

Gatherly - Real-Time WebRTC Video Conferencing

A PROJECT REPORT

Submitted by

Yeshika(23BDA70132)

in partial fulfillment for the award of the degree of

Bachelor of Engineering

IN

Big Data Analyst



Chandigarh University

November 2025



BONAFIDE CERTIFICATE

Certified that this project report "**Gatherly - Real-Time WebRTC Video Conferencing**" is a work of "**Yeshika**" (23BDA70132) who carried out the project under my supervision.

SIGNATURE

Of the Supervisor

SIGNATURE

of the AGM-Technical

Submitted for the project viva-voce

Examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

1. PROJECT DESCRIPTION

1.1 Overview

Gatherly is a fully functional, real-time video conferencing application built on a modern, full-stack architecture. The system demonstrates a practical approach to building a scalable and user-friendly communication platform using Node.js for the backend and React for the frontend. It leverages the power of WebRTC (Web Real-Time Communication) to establish direct, peer-to-peer video and audio streams between clients, ensuring low-latency and high-performance communication. A central signaling server built with Socket.IO orchestrates the connection lifecycle, managing user sessions within distinct rooms and facilitating the initial WebRTC handshake..

1.2 Problem Statement

The primary challenge addressed by this project is the implementation of a real-time, multi-user video communication system from the ground up. While many commercial solutions exist, they often obscure the underlying complexities. This project aims to solve the following key problems:

- Establish direct, low-latency video and audio streams between multiple browsers over the internet.
- Create a mechanism to allow users in the same "room" to discover and connect to one another.
- Design a responsive, intuitive, and feature-rich user interface that provides users with essential meeting controls.
- Implement a clean, scalable project structure with a clear separation between frontend and backend concerns.
- Provide a seamless user flow, from creating a meeting to inviting others via a simple, shareable link

1.3 Main Features

- **Real-Time Video/Audio Communication:** High-quality, low-latency video and audio streaming directly between participants' browsers using WebRTC.
- **Dynamic Room Management:** Users can create new meeting rooms with a unique, randomly generated 6-character ID, or join existing rooms.
- **Shareable Invitation Links:** A core feature is the ability to invite others via a URL. When a user clicks an invite link (e.g., .../?roomId=ABCDEF), they are taken directly to the lobby for that specific meeting.
- **Interactive Lobby:** Before joining a call, users enter a lobby where they can preview their camera, set their display name, and confirm their media devices are working correctly.
- **Responsive Video Gallery:** The main meeting UI features a dynamic grid that automatically arranges participant video tiles in a balanced and aesthetically pleasing layout that adapts to the number of people in the call.
- **Core In-Meeting Controls:** A modern, floating control bar provides users with all essential functions: Mute/Unmute, Start/Stop Camera, and Screen Sharing.
- **Participant List & Hand Raising:** A toggleable side panel displays a real-time list of all attendees. Users can use the "Raise Hand" feature to signal they wish to speak, which displays a visual indicator next to their name.
- **Simple Text Chat:** The application includes a basic, non-persistent text chat in a side panel for in-meeting communication.

1.4 System Architecture

The application is built on a hybrid architecture that is standard for WebRTC applications: a centralized server for signaling and a decentralized, peer-to-peer network for media streaming.

1. **Backend (Signaling Server):** A lightweight Node.js server using Express and Socket.IO. Its only responsibility is to act as a "switchboard." When a user wants to join a room, they connect to this server. The server then informs other users in the room of the new arrival and helps them exchange the necessary information (SDP offers/answers and ICE candidates) to establish a direct connection.
2. **Frontend (Client):** A React Single Page Application (SPA) that runs in the user's browser. It captures the user's camera and microphone, connects to the signaling server, and, most importantly, establishes direct WebRTC RTCPeerConnections with all other participants in the room.
3. **Communication Flow:**
 - All users connect to the signaling server via a persistent WebSocket (Socket.IO) connection.
 - The server manages room logic and relays handshake messages between clients.
 - Once the handshake is complete, all video and audio data is streamed directly between the clients' browsers over an encrypted P2P channel. This minimizes latency and server costs.

1.5 Technology Stack

- **Backend Services**
 - **Node.js:** JavaScript runtime for server-side execution.
 - **Express.js:** Web framework used to create the basic server structure.
 - **Socket.IO:** A library for enabling real-time, bidirectional, event-based communication for signaling.
- **Frontend Application**
 - **React:** A JavaScript library for building component-based, interactive user interfaces.
 - **TypeScript:** Adds static typing to the project for improved code quality, readability, and error detection.
 - **Vite:** A modern, high-performance development server and build tool that provides instant transpilation and hot-reloading.
 - **Tailwind CSS:** A utility-first CSS framework for rapidly styling the user interface.
- **Real-Time Protocol**
 - **WebRTC (Web Real-Time Communication):** The core browser API that enables peer-to-peer streaming of video and audio without the need for plugins.

1.6 Future Enhancements

- **Cloud Deployment:** The most critical next step is to deploy the backend to a Platform-as-a-Service (PaaS) like **Render** and the frontend to a static hosting service like **Netlify** or **Vercel**. This would make the application publicly accessible to anyone on the internet.
- **User Authentication:** Implement a full user registration and login system, likely using JSON Web Tokens (JWTs). This would secure meetings and enable user-specific features like contact lists or saved meeting history.
- **Persistent Chat:** Integrate a NoSQL database like MongoDB or a real-time database like Firebase to store chat messages, allowing users to see the history if they join late or reconnect to a meeting.
- **Configure a TURN Server:** To improve connection reliability, a TURN server could be implemented (e.g., using an open-source project like Coturn or a paid service). This would help users connect even when they are behind highly restrictive corporate or university firewalls.
- **Externalize Configuration:** Remove all hardcoded values like the server URL (`http://localhost:3001`) and manage them using environment variables. This is a standard practice for building portable, "12-factor" applications that can easily be deployed to different environments (like staging or production).

2. HARDWARE/SOFTWARE REQUIREMENTS

2.1 Hardware Requirements

Component	Minimum Specification	Recommended Specification
Processor	Intel i5 / Ryzen 5	Intel Core i5 / AMD Ryzen 5
RAM	8 GB	16 GB
Storage	128 GB SSD	256 GB SSD
Network	5 Mbps	25 Mbps

2.2 Software Requirements

- **Node.js:** Version 18.x (LTS) or higher is recommended for performance and security.
- **npm:** Version 8.x or higher (typically bundled with Node.js).
- **Git:** Version 2.30.0 or higher for source code management.
- **Web Framework:** Express.js
- **Real-time Engine:** Socket.IO
- **UI Library:** React (Version 18 or higher).
- **Language:** TypeScript
- **Project Scaffolding:** Vite
- **Styling:** Tailwind CSS
- **Real-time Client:** Socket.IO Client
- **Real-Time Protocol:** WebRTC (natively supported in modern browsers).
- **Signaling Transport:** WebSockets (managed by Socket.IO).
- **Windows:** Windows 10 or 11.
- **macOS:** macOS 11 (Big Sur) or later.
- **Linux:** Ubuntu 20.04 LTS, Debian 11, Fedora 36 or later distributions.

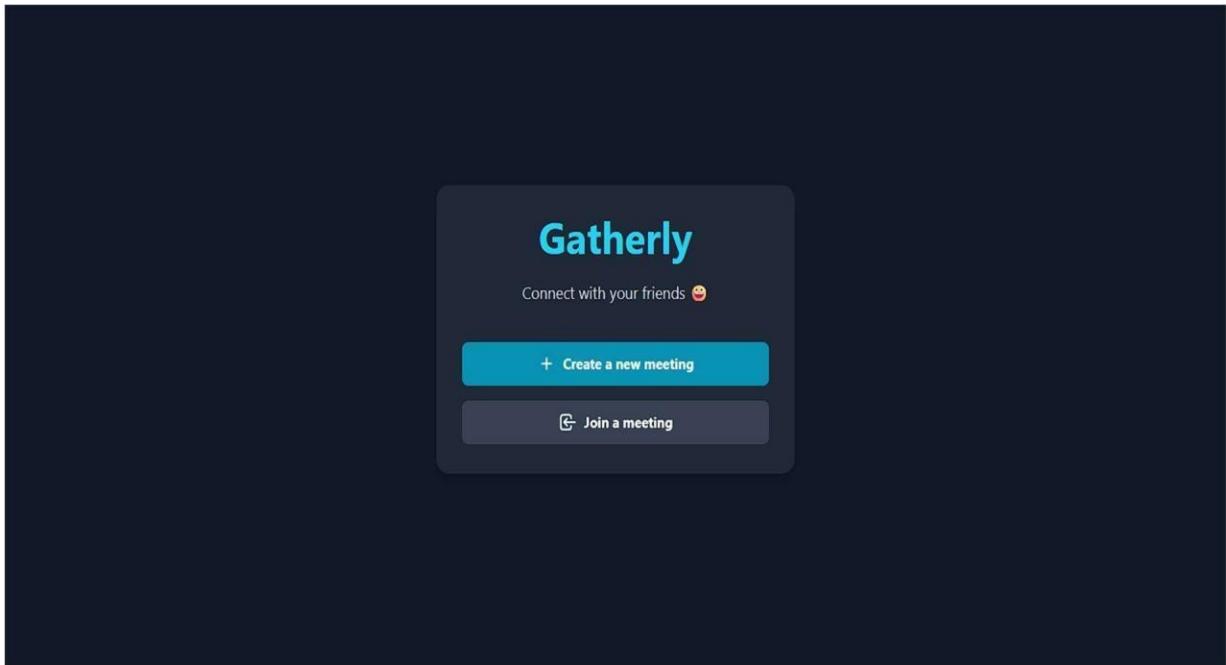
2.3 Deployment Requirements

- **Application Hosting (Backend):** A platform that supports Node.js applications, such as Heroku, Render, or Railway.
- **Static Hosting (Frontend):** A platform for hosting static files, such as Netlify, Vercel, or GitHub Pages.
- **HTTPS:** A valid SSL/TLS certificate is mandatory for WebRTC to function securely in production environments. Services like Let's Encrypt provide free certificates, and most hosting platforms handle this automatically.

3. FRONT-END SCREENS

5.2 Login Page

The Gatherly application does not feature a traditional login system with user accounts. Instead, its entry point is a clean and functional Home Page. This page serves as the main gateway, presenting the user with two clear choices: creating a brand-new meeting or joining a pre-existing one, ensuring a frictionless start to their video conferencing experience. The application deliberately forgoes a traditional user account and login system to maximize accessibility and speed. The Home Page serves as the primary entry point, immediately presenting the user with the application's core purpose. The minimalist design reduces cognitive load.



Key Elements of the User Interface

1. Header and Title

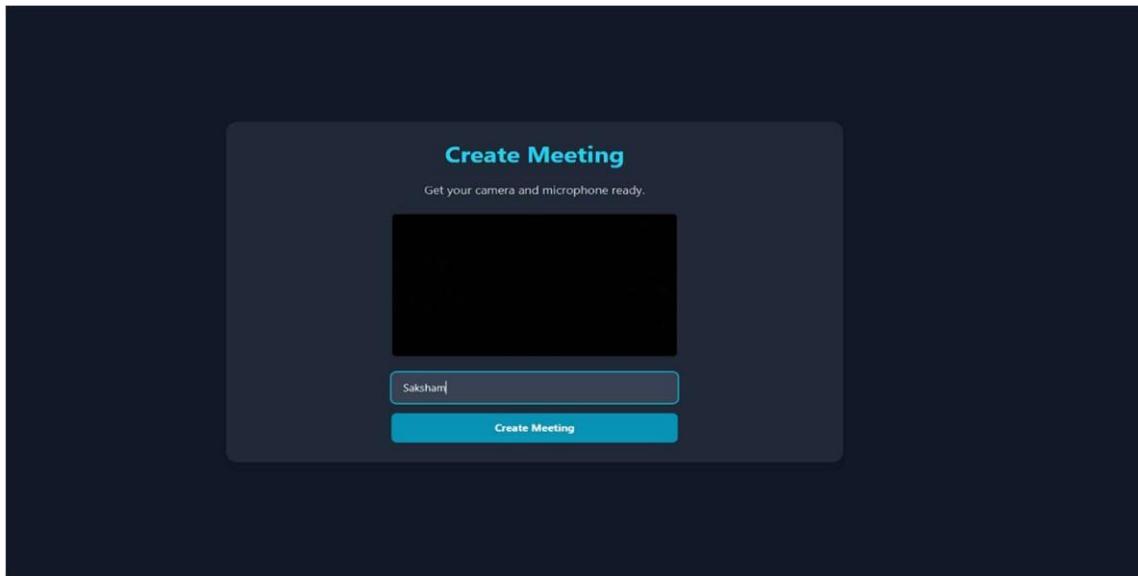
The top of the page features a prominent `<h1>` title, "Gatherly," to establish the application's brand identity. Below it, a friendly and concise subtitle, "Connect with your friends 🎉," communicates the tool's purpose in a welcoming tone. This combination immediately informs the user what the application is and what it does, setting a positive context for their interaction.

2. Action Buttons :

The primary interactive components are two large, clearly labeled buttons. The "Create a new meeting" button is styled as the primary call-to-action with a vibrant color, encouraging users to start a new session. The "Join a meeting" button is styled as a secondary action, providing an equally clear but visually distinct path for users who have been invited to a call.

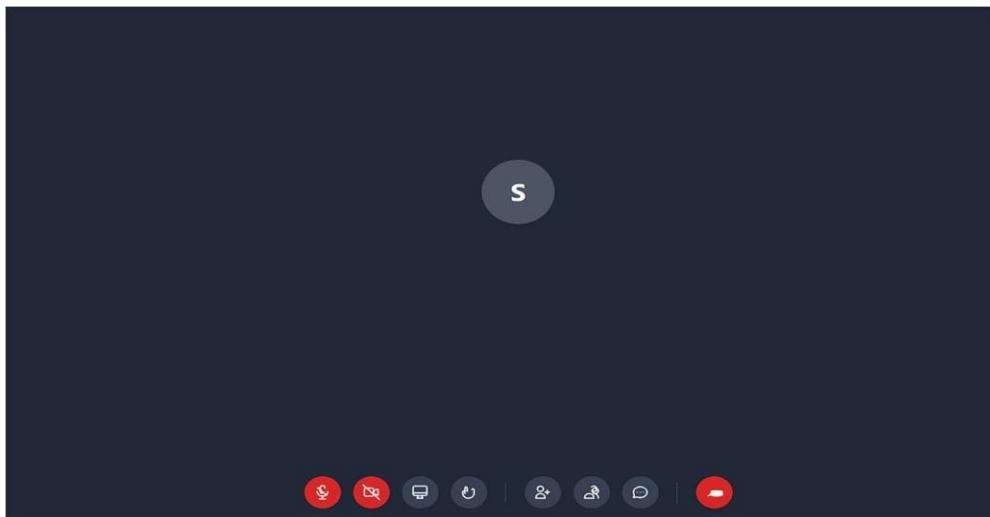
3. Lobby Screen :

This screen acts as an essential intermediate step before a user enters a live call. It contains a real-time video preview for the user to check their appearance and background, a mandatory text input field to set their display name, and a context-aware action button ("Create Meeting" or "Join Meeting") that is only enabled once the user is ready. This ensures a smooth and prepared entry into the meeting.



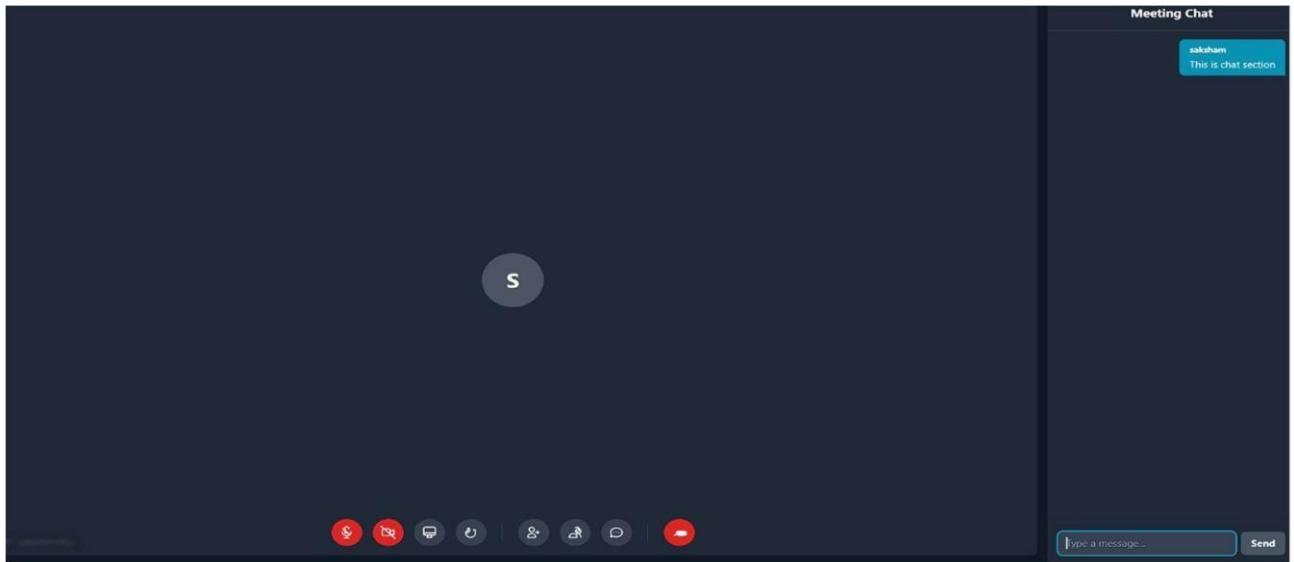
4. Main Meeting Interface :

The core meeting UI is dominated by a dynamic video gallery that responsively arranges participant tiles. A modern, floating Control Bar is overlaid at the bottom of the screen, providing persistent access to critical in-call functions like mute, camera control, screen sharing, and ending the call. This layered design ensures that user controls are always accessible and never obscured by the video content.



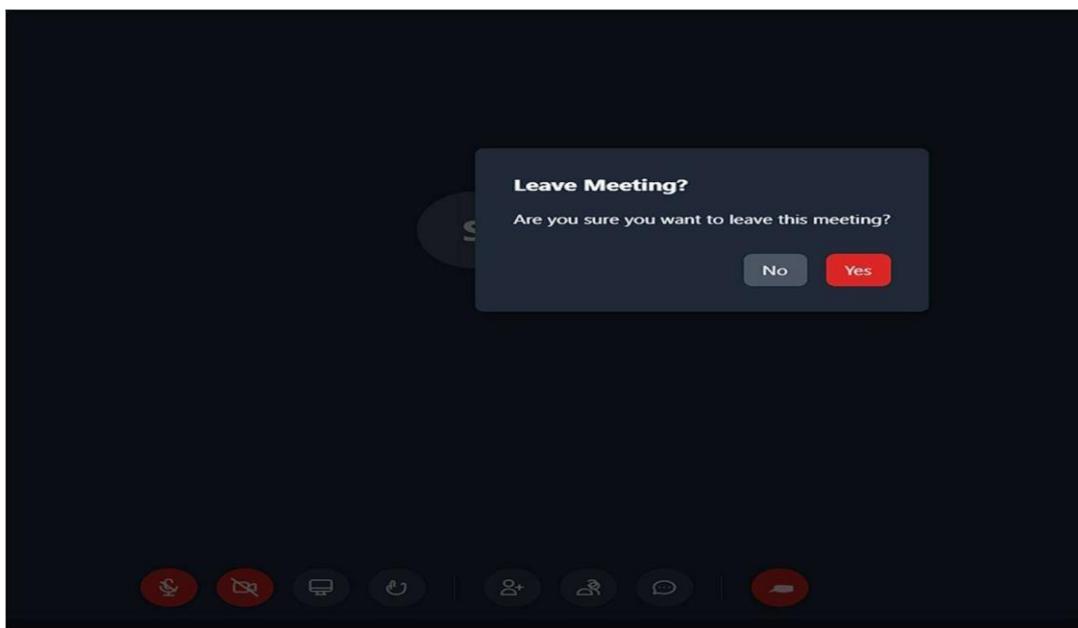
5. Chat Interface:

The application includes a real-time text chat feature, accessible via a button on the control bar. The chat opens in a side panel, allowing users to communicate without interrupting the speaker. It is ideal for sharing links, asking questions, or troubleshooting. The interface clearly distinguishes between messages sent by the user and those from other participants.



6. Leave Confirmation Dialog:

To improve the user experience and prevent accidental departures, the application features a confirmation dialog. When a user clicks the "End Call" button, a modal window appears asking them to confirm their decision. This simple step ensures that leaving a meeting is always an intentional act.



4. LIMITATION & FUTURE SCOPE

The Gatherly project successfully implements a core Minimum Viable Product (MVP) that demonstrates a sophisticated, real-time communication architecture. However, to be considered a fully production-ready system, it has several important limitations that provide a clear roadmap for future development.

4.1 Current Limitations:

1. **No Authentication or Security:** The application is completely open, lacking a user login or authorization system. This means meeting rooms are not private, and any user with a room ID can join. This poses a significant security risk in a production environment as there is no way to control access or verify user identity.
2. **Ephemeral Data and State:** All application data, including chat messages and the server-side room list, is stored in memory. If the server restarts, all active meetings are terminated. Similarly, chat history is lost for any user who refreshes their browser, creating a disjointed user experience.
3. **Limited Network Traversal:** The system relies exclusively on public STUN servers to facilitate peer-to-peer connections. It does not have a TURN server configured, which means users behind certain restrictive corporate or university firewalls (symmetric NATs) may be unable to establish a connection with other peers, leading to call failures.
4. **Basic State Synchronization:** While the core video and audio streams are synchronized, other important user states are not. For instance, when a user mutes their microphone or turns off their camera, this status is not broadcast, so other participants do not see a visual indicator of their status.
5. **Hardcoded Configuration:** Critical configuration values, most notably the backend signaling server URL, are hardcoded directly into the frontend source code. This makes the application inflexible and difficult to deploy to different environments (e.g., from a local machine to a live production server) without manually changing the code.

4.2 Future Scope and Enhancements

1. Implement User Authentication:

The highest priority enhancement is to secure the application by adding a full user registration and login system, likely using JSON Web Tokens (JWTs). This would enable private, user-specific meeting rooms, prevent unauthorized access, and provide a foundation for features like contact lists and meeting history.

2. Deploy to the Cloud:

To make Gatherly publicly accessible, the backend server should be deployed to a Platform-as-a-Service (PaaS) like Render or Heroku. The frontend application should be built and deployed to a static hosting service like Netlify or Vercel. This would allow anyone on the internet to use the application.

3. Configure a TURN Serve:

To dramatically improve connection reliability, a TURN server should be implemented. This could be done by setting up an open-source project like Coturn on a cloud server or by using a paid third-party service. This would ensure users behind restrictive firewalls can still connect to meetings successfully.

4. Add Persistent Chat:

The chat feature can be greatly improved by integrating a real-time, NoSQL database like MongoDB or Firebase. This would allow chat messages to be saved and retrieved, meaning users who join late or reconnect to a meeting could see the entire conversation history, making the feature far more useful.

5. Advanced State Synchronization:

The signaling server's logic should be expanded to broadcast additional user states. This would include events for when a participant mutes/unmutes their microphone, turns their camera on/off, or raises/lowers their hand. The frontend would then display real-time visual indicators for these actions on every participant's tile.

6. Implement Meeting Recording:

A powerful feature would be the ability to record meetings. This could be implemented either client-side, using the browser's MediaRecorder API, or server-side, using a more complex media server. The resulting video files could then be saved to a cloud storage service like AWS S3 for later viewing and sharing.

7. Externalize Configuration:

All hardcoded values, especially the backend URL, should be removed from the source code and managed using environment variables (e.g., via a .env file). This is a standard "12-factor app" practice that makes the application portable, secure, and easy to configure for different environments like development, staging, and production.

8. Speaker Detection Highlighting:

To improve the user experience in calls with many participants, a speaker detection feature could be implemented. By monitoring audio levels from each remote stream, the UI could automatically place a colored border or highlight around the video tile of the person who is currently speaking, making it easier to follow the conversation.

5. GITHUB URL

Project Repository: https://github.com/rakshit2703/fullstack_project-2.git

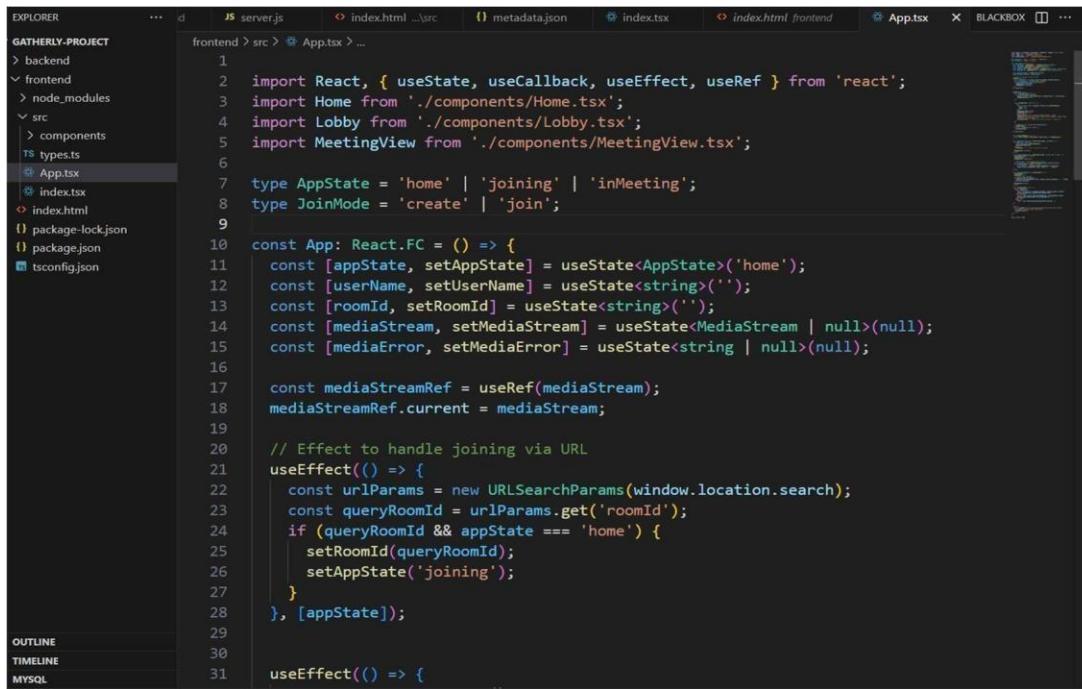
Repository Structure:

The screenshot shows a GitHub repository page for 'fullstack_project-2'. The repository is public. The main branch is 'main'. There is 1 branch and 0 tags. A search bar is at the top right. Buttons for 'Add file' and 'Code' are also present. A recent commit by 'rakshit2703' is listed, which adds files via upload. The commit hash is b0917e9, it was made 4 minutes ago, and it includes 1 commit. The commit message is 'Add files via upload'. The files added are 'README.md', 'backend', and 'frontend', all added 4 minutes ago via upload.

File	Action	Time Ago
README.md	Add files via upload	4 minutes ago
backend	Add files via upload	4 minutes ago
frontend	Add files via upload	4 minutes ago

6. Project Code:

6.1 Frontend:



```
frontend > src > App.tsx > ...
1 import React, { useState, useCallback, useEffect, useRef } from 'react';
2 import Home from './components/Home.tsx';
3 import Lobby from './components/Lobby.tsx';
4 import MeetingView from './components/MeetingView.tsx';
5
6 type AppState = 'home' | 'joining' | 'inMeeting';
7 type JoinMode = 'create' | 'join';
8
9 const App: React.FC = () => {
10   const [appState, setAppState] = useState<AppState>('home');
11   const [userName, setUserName] = useState<string>('');
12   const [roomId, setRoomId] = useState<string>('');
13   const [mediaStream, setMediaStream] = useState<MediaStream | null>(null);
14   const [mediaError, setMediaError] = useState<string | null>(null);
15
16   const mediaStreamRef = useRef(mediaStream);
17   mediaStreamRef.current = mediaStream;
18
19   // Effect to handle joining via URL
20   useEffect(() => {
21     const urlParams = new URLSearchParams(window.location.search);
22     const queryRoomId = urlParams.get('roomId');
23     if (queryRoomId && appState === 'home') {
24       setRoomId(queryRoomId);
25       setAppState('joining');
26     }
27   }, [appState]);
28
29
30   useEffect(() => {
31
32     // ...
33
34     // ...
35
36     // ...
37
38     // ...
39
40     // ...
41
42     // ...
43
44     // ...
45
46     // ...
47
48     // ...
49
50     // ...
51
52     // ...
53
54     // ...
55
56     // ...
57
58     // ...
59
60     // ...
61
62     // ...
63
64     // ...
65
66     // ...
67
68     // ...
69
70     // ...
71
72     // ...
73
74     // ...
75
76     // ...
77
78     // ...
79
79
80     // ...
81
82     // ...
83
84     // ...
85
86     // ...
87
87
88     // ...
89
90     // ...
91
92     // ...
93
93     // ...
94
94
95     // ...
96
97     // ...
98
98
99     // ...
100
101
102     // ...
103
104     // ...
105
106     // ...
107
107
108     // ...
109
109
110     // ...
111
112     // ...
113
113
114     // ...
115
115
116     // ...
117
117
118     // ...
119
119
120     // ...
121
121
122     // ...
123
123
124     // ...
125
125
126     // ...
127
127
128     // ...
129
129
130     // ...
131
131
132     // ...
133
133
134     // ...
135
135
136     // ...
137
137
138     // ...
139
139
140     // ...
141
141
142     // ...
143
143
144     // ...
145
145
146     // ...
147
147
148     // ...
149
149
150     // ...
151
151
152     // ...
153
153
154     // ...
155
155
156     // ...
157
157
158     // ...
159
159
160     // ...
161
161
162     // ...
163
163
164     // ...
165
165
166     // ...
167
167
168     // ...
169
169
170     // ...
171
171
172     // ...
173
173
174     // ...
175
175
176     // ...
177
177
178     // ...
179
179
180     // ...
181
181
182     // ...
183
183
184     // ...
185
185
186     // ...
187
187
187
188     // ...
189
189
190     // ...
191
191
192     // ...
193
193
194     // ...
195
195
196     // ...
197
197
198     // ...
199
199
200     // ...
201
201
202     // ...
203
203
204     // ...
205
205
206     // ...
207
207
208     // ...
209
209
210     // ...
211
211
212     // ...
213
213
214     // ...
215
215
216     // ...
217
217
217
218     // ...
219
219
220     // ...
221
221
222     // ...
223
223
223
224     // ...
225
225
226     // ...
227
227
227
228     // ...
229
229
230     // ...
231
231
232     // ...
233
233
233
234     // ...
235
235
236     // ...
237
237
237
238     // ...
239
239
240     // ...
241
241
242     // ...
243
243
243
244     // ...
245
245
246     // ...
247
247
247
248     // ...
249
249
250     // ...
251
251
252     // ...
253
253
253
254     // ...
255
255
256     // ...
257
257
257
258     // ...
259
259
260     // ...
261
261
262     // ...
263
263
263
264     // ...
265
265
266     // ...
267
267
267
268     // ...
269
269
270     // ...
271
271
272     // ...
273
273
273
274     // ...
275
275
276     // ...
277
277
277
278     // ...
279
279
280     // ...
281
281
282     // ...
283
283
283
284     // ...
285
285
286     // ...
287
287
287
288     // ...
289
289
290     // ...
291
291
292     // ...
293
293
293
294     // ...
295
295
296     // ...
297
297
297
298     // ...
299
299
300     // ...
301
301
302     // ...
303
303
303
304     // ...
305
305
306     // ...
307
307
307
308     // ...
309
309
310     // ...
311
311
312     // ...
313
313
313
314     // ...
315
315
316     // ...
317
317
317
318     // ...
319
319
320     // ...
321
321
322     // ...
323
323
323
324     // ...
325
325
326     // ...
327
327
327
328     // ...
329
329
330     // ...
331
331
332     // ...
333
333
333
334     // ...
335
335
336     // ...
337
337
337
338     // ...
339
339
340     // ...
341
341
342     // ...
343
343
343
344     // ...
345
345
346     // ...
347
347
347
348     // ...
349
349
350     // ...
351
351
352     // ...
353
353
353
354     // ...
355
355
356     // ...
357
357
357
358     // ...
359
359
360     // ...
361
361
362     // ...
363
363
363
364     // ...
365
365
366     // ...
367
367
367
368     // ...
369
369
370     // ...
371
371
372     // ...
373
373
373
374     // ...
375
375
376     // ...
377
377
377
378     // ...
379
379
380     // ...
381
381
382     // ...
383
383
383
384     // ...
385
385
386     // ...
387
387
387
388     // ...
389
389
390     // ...
391
391
392     // ...
393
393
393
394     // ...
395
395
396     // ...
397
397
397
398     // ...
399
399
400     // ...
401
401
402     // ...
403
403
403
404     // ...
405
405
406     // ...
407
407
407
408     // ...
409
409
410     // ...
411
411
412     // ...
413
413
413
414     // ...
415
415
416     // ...
417
417
417
418     // ...
419
419
420     // ...
421
421
422     // ...
423
423
423
424     // ...
425
425
426     // ...
427
427
427
428     // ...
429
429
430     // ...
431
431
432     // ...
433
433
433
434     // ...
435
435
436     // ...
437
437
437
438     // ...
439
439
440     // ...
441
441
442     // ...
443
443
443
444     // ...
445
445
446     // ...
447
447
447
448     // ...
449
449
450     // ...
451
451
452     // ...
453
453
453
454     // ...
455
455
456     // ...
457
457
457
458     // ...
459
459
460     // ...
461
461
462     // ...
463
463
463
464     // ...
465
465
466     // ...
467
467
467
468     // ...
469
469
470     // ...
471
471
472     // ...
473
473
473
474     // ...
475
475
476     // ...
477
477
477
478     // ...
479
479
480     // ...
481
481
482     // ...
483
483
483
484     // ...
485
485
486     // ...
487
487
487
488     // ...
489
489
490     // ...
491
491
492     // ...
493
493
493
494     // ...
495
495
496     // ...
497
497
497
498     // ...
499
499
500     // ...
501
501
502     // ...
503
503
503
504     // ...
505
505
506     // ...
507
507
507
508     // ...
509
509
510     // ...
511
511
512     // ...
513
513
513
514     // ...
515
515
516     // ...
517
517
517
518     // ...
519
519
520     // ...
521
521
522     // ...
523
523
523
524     // ...
525
525
526     // ...
527
527
527
528     // ...
529
529
530     // ...
531
531
532     // ...
533
533
533
534     // ...
535
535
536     // ...
537
537
537
538     // ...
539
539
540     // ...
541
541
542     // ...
543
543
543
544     // ...
545
545
546     // ...
547
547
547
548     // ...
549
549
550     // ...
551
551
552     // ...
553
553
553
554     // ...
555
555
556     // ...
557
557
557
558     // ...
559
559
560     // ...
561
561
562     // ...
563
563
563
564     // ...
565
565
566     // ...
567
567
567
568     // ...
569
569
570     // ...
571
571
572     // ...
573
573
573
574     // ...
575
575
576     // ...
577
577
577
578     // ...
579
579
580     // ...
581
581
582     // ...
583
583
583
584     // ...
585
585
586     // ...
587
587
587
588     // ...
589
589
590     // ...
591
591
592     // ...
593
593
593
594     // ...
595
595
596     // ...
597
597
597
598     // ...
599
599
600     // ...
601
601
602     // ...
603
603
603
604     // ...
605
605
606     // ...
607
607
607
608     // ...
609
609
610     // ...
611
611
612     // ...
613
613
613
614     // ...
615
615
616     // ...
617
617
617
618     // ...
619
619
620     // ...
621
621
622     // ...
623
623
623
624     // ...
625
625
626     // ...
627
627
627
628     // ...
629
629
630     // ...
631
631
632     // ...
633
633
633
634     // ...
635
635
636     // ...
637
637
637
638     // ...
639
639
640     // ...
641
641
642     // ...
643
643
643
644     // ...
645
645
646     // ...
647
647
647
648     // ...
649
649
650     // ...
651
651
652     // ...
653
653
653
654     // ...
655
655
656     // ...
657
657
657
658     // ...
659
659
660     // ...
661
661
662     // ...
663
663
663
664     // ...
665
665
666     // ...
667
667
667
668     // ...
669
669
670     // ...
671
671
672     // ...
673
673
673
674     // ...
675
675
676     // ...
677
677
677
678     // ...
679
679
680     // ...
681
681
682     // ...
683
683
683
684     // ...
685
685
686     // ...
687
687
687
688     // ...
689
689
690     // ...
691
691
692     // ...
693
693
693
694     // ...
695
695
696     // ...
697
697
697
698     // ...
699
699
700     // ...
701
701
702     // ...
703
703
703
704     // ...
705
705
706     // ...
707
707
707
708     // ...
709
709
710     // ...
711
711
712     // ...
713
713
713
714     // ...
715
715
716     // ...
717
717
717
718     // ...
719
719
720     // ...
721
721
722     // ...
723
723
723
724     // ...
725
725
726     // ...
727
727
727
728     // ...
729
729
730     // ...
731
731
732     // ...
733
733
733
734     // ...
735
735
736     // ...
737
737
737
738     // ...
739
739
740     // ...
741
741
742     // ...
743
743
743
744     // ...
745
745
746     // ...
747
747
747
748     // ...
749
749
750     // ...
751
751
752     // ...
753
753
753
754     // ...
755
755
756     // ...
757
757
757
758     // ...
759
759
760     // ...
761
761
762     // ...
763
763
763
764     // ...
765
765
766     // ...
767
767
767
768     // ...
769
769
770     // ...
771
771
772     // ...
773
773
773
774     // ...
775
775
776     // ...
777
777
777
778     // ...
779
779
780     // ...
781
781
782     // ...
783
783
783
784     // ...
785
785
786     // ...
787
787
787
788     // ...
789
789
790     // ...
791
791
792     // ...
793
793
793
794     // ...
795
795
796     // ...
797
797
797
798     // ...
799
799
800     // ...
801
801
802     // ...
803
803
803
804     // ...
805
805
806     // ...
807
807
807
808     // ...
809
809
810     // ...
811
811
812     // ...
813
813
813
814     // ...
815
815
816     // ...
817
817
817
818     // ...
819
819
820     // ...
821
821
822     // ...
823
823
823
824     // ...
825
825
826     // ...
827
827
827
828     // ...
829
829
830     // ...
831
831
832     // ...
833
833
833
834     // ...
835
835
836     // ...
837
837
837
838     // ...
839
839
840     // ...
841
841
842     // ...
843
843
843
844     // ...
845
845
846     // ...
847
847
847
848     // ...
849
849
850     // ...
851
851
852     // ...
853
853
853
854     // ...
855
855
856     // ...
857
857
857
858     // ...
859
859
860     // ...
861
861
862     // ...
863
863
863
864     // ...
865
865
866     // ...
867
867
867
868     // ...
869
869
870     // ...
871
871
872     // ...
873
873
873
874     // ...
875
875
876     // ...
877
877
877
878     // ...
879
879
880     // ...
881
881
882     // ...
883
883
883
884     // ...
885
885
886     // ...
887
887
887
888     // ...
889
889
890     // ...
891
891
892     // ...
893
893
893
894     // ...
895
895
896     // ...
897
897
897
898     // ...
899
899
900     // ...
901
901
902     // ...
903
903
903
904     // ...
905
905
906     // ...
907
907
907
908     // ...
909
909
910     // ...
911
911
912     // ...
913
913
913
914     // ...
915
915
916     // ...
917
917
917
918     // ...
919
919
920     // ...
921
921
922     // ...
923
923
923
924     // ...
925
925
926     // ...
927
927
927
928     // ...
929
929
930     // ...
931
931
932     // ...
933
933
933
934     // ...
935
935
936     // ...
937
937
937
938     // ...
939
939
940     // ...
941
941
942     // ...
943
943
943
944     // ...
945
945
946     // ...
947
947
947
948     // ...
949
949
950     // ...
951
951
952     // ...
953
953
953
954     // ...
955
955
956     // ...
957
957
957
958     // ...
959
959
960     // ...
961
961
962     // ...
963
963
963
964     // ...
965
965
966     // ...
967
967
967
968     // ...
969
969
970     // ...
971
971
972     // ...
973
973
973
974     // ...
975
975
976     // ...
977
977
977
978     // ...
979
979
980     // ...
981
981
982     // ...
983
983
983
984     // ...
985
985
986     // ...
987
987
987
988     // ...
989
989
990     // ...
991
991
992     // ...
993
993
993
994     // ...
995
995
996     // ...
997
997
997
998     // ...
999
999
1000     // ...
1001
1001
1002     // ...
1003
1003
1003
1004     // ...
1005
1005
1006     // ...
1007
1007
1007
1008     // ...
1009
1009
1010     // ...
1011
1011
1012     // ...
1013
1013
1013
1014     // ...
1015
1015
1016     // ...
1017
1017
1017
1018     // ...
1019
1019
1020     // ...
1021
1021
1022     // ...
1023
1023
1023
1024     // ...
1025
1025
1026     // ...
1027
1027
1027
1028     // ...
1029
1029
1030     // ...
1031
1031
1032     // ...
1033
1033
1033
1034     // ...
1035
1035
1036     // ...
1037
1037
1037
1038     // ...
1039
1039
1040     // ...
1041
1041
1042     // ...
1043
1043
1043
1044     // ...
1045
1045
1046     // ...
1047
1047
1047
1048     // ...
1049
1049
1050     // ...
1051
1051
1052     // ...
1053
1053
1053
1054     // ...
1055
1055
1056     // ...
1057
1057
1057
1058     // ...
1059
1059
1060     // ...
1061
1061
1062     // ...
1063
1063
1063
1064     // ...
1065
1065
1066     // ...
1067
1067
1067
1068     // ...
1069
1069
1070     // ...
1071
1071
1072     // ...
1073
1073
1073
1074     // ...
1075
1075
1076     // ...
1077
1077
1077
1078     // ...
1079
1079
1080     // ...
1081
1081
1082     // ...
1083
1083
1083
1084     // ...
1085
1085
1086     // ...
1087
1087
1087
1088     // ...
1089
1089
1090     // ...
1091
1091
1092     // ...
1093
1093
1093
1094     // ...
1095
1095
1096     // ...
1097
1097
1097
1098     // ...
1099
1099
1100     // ...
1101
1101
1102     // ...
1103
1103
1103
1104     // ...
1105
1105
1106     // ...
1107
1107
1107
1108     // ...
1109
1109
1110     // ...
1111
1111
1112     // ...
1113
1113
1113
1114     // ...
1115
1115
1116     // ...
1117
1117
1117
1118     // ...
1119
1119
1120     // ...
1121
1121
1122     // ...
1123
1123
1123
1124     // ...
1125
1125
1126     // ...
1127
1127
1127
1128     // ...
1129
1129
1130     // ...
1131
1131
1132     // ...
1133
1133
1133
1134     // ...
1135
1135
1136     // ...
1137
1137
1137
1138     // ...
1139
1139
1140     // ...
1141
1141
1142     // ...
1143
1143
1143
1144     // ...
1145
1145
1146     // ...
1147
1147
1147
1148     // ...
1149
1149
1150     // ...
1151
1151
1152     // ...
1153
1153
1153
1154     // ...
1155
1155
1156     // ...
1157
1157
1157
1158     // ...
1159
1159
1160     // ...
1161
1161
1162     // ...
1163
1163
1163
1164     // ...
1165
1165
1166     // ...
1167
1167
1167
1168     // ...
1169
1169
1170     // ...
1171
1171
1172     // ...
1173
1173
1173
1174     // ...
1175
1175
1176     // ...
1177
1177
1177
1178     // ...
1179
1179
1180     // ...
1181
1181
1182     // ...
1183
1183
1183
1184     // ...
1185
1185
1186     // ...
1187
1187
1187
1188     // ...
1189
1189
1190     // ...
1191
1191
1192     // ...
1193
1193
1193
1194     // ...
1195
1195
1196     // ...
1197
1197
1197
1198     // ...
1199
1199
1200     // ...
1201
1201
1202     // ...
1203
1203
1203
1204     // ...
1205
1205
1206     // ...
1207
1207
1207
1208     // ...
1209
1209
1210     // ...
1211
1211
1212     // ...
1213
1213
1213
1214     // ...
1215
1215
1216     // ...
1217
1217
1217
1218     // ...
1219
1219
1220     // ...
1221
1221
1222     // ...
1223
1223
1223
1224     // ...
1225
1225
1226     // ...
1227
1227
1227
1228     // ...
1229
1229
1230     // ...
1231
1231
1232     // ...
1233
1233
1233
1234     // ...
1235
1235
1236     // ...
1237
1237
1237
1238     // ...
1239
1239
1240     // ...
1241
1241
1242     // ...
1243
1243
1243
1244     // ...
1245
1245
1246     // ...
1247
1247
1247
1248     // ...
1249
1249
1250     // ...
1251
1251
1252     // ...
1253
1253
1253
1254     // ...
1255
1255
1256     // ...
1257
1257
1257
1258     // ...
1259
1259
1260     // ...
1261
1261
1262     // ...
1263
1263
1263
1264     // ...
1265
1265
1266     // ...
1267
1267
1267
1268     // ...
1269
1269
1270     // ...
1271
1271
1272     // ...
1273
1273
1273
1274     // ...
1275
1275
1276     // ...
1277
1277
1277
1278     // ...
1279
1279
1280     // ...
1281
1281
1282     // ...
1283
1283
1283
1284     // ...
1285
1285
1286     // ...
1287
1287
1287
1288     // ...
1289
1289
1290     // ...
1291
1291
1292     // ...
1293
1293
1293
1294     // ...
1295
1295
1296     // ...
1297
1297
1297
1298     // ...
1299
1299
1300     // ...
1301
1301
1302     // ...
1303
1303
1303
1304     // ...
1305
1305
1306     // ...
1307
1307
1307
1308     // ...
1309
1309
1310     // ...
1311
1311
1312     // ...
1313
1313
1313
1314     // ...
1315
1315
1316     // ...
1317
1317
1317
1318     // ...
1319
1319
1320     // ...
1321
1321
1322     // ...
1323
1323
1323
1324     // ...
1325
1325
1326     // ...
1327
1327
1327
1328     // ...
1329
1329
1330     // ...
1331
1331
1332     // ...
1333
1333
1333
1334     // ...
1335
1335
1336     // ...
1337
1337
1337
1338     // ...
1339
1339
1340     // ...
1341
1341
1342     // ...
1343
1343
1343
1344     // ...
1345
1345
1346     // ...
1347
1347
1347
1348     // ...
1349
1349
1350     // ...
1351
1351
1352     // ...
1353
1353
1353
1354     // ...
1355
1355
1356     // ...
1357
1357
1357
1358     // ...
1359
1359
1360     // ...
1361
1361
1362     // ...
1363
1363
1363
1364     // ...
1365
1365
1366     // ...
1367
1367
1367
1368     // ...
1369
1369
1370     // ...
1371
1371
1372     // ...
1373
1373
1373
1374     // ...
1375
1375
1376     // ...
1377
1377
1377
1378     // ...
1379
1379
1380     // ...
1381

```

6.2 Backend:

EXPLORER

GATHERLY-PROJECT

backend

frontend

App.tsx

types.ts

Home.tsx

Lobby.tsx

MeetingView.tsx

VideoTile.tsx

ControlBar

frontend > src > App.tsx > App > useEffect() callback

```
1
2 import React, { useState, useCallback, useEffect, useRef } from 'react';
3 import Home from './components/Home.tsx';
4 import Lobby from './components/Lobby.tsx';
5 import MeetingView from './components/MeetingView.tsx';
6
7 type AppState = 'home' | 'joining' | 'inMeeting';
8 type JoinMode = 'create' | 'join';
9
10 const App: React.FC = () => {
11   const [appState, setAppState] = useState<AppState>('home');
12   const [userName, setUserName] = useState<string>('');
13   const [roomId, setRoomId] = useState<string>('');
14   const [mediaStream, setMediaStream] = useState<MediaStream | null>(null);
15   const [mediaError, setMediaError] = useState<string | null>(null);
16
17   const mediaStreamRef = useRef(mediaStream);
18   mediaStreamRef.current = mediaStream;
19 }
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

found 0 vulnerabilities

C:\Users\Saksham Gupta\Desktop\gatherly-project\backend>npm start

> gatherly-server@1.0.0 start

> node server.js

Signaling server listening on port 3001

A user connected: JmY2mGIjv_0WYHS6AAC

User Saksham (JmY2mGIjv_0WYHS6AAC) joined room BPZLNH

REFERENCES

1. **React Documentation.** (2023). *Meta Platforms, Inc.* Retrieved from <https://react.dev/>
(Primary reference for building all user interface components, managing state with Hooks, and structuring the single-page application.)
2. **Node.js Documentation.** (2023). *Node.js Foundation.* Retrieved from <https://nodejs.org/en/docs/>
(Consulted for the JavaScript runtime environment, core modules like http, and for setting up the backend signaling server.)
3. **WebRTC API Documentation.** (2023). MDN Web Docs. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API (The foundational resource for understanding the core peer-to-peer communication APIs, including RTCPeerConnection, MediaStream, and the signaling lifecycle.)
4. **Socket.IO Documentation.** (2023). *Socket.IO Team.* Retrieved from <https://socket.io/docs/>
(The definitive guide for implementing the real-time, event-based signaling logic on both the Node.js server and the React client.)
5. **Vite Documentation.** (2023). *Evan You & Vite Contributors.* Retrieved from <https://vitejs.dev/>
(Used extensively for setting up the modern frontend development environment, configuring the dev server, and bundling the application.)
6. **Tailwind CSS Documentation.** (2023). *Tailwind Labs Inc.* Retrieved from <https://tailwindcss.com/docs/> (The primary resource for all styling, utilizing its utility-first classes to rapidly build the responsive and modern user interface for the application.)

BIBLIOGRAPHY

1. **MDN Web Docs. (n.d.). WebRTC API. Mozilla.**
Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API (An essential resource for the low-level technical details of the WebRTC protocol. This was the primary reference for understanding the RTCPeerConnection API, the signaling state machine, and the methods for capturing and attaching media streams.)
2. **Fowler, M. (2014, March 25). Microservices. martinfowler.com.**
Retrieved from <https://martinfowler.com/articles/microservices.html>
(A seminal article defining the principles of microservices architecture. Although Gatherly's backend is a single service, this resource provided the foundational concept of separating the signaling logic into a distinct, single-responsibility service, decoupled from the frontend clients.)
3. **Socket.IO Documentation. (n.d.). Socket.IO.**
Retrieved from <https://socket.io/docs/>
(The official documentation for the WebSocket library used for signaling. This was the definitive guide for implementing the event-based communication between the server and clients, including room management, broadcasting events, and handling connections.)
4. **Google Developers. (n.d.). Real-time communication with WebRTC. web.dev.**
Retrieved from <https://web.dev/articles/webrtc-basics>
(Consulted for practical examples and simplified explanations of the core WebRTC concepts, particularly the signaling process and the roles of STUN/TURN servers in establishing a peer-to-peer connection.)
5. **Stack Overflow. (Community resource).**
Various threads related to "webrtc react hooks", "socket.io connection", "vite failed to resolve import", "webrtc multiple peers".
(An indispensable resource for troubleshooting specific technical errors, React Hook implementation patterns for managing media streams, Vite configuration issues, and common hurdles encountered when scaling from a two-peer to a multi-peer connection.)
6. **Grigorik, I. (2013). High Performance Browser Networking. O'Reilly Media.**
Retrieved from <https://hpbn.co/webrtc/>
(An in-depth resource that provides a deep understanding of the networking principles behind WebRTC, including the roles of SDP, ICE, STUN, and TURN in navigating network address translators (NATs) to establish reliable peer connections.)