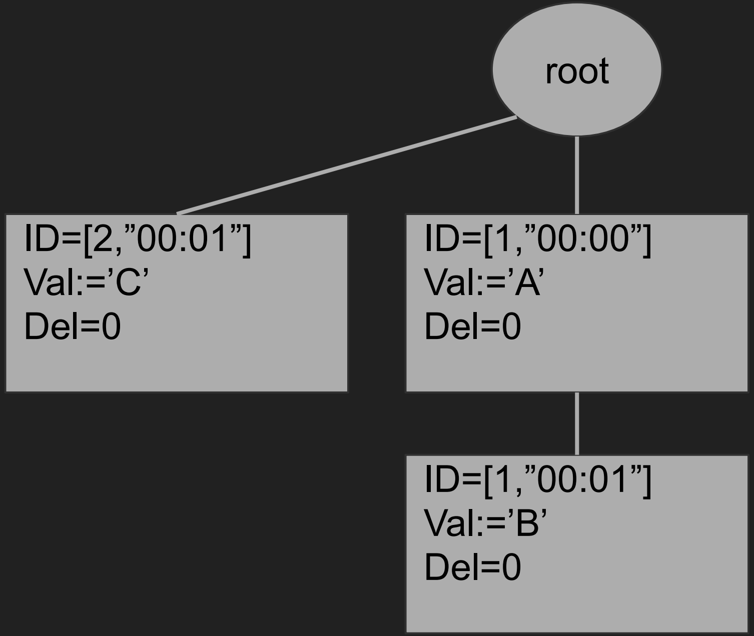


CRDT: Ordering

In case two characters have the same parent:

1. Order descending by clock
2. Order by User ID

- Document: CAB



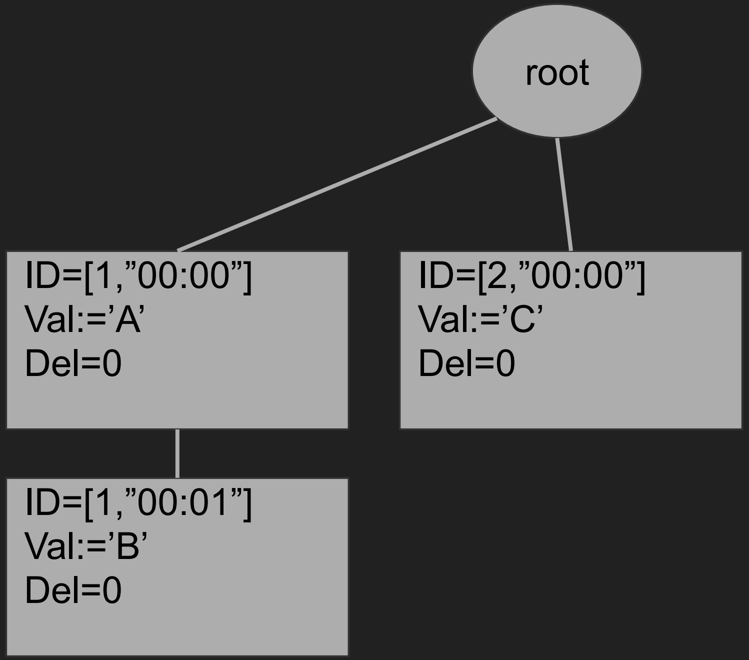
CRDT: Ordering

In case two characters have the same parent:

1. Order descending by timestamp
2. Order by User ID

- Assuming user 1 has higher priority

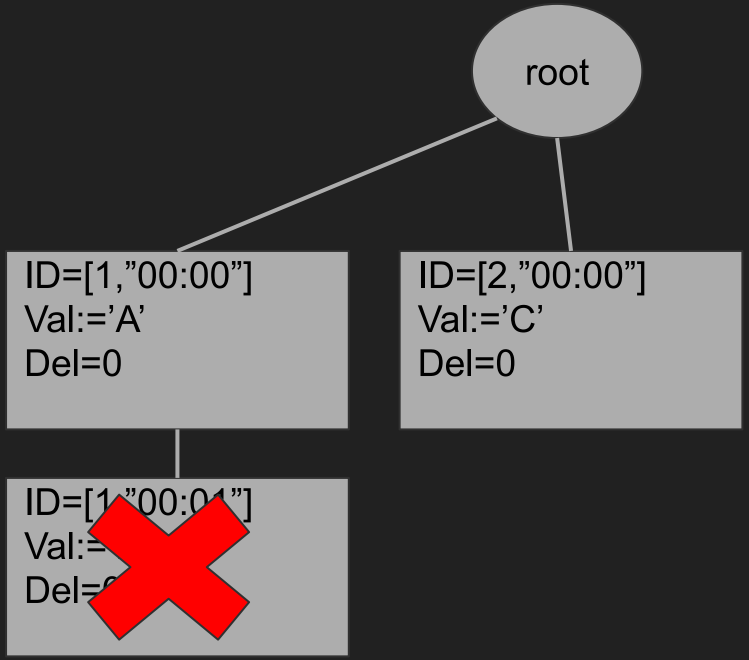
  Document: ABC



CRDT: Tombstones

- What happens if we remove a character from the tree?

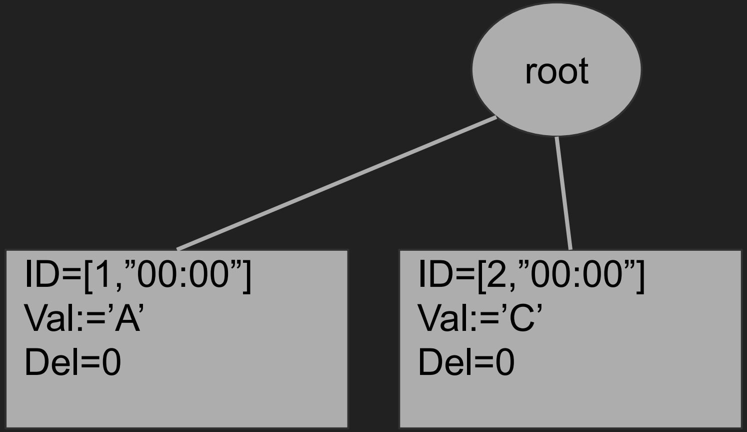
```
{
“Op”:”delete”
“UID”:1
“Clock”: “00:02”
“ID”: [1,”00:01”]
}
```



CRDT: Tombstones

- Deleted character could be the parent of character inserted by another user
- The position of a new character is unknown

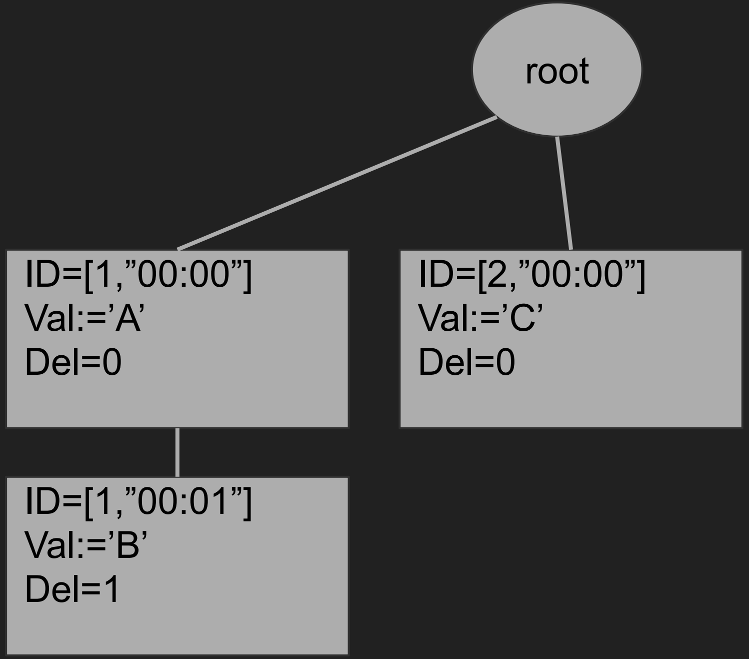
```
{
“Op”:”insert”
“UID”:2
“Clock”: “00:03”
“Value”:’D’
“Parent”:
[1,”00:01”]
}
```



CRDT: Tombstones

- Use a deleted flag
- Skip deleted characters on display/ file export

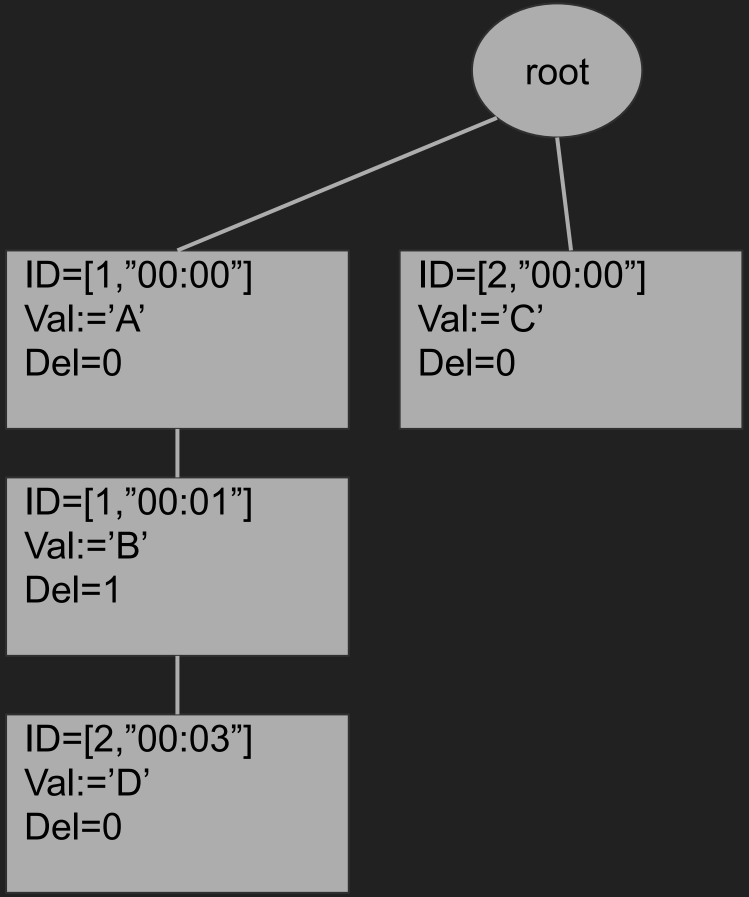
```
{
“Op”:”delete”
“UID”:1
“Clock”: “00:02”
“ID”: [1,”00:01”]
}
```



CRDT: Tombstones

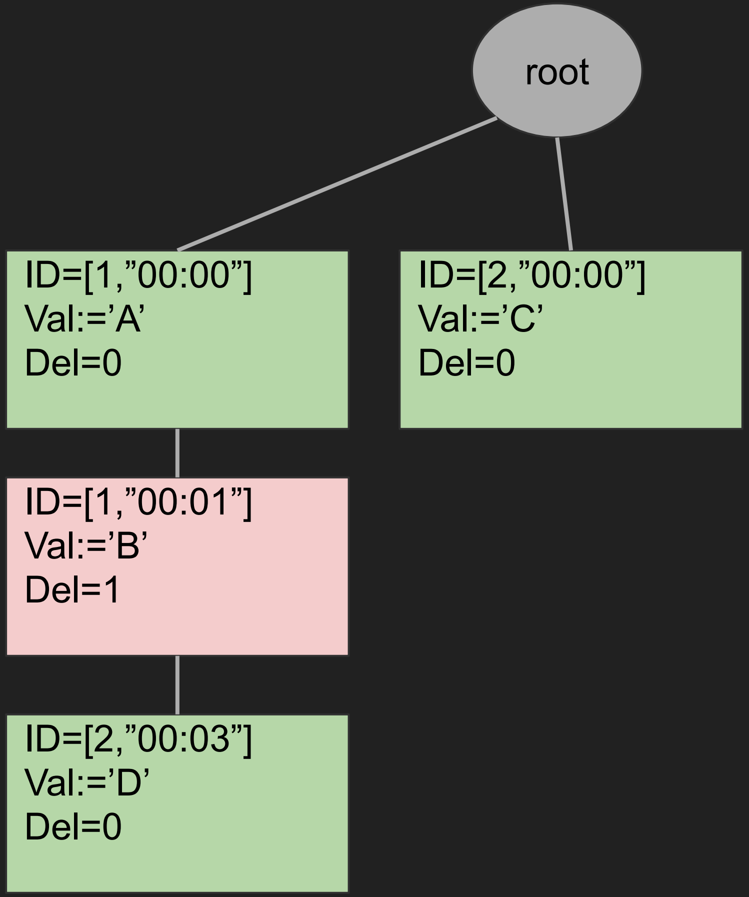
- New characters easily placed

```
{
“Op”:”insert”
“UID”:2
“Clock”: “00:03”
“Value”:’D’
“Parent”:
[1,”00:01”]
}
```



CRDT: Tombstones

- Document: ADC



CRDT: Optimizations

- We have given you the core idea (IDs, insert/delete message format, document ordering technique, tombstones)
- A tree can be represented in many ways
- Use the best data structures to optimize the time needed for insertion, deletion and document generation

Cursor Tracking

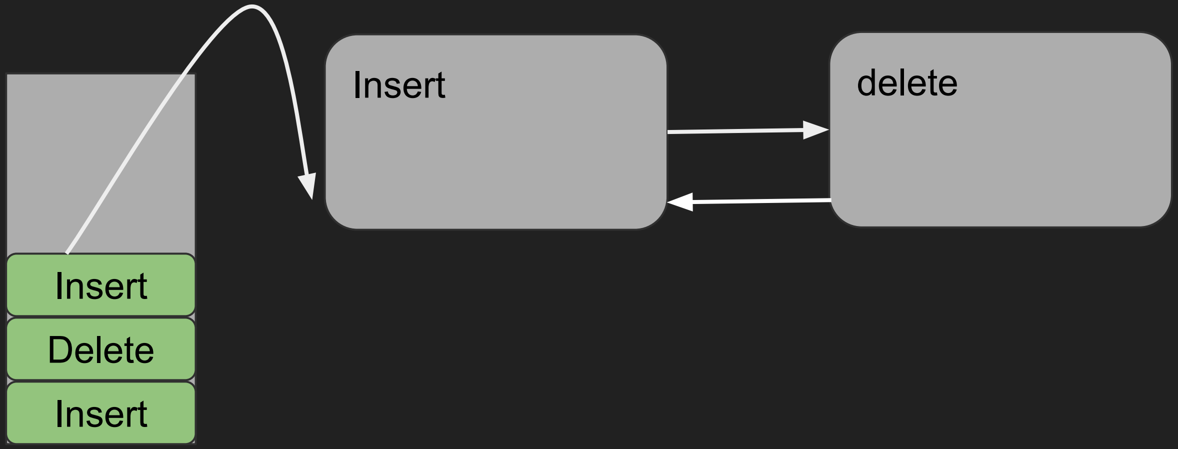
- Send updates to cursor position
- Use the same real-time channel you set up for broadcasting operations
- Think how frequently you should send updates

User Presence

- List active user
- Remove user from active list when:
  a. The user exited the program/session
  b. Network connection drop

Undo/Redo

- Store local operations a stack
- Pop the operation and reverse it
- Reverse operation should **not cause inconsistencies**



Error Handling

- Handle different error sources
- For example
  a. Wrong formatted requests and responses
  b. Wrong collaboration code
  c. Unsupported file format
  d. Duplicate operation -> ignore operation

Project Guidelines

- All logic (including CRDTs) must be implemented in Java
- You may use any library/framework to build the UI
- You **must** build the CRDT by yourself
- You may use libraries for ID generation,handling JSON (e.g. Gson), and networking

Project Guidelines

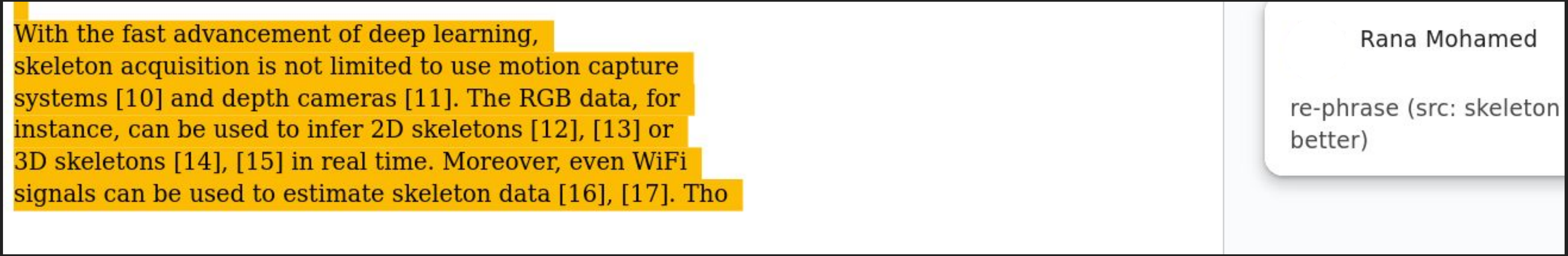
- You should know what are the weak points of your current implementation
- You should justify your design choices
- All team members should have basic knowledge about how the project works
- Team members should have deep knowledge about the part they implemented

Tips

- Write your design choices and the reason behind them
- Start early with CRDT implementation followed by collaboration session management
- CRDTs are mainly used at the client, so choose a UI that easily integrates with Java

Bonus: Comments

- Any user can add/remove comments over a piece of text
- The comment should be visible to all collaborators
- The comment is deleted automatically when the commented text is deleted


With the fast advancement of deep learning, skeleton acquisition is not limited to use motion capture systems [10] and depth cameras [11]. The RGB data, for instance, can be used to infer 2D skeletons [12], [13] or 3D skeletons [14], [15] in real time. Moreover, even WiFi signals can be used to estimate skeleton data [16], [17]. Tho

Rana Mohamed

re-phrase (src: skeleton better)

Bonus: Reconnection

- Allow user to reconnect after WebSocket connection drops
- Server should do the following
  - User is removed from active user list to reconnecting list
  - Give reconnecting users a 5 minutes period before removing them
  - Save operations received during the network drop
  - Send any operations the user missed

Bonus: Reconnection

- Client should do the following
  - Attempt to reconnect
  - Save the local operations done while reconnecting
  - Send local operations after reconnecting
  - Give an error message if you cannot reconnect after 5 minutes

References

- CRDT Demo: https://madebyevan.com/algos/crdt-text-buffer/
- https://www.bartoszsypytkowski.com/operation-based-crdts-protocol/