Develop a web based collaborative editing tool that allows multiple users to edit documents simultaneously in real time. Implement features like cursor synchronization , document versioning , and user presence indicators, use technologies like web sockets for real time communication, react.js for the frontend and node.js with a nosql database like mongodb for the backend.

Below is a basic implementation of a web-based collaborative editing tool with features like cursor synchronization, document versioning, and user presence indicators using React.js for the frontend, Node.js with WebSocket for real-time communication, and MongoDB for the backend.

Backend (Node.js with WebSocket and MongoDB):

1. **Initialize the project**:

```
bash
Copy code
mkdir collaborative-editor-backend
cd collaborative-editor-backend
npm init -y
```

2. **Install dependencies**:

```
bash
Copy code
npm install express socket.io mongoose
```

3. **Create the main server file (`server.js`)**:

```
javascript
Copy code
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
const mongoose = require('mongoose');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

const PORT = process.env.PORT || 5000;

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/collaborative-editor', {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('Error connecting to MongoDB:', err));

// Define MongoDB schema and model
const Document = mongoose.model('Document', {
    content: String,
```

```
    version: Number
});

// WebSocket connection handling
io.on('connection', (socket) => {
    console.log('A user connected: ' + socket.id);

    // Listen for changes to the document
    socket.on('document_change', async (data) => {
        try {
            // Update the document content and version in MongoDB
            await Document.findByIdAndUpdate(data.documentId, { content:
data.content, version: data.version });
            // Broadcast the changes to all other connected users
            socket.broadcast.emit('document_change', data);
        } catch (error) {
            console.error('Error updating document:', error);
        }
    });

    // Handling disconnect
    socket.on('disconnect', () => {
        console.log('User disconnected: ' + socket.id);
    });
});

// Start the server
server.listen(PORT, () => {
    console.log(`Server listening on port ${PORT}`);
});
```

Frontend (React.js):

1. **Initialize the project**:

```
bash
Copy code
npx create-react-app collaborative-editor-frontend
cd collaborative-editor-frontend
```

2. **Install dependencies**:

```
bash
Copy code
npm install socket.io-client
```

3. **Update the main component (`App.js`)**:

```
javascript
Copy code
import React, { useState, useEffect } from 'react';
import io from 'socket.io-client';

const socket = io('http://localhost:5000'); // WebSocket connection
```

```
function App() {
    const [content, setContent] = useState('');
    const [version, setVersion] = useState(0);

    useEffect(() => {
        // Fetch initial document content from the backend
        fetchDocument();

        // Listen for changes to the document from other users
        socket.on('document_change', (data) => {
            if (data.documentId === documentId) {
                setContent(data.content);
                setVersion(data.version);
            }
        });

        return () => {
            socket.disconnect(); // Cleanup WebSocket connection
        };
    }, []);

    const fetchDocument = async () => {
        try {
            const response = await fetch('/api/document');
            const data = await response.json();
            setContent(data.content);
            setVersion(data.version);
        } catch (error) {
            console.error('Error fetching document:', error);
        }
    };

    const handleInputChange = (event) => {
        const newContent = event.target.value;
        setContent(newContent);
        updateDocument(newContent);
    };

    const updateDocument = (newContent) => {
        // Send document changes to the backend
        socket.emit('document_change', { documentId, content: newContent,
version });
    };

    return (
        <div className="App">
            <textarea value={content} onChange={handleInputChange} />
        </div>
    );
}

export default App;
```

4. **Stylesheet (`App.css`)**:

```
css
Copy code
.App {
    text-align: center;
}

textarea {
    width: 80%;
    height: 400px;
    margin: 20px auto;
}
```

MongoDB Setup:

Make sure you have MongoDB installed and running locally on port 27017.

Additional Considerations:

1. Implement user authentication and authorization for secure document editing.
2. Add user presence indicators to show who else is currently editing the document.
3. Implement document versioning to allow users to revert to previous versions of the document.
4. Enhance the frontend with more features like formatting options, real-time cursor synchronization, etc.

This is a basic setup to get you started with building a collaborative editing tool. Depending on your specific requirements, you may need to add more features and handle edge cases accordingly.