Project Report on

"Online Meeting platform"

Is submitted partial fulfillment of the requirement of

T.Y.B.Sc. (Computer Science)

Under the guidance of

Mrs. Bhavna Patil

SUBMITTED BY

Gupta Arun Raju

PRN no. 2020016400342801

Padmashri Annasaheb Jadhav Bharatiya Samaj Unnati Mandal's

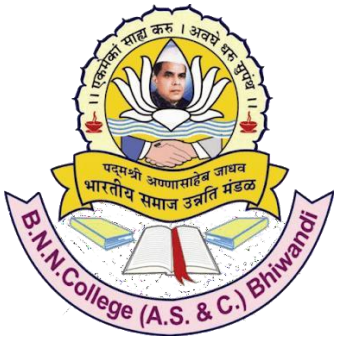
B. N. N. College of Arts, Science & Commerce, Bhiwandi.

(Self Funded Courses)

Department of computer science

University Of Mumbai

2022 - 2023



Padmashri Annasaheb Jadhav Bharatiya Samaj Unnati Mandal's

B. N. N. College of Arts, Science & Commerce, Bhiwandi.

(Self-Funded Courses)

(Department of Computer Science)

CERTIFICATE

This is to certify that **Mr. Gupta Arun Raju** has successfully completed the project titled "**Online Meeting platfom**" and duly submitted the project in partial fulfilment of the "B.Sc. (Computer Science)" degree from the University of Mumbai during the academic year 2022-23.

It is further certified that he has completed all the required phases of project.

Project Guide	External Examiner	Signature of HOD	Signature Principal
(Mrs. Bhavna Patil)		**(Prof. Pramod Shewale)**	**(Dr. Ashok D. Wagh)**

DECLARATION BY THE STUDENT

I, **Gupta Arun Raju** student of T.Y. B.Sc. (Computer Science) hereby declare that the project for the Computer Science, "**Online Meeting platform**" submitted by me for Semester-VI during the academic year 2022-23, is based on actual work carried out by me under the guidance and supervision of **Mrs. Bhavna Patil**. I further state that this project is originally built by me and not submitted anywhere else for any examination.

_____

Student Signature

EVALUATION CERTIFICATE

This is to certify that the undersigned have assessed and evaluated the project on "**Online Meeting platform**" submitted by "Mr **Gupta Arun Raju**", student of T.Y.B.Sc.(computer science). This project is build by best of my knowledge and has accepted for Assessment.

_____

External Examiner

ACKNOWLEDGEMENT

I would like to extend our heartiest thanks with a deep sense of gratitude and respect to all those who provides me immense help and guidance during my period

I would like to thank my Project Guide Mrs Bhavna Patil for providing a vision about the system. I have been greatly benefited from their regular critical reviews and inspiration throughout my work. I am grateful to them for their guidance, encouragement, understanding and insightful support in the development process. I would also like to thank my college for giving required resources whenever I wanted and for giving the opportunity to develop the project.

I would like to express my sincere thanks to our Principal **Dr. Ashok D. Wagh** and our Head of Department **Mr. Pramod Lala Shewale** for having facilitated us with the essential infrastructure & resources without which this project would not have seen light of the day.

With sincere regards,
Gupta Arun Raju

ONLINE MEETING PLATFORM

ABSTRACT

The aim of this project is to develop an online meeting platform that enables individuals and businesses to connect and collaborate remotely in a secure and efficient manner. The platform will feature a user-friendly interface that allows participants to easily schedule, join and manage meetings from any device, anywhere in the world. It will include a range of communication tools such as video conferencing, screen sharing, file sharing, and chat features to facilitate collaboration and enable real-time communication. The platform will prioritize user privacy and security by implementing advanced encryption protocols and access control measures to safeguard confidential information. Overall, this online meeting platform will be an effective and reliable solution for businesses and individuals seeking to conduct meetings and collaborate remotely.

1.Introduction

The increasing demand for remote collaboration and communication has led to the emergence of various online meeting platforms. This project aims to develop a robust online meeting platform that allows users to collaborate and communicate in real-time from anywhere in the world.

Our platform will provide a user-friendly interface that enables users to schedule, join and manage meetings, and share screens and files seamlessly. With this platform, users will be able to communicate effectively, increase productivity, and reduce the time and cost associated with traditional in-person meetings.

In addition, our platform prioritizes user privacy and security. We implement advanced encryption protocols and access control measures to safeguard confidential information. This platform is ideal for businesses, institutions, and individuals who seek to conduct meetings remotely in a secure and efficient manner.

This documentation provides a comprehensive guide on how to use our online meeting platform, how it works, and how it was developed. It is designed for users, developers, and administrators who want to understand the features and functionality of the platform. The document includes user requirements, architecture and design, implementation details, deployment information, and user and administration guides.

Overall, this project aims to provide an effective and reliable solution for remote collaboration and communication that enhances productivity and security for users.

1.2 Purpose

The purpose of this online meeting platform is to provide users with a secure, efficient, and reliable way to collaborate and communicate remotely. The platform aims to enhance productivity by enabling users to conduct meetings, share information, and collaborate in real-time, from anywhere in the world.

The platform is designed to be user-friendly, flexible, and scalable, accommodating a wide range of users and organizations. It prioritizes user privacy and security by implementing advanced encryption protocols and access control measures to safeguard confidential information.

The online meeting platform aims to provide a cost-effective and time-saving solution for businesses, institutions, and individuals who need to collaborate and communicate remotely. It enables users to conduct meetings without the need for in-person meetings, reducing the time and cost associated with travel and logistics.

Overall, the purpose of this online meeting platform is to enhance productivity and security for users who seek to collaborate and communicate remotely. It aims to provide an effective and reliable solution that meets the needs of a wide range of users and organizations.

1.3 Goals

To provide users with a secure and reliable platform for remote collaboration and communication.
To enhance productivity by enabling users to conduct meetings and collaborate in real-time, from anywhere in the world.
To reduce the time and cost associated with in-person meetings by providing a cost-effective and time-saving solution.
To accommodate a wide range of users and organizations by providing a flexible and scalable platform.
To prioritize user privacy and security by implementing advanced encryption protocols and access control measures.

1.4 Objectives

To develop a user-friendly interface that allows users to easily schedule, join, and manage meetings.
To implement screen sharing and file sharing features that enable users to collaborate on documents in real-time.
To provide a chat feature that allows users to communicate during meetings.
To ensure the platform is accessible from any device with an internet connection.
To implement advanced encryption protocols and access control measures to safeguard confidential information.
To ensure the platform is scalable and can accommodate a wide range of users and organizations.
To provide comprehensive user and administration guides to ensure ease of use and management.
To conduct testing and quality assurance to ensure the platform is reliable and efficient.
To ensure the platform is deployed and configured properly to ensure maximum performance and scalability.
These goals and objectives provide a clear direction for the development of the online meeting platform. They are designed to ensure the platform is effective, reliable, and user-friendly, and prioritizes user privacy and security.

1.5 Target Audience

Businesses and organizations of all sizes that require remote collaboration and communication solutions.
Educational institutions that need to facilitate remote learning and online classes.
Non-profit organizations that need to conduct remote meetings and collaborations.
Individuals who need to collaborate remotely with colleagues, friends, or family members.
Freelancers and remote workers who need to communicate and collaborate with clients and team members.
Government agencies that need to facilitate remote communication and collaboration.
Healthcare institutions that need to conduct remote consultations and meetings.
Legal firms that need to conduct remote depositions and meetings.
Financial institutions that need to conduct remote meetings and collaborations.
Any other organization or individual that requires a secure and efficient solution for remote collaboration and communication.
Overall, the online meeting platform is designed to be flexible and scalable, accommodating a wide range of users and organizations. It is ideal for anyone who needs to collaborate and communicate remotely, regardless of industry or organization size.

1.6 Scope

The online meeting platform will provide users with a web-based solution for remote collaboration and communication. The platform will allow users to conduct meetings and collaborate in real-time, from anywhere in the world, using any device with an internet connection.

The platform will include a user-friendly interface that allows users to easily schedule, join, and manage meetings. Users will be able to invite others to attend meetings, share their screens, and collaborate on documents in real-time. The platform will also include a chat feature that allows users to communicate during meetings.

The platform will prioritize user privacy and security by implementing advanced encryption protocols and access control measures to safeguard confidential information. The platform will be designed to be scalable and flexible, accommodating a wide range of users and organizations.

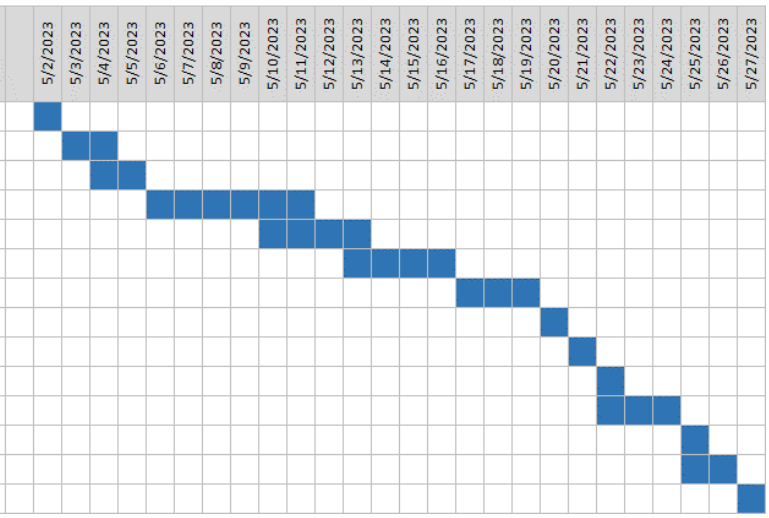
The platform will be developed using the latest web development technologies, including HTML5, CSS3, and JavaScript. It will be compatible with all major web browsers, including Google Chrome, Mozilla Firefox, and Apple Safari. The platform will also be designed to be responsive, providing optimal viewing experiences across a range of devices, including desktop computers, laptops, tablets, and smartphones.

Overall, the online meeting platform will provide an effective and reliable solution for remote collaboration and communication, prioritizing user privacy and security, and accommodating a wide range of users and organizations.

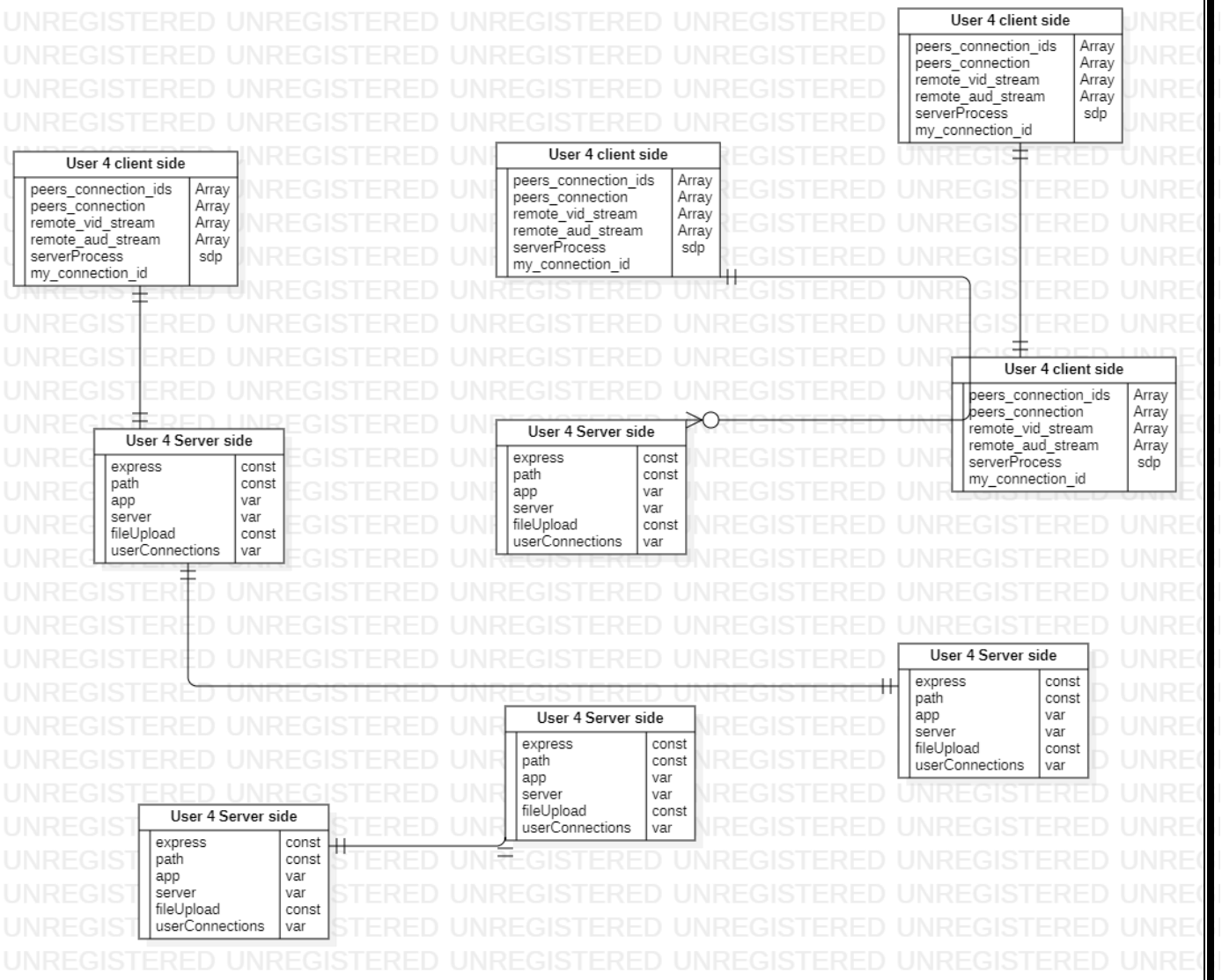
1.7 Gannt Chart

online meeting platform | **Gupta Arun Raju** (Sem 6 Project) | **05-Dec-22** START DATE | **10-Mar-23** END DATE

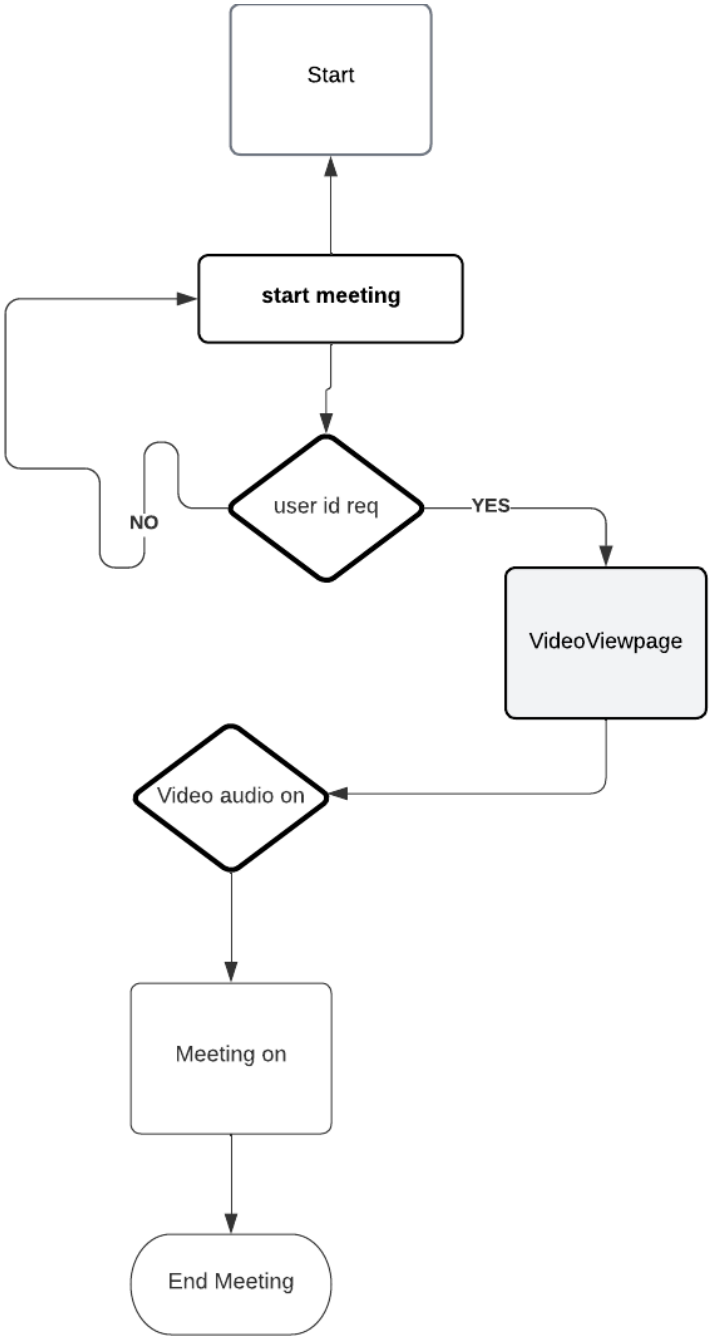
Task ID	Task Name	Start Date	End Date	Duration (In Days)
	Planning Phase	5/2/2023	5/2/2023	1
	System Analysis	5/3/2023	5/4/2023	2
	Requirements gathering	5/4/2023	5/5/2023	2
	Functional and Non-Functior	5/6/2023	5/11/2023	6
	Technology Selection and An	5/10/2023	5/13/2023	4
	Design and Architecture Phas	5/13/2023	5/16/2023	4
	User Interface	5/17/2023	5/19/2023	3
	Front End Development	5/20/2023	5/20/2023	1
	Back End Development	5/21/2023	5/21/2023	1
	Testing and Maintenance Pha	5/22/2023	5/22/2023	1
	Functional testing	5/22/2023	5/24/2023	3
	Non-functional testing	5/25/2023	5/25/2023	1
	deployment	5/25/2023	5/26/2023	2
	documentation	5/27/2023	5/27/2023	1



2.1 ER Diagram



2.2 Flow chart



2.3 System Analysis

Use Case Analysis

1. Actor: User Use Cases:

- Schedule Meeting: User creates a new meeting, specifies date, time, duration, attendees, and other details.
- Join Meeting: User joins a meeting by entering a meeting ID or clicking on a web link.
- View Meetings: User views a list of upcoming and past meetings.
- Modify Meeting: User modifies the details of a scheduled meeting.
- Cancel Meeting: User cancels a scheduled meeting.

2. Actor: Administrator Use Cases:

- Manage Users: Administrator manages user accounts, adds or removes users, and modifies user information.
- Manage Meetings: Administrator manages meetings, approves or rejects meeting requests, and modifies meeting details.
- Manage Resources: Administrator manages platform resources, such as storage and bandwidth, to ensure optimal performance.

3. Actor: Meeting Attendee Use Cases:

- Participate in Meeting: Attendee joins a scheduled meeting, can view presentations, ask questions, participate in discussions, and share screens or files.
- View Meeting Recordings: Attendee can access recordings of past meetings for reference.

2.4 Functional Requirements

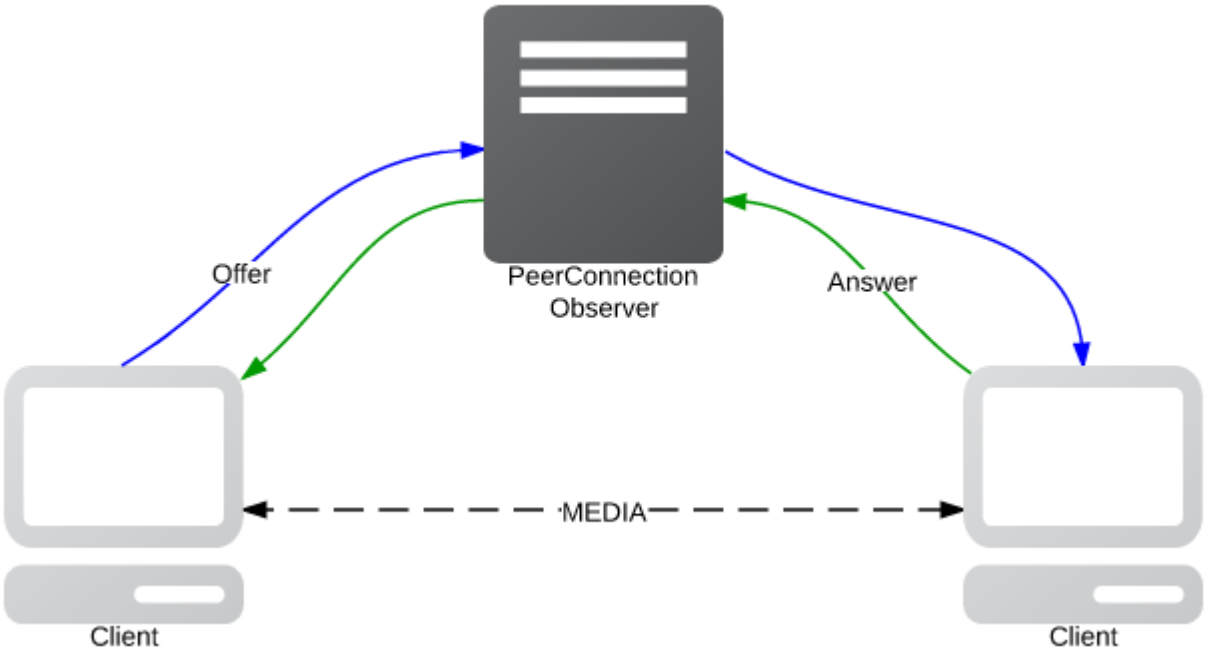
1. User Registration: The platform should allow users to register and create accounts, providing basic information such as name, email address, and password.
2. Scheduling: The platform should allow users to schedule meetings, specifying the date, time, duration, and attendees.
3. Joining: The platform should allow users to join meetings via a web link or meeting ID, with the ability to join via a web browser or mobile app.
4. Screen Sharing: The platform should allow users to share their screens during meetings, with the option to share their entire screen or a specific application window.
5. Chat: The platform should provide a chat feature that allows users to communicate during meetings, with the option to send private messages or messages to the entire group.
6. File Sharing: The platform should allow users to share files and documents during meetings, with the ability to upload and download files in real-time.
7. Recording: The platform should allow users to record meetings for future reference, with the ability to save recordings locally or to the cloud

2.5 Non-Functional Requirements

1. Performance: The platform should be able to handle a large number of users and meetings simultaneously, without experiencing any lag or downtime.
2. Security: The platform should implement robust security measures, including encryption protocols and access control mechanisms, to protect user data and prevent unauthorized access.
3. Usability: The platform should have a user-friendly interface that is easy to navigate and understand, with clear instructions and intuitive design.
4. Compatibility: The platform should be compatible with a wide range of devices and operating systems, including desktop computers, laptops, tablets, and smartphones.
5. Accessibility: The platform should be accessible to users with disabilities, with support for assistive technologies and compliance with accessibility guidelines.

   These are just some possible requirements for the online meeting platform, and they may need to be refined and prioritized based on stakeholder feedback and project goals.

3 Desing and architecture



1.1 Peer connection using webrtc

1.2 RTCPeerConnection is a web API in the WebRTC (Web Real-Time Communications) technology stack that enables real-time communication between web browsers. It provides a way for two browsers to establish a peer-to-peer connection and exchange audio, video, and data directly without the need for a centralized server.

1.3 The RTCPeerConnection API is used to create and manage the connection between the two browsers. It handles the signaling and negotiation process to establish a connection, and manages the media streams being sent and received. Once the connection is established, the browsers can send and receive data, such as audio and video streams, using the RTCPeerConnection API.

1.4 Some of the key features of RTCPeerConnection include:

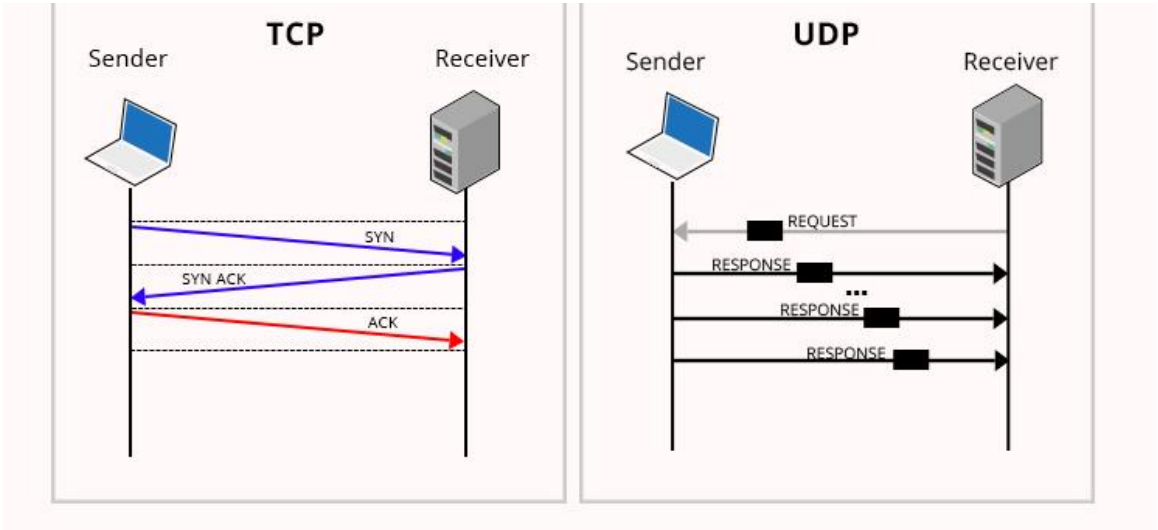
1.5 NAT traversal: RTCPeerConnection uses techniques such as STUN (Session Traversal Utilities for NAT) and ICE (Interactive Connectivity Establishment) to allow communication across network address translators (NATs) and firewalls.

1.6 Encryption: RTCPeerConnection uses SRTP (Secure Real-time Transport Protocol) to encrypt audio and video streams for secure communication.

1.7 Adaptive bitrate: RTCPeerConnection can automatically adjust the bitrate of audio and video streams based on network conditions to maintain optimal quality.

1.8 Signaling: RTCPeerConnection requires signaling to establish a connection between two browsers. This can be done using a signaling server, which can be implemented using a variety of protocols such as WebSocket, HTTP, or even a peer-to-peer network.

1.9 In summary, RTCPeerConnection is a web API that enables real-time communication between web browsers using the WebRTC technology stack. It provides a way for two browsers to establish a peer-to-peer connection and exchange audio, video, and data directly without the need for a centralized server



TCP Vs UDP connection

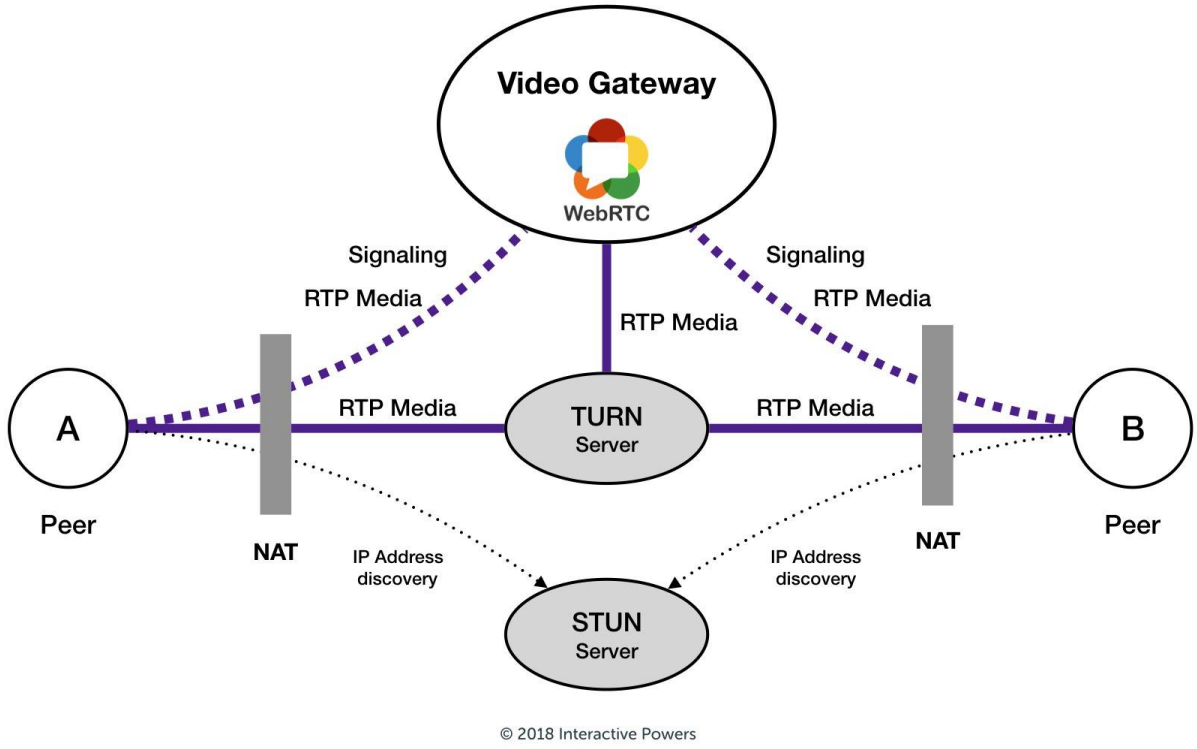
TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are both transport layer protocols in the Internet Protocol (IP) suite, which is the set of protocols that enables communication over the internet.

TCP and UDP have different characteristics and are designed for different purposes.

TCP is a connection-oriented protocol that provides reliable, ordered, and error-checked delivery of data between applications. It establishes a connection between the two endpoints before any data is transmitted, and it ensures that all data is received by the receiver and in the correct order. TCP also includes mechanisms for flow control and congestion control to prevent network congestion.

UDP, on the other hand, is a connectionless protocol that provides unreliable, unordered, and unchecked delivery of data between applications. It does not establish a connection before data transmission and does not provide any mechanisms for error checking, flow control, or congestion control. UDP is faster and less resource-intensive than TCP, making it a good choice for applications that require fast, low-latency transmission of data, such as online gaming, video streaming, and VoIP (Voice over IP) applications.

In summary, TCP is a connection-oriented protocol that provides reliable delivery of data with error checking, flow control, and congestion control, while UDP is a connectionless protocol that provides fast, low-latency delivery of data without any of these mechanisms. Both protocols have their strengths and weaknesses and are used for different purposes depending on the specific needs of the application.



2.1 Session Traversal Utilities for NAT

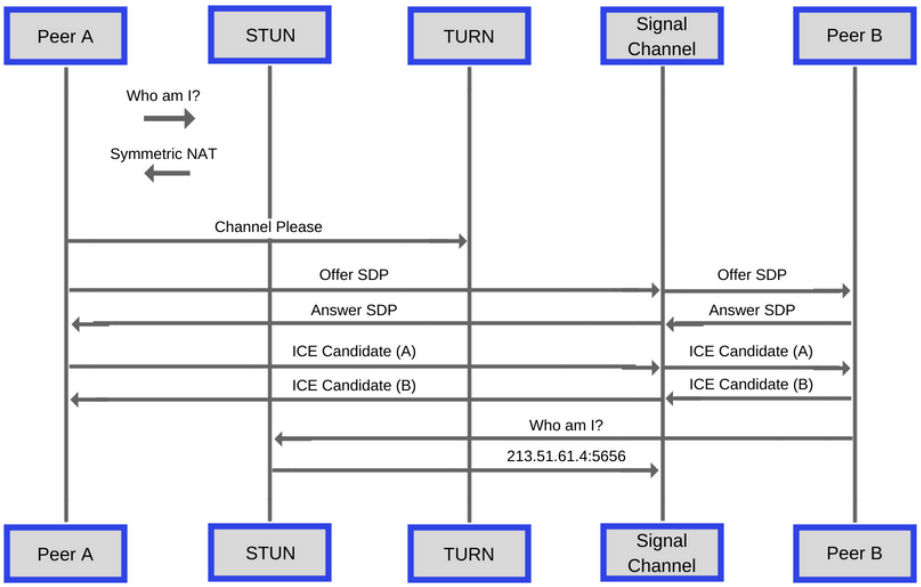
A STUN (Session Traversal Utilities for NAT) server is an essential component of the WebRTC (Web Real-Time Communications) technology stack. It is used to help establish peer-to-peer communication between two WebRTC endpoints that are located behind NAT (Network Address Translation) devices, which are commonly used in home and office networks.

When two WebRTC endpoints try to establish a peer-to-peer connection, they may encounter NAT devices that prevent direct communication. In this case, the endpoints can use a STUN server to discover their public IP address and port, which can be used by the other endpoint to send data directly to them.

The STUN server works by responding to a STUN request from the endpoint with a message that contains the endpoint's public IP address and port number. The endpoint can then use this information to set up a direct connection with the other endpoint.

WebRTC also supports another technology called TURN (Traversal Using Relay NAT), which can be used as a fallback mechanism if the STUN server is unable to establish a direct connection between the two endpoints. TURN uses a relay server to forward data between the endpoints if a direct connection cannot be established.

In summary, a STUN server is an important component of the WebRTC technology stack that helps establish peer-to-peer communication between two endpoints that are located behind NAT devices. It enables the endpoints to discover their public IP address and port, which can be used to establish a direct connection or as a fallback mechanism for relay-based communication using TURN.



2.2 **SDP (Session Description Protocol)**

SDP (Session Description Protocol) is a format for describing the configuration and capabilities of multimedia sessions, including WebRTC

sessions. SDP is used in WebRTC to negotiate the parameters and settings for a peer-to-peer connection between two endpoints, such as the type of media to be exchanged, the codecs to be used, and the network configurations.

In WebRTC, the SDP information is exchanged between the two endpoints as part of the signaling process, which is the process of exchanging information to establish a connection. The SDP information is contained in a text-based message that describes the configuration of the media streams to be exchanged.

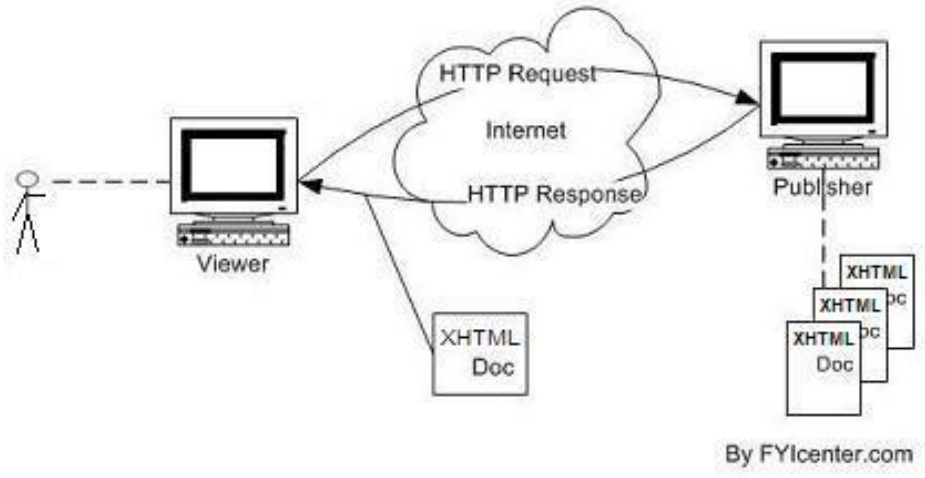
The SDP message typically includes several sections, each describing different aspects of the session configuration, such as:

- Session Description: contains information about the session, including the session name, session ID, and session version.
- Media Description: describes the media streams to be exchanged, including the media type (e.g. audio or video), the transport protocol (e.g. RTP or SRTP), the codec type (e.g. VP8 or Opus), and the network configuration (e.g. IP address and port number).
- Connection Data: provides information about the network configuration, such as the IP address and port number.
- ICE Candidates: describes the possible network paths between the two endpoints.

Once the two endpoints exchange their SDP messages, they can negotiate the session configuration and establish a peer-to-peer connection. The SDP negotiation process can involve multiple rounds of exchange and modification of SDP messages until both endpoints agree on the session configuration.

In summary, SDP is a format for describing the configuration and capabilities of multimedia sessions, including WebRTC sessions. SDP is used in WebRTC to negotiate the parameters and settings for a peer-to-peer connection

between two endpoints, and the SDP information is exchanged as part of the signaling process. The SDP message typically includes several sections that describe the session configuration, media streams, network configuration, and ICE candidates.



2.3 Http request response

HTTP (Hypertext Transfer Protocol) is a protocol used to transmit data over the internet. When a client (such as a web browser) requests a resource from a web server, it sends an HTTP request message to the server. The server then processes the request and sends an HTTP response message back to the client.

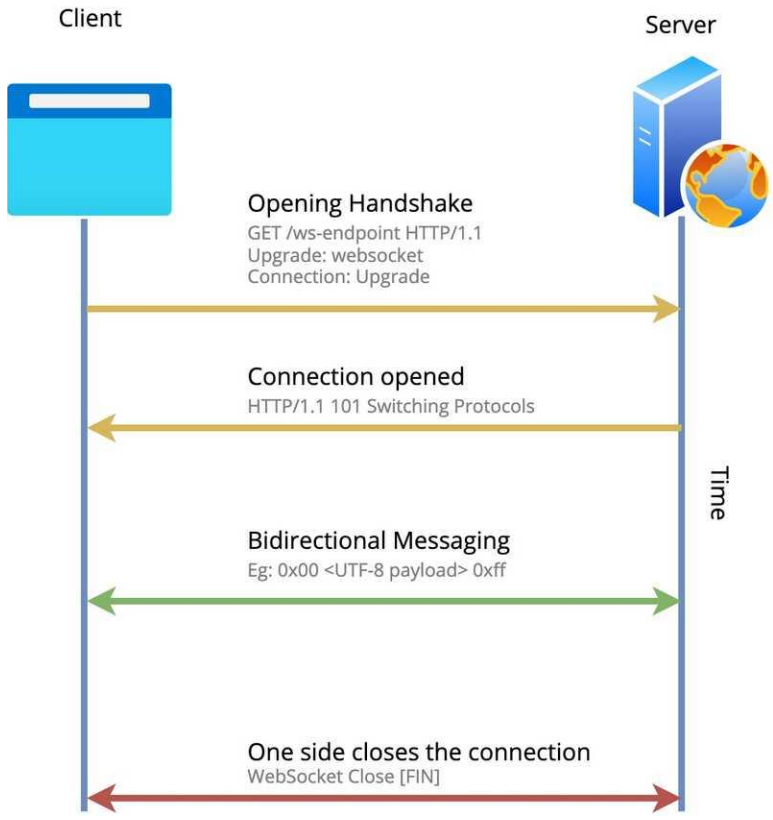
An HTTP request message typically consists of several parts, including:

- Request line: includes the HTTP method (such as GET or POST), the URL of the resource being requested, and the HTTP version.
- Request headers: provide additional information about the request, such as the user agent, the type of data being accepted, and authentication information.
- Request body: contains any data that is being sent to the server, such as form data or JSON payloads.

Once the server receives the HTTP request message, it processes the request and sends an HTTP response message back to the client. An HTTP response message typically consists of several parts, including:

- Status line: includes the HTTP version, a status code (such as 200 OK or 404 Not Found), and a status message.
- Response headers: provide additional information about the response, such as the content type, the length of the response, and caching information.
- Response body: contains the actual data being sent back to the client, such as HTML, JSON, or an image file.

The client then receives the HTTP response message and processes it accordingly. For example, if the response contained HTML content, the client would render the HTML in a web page.



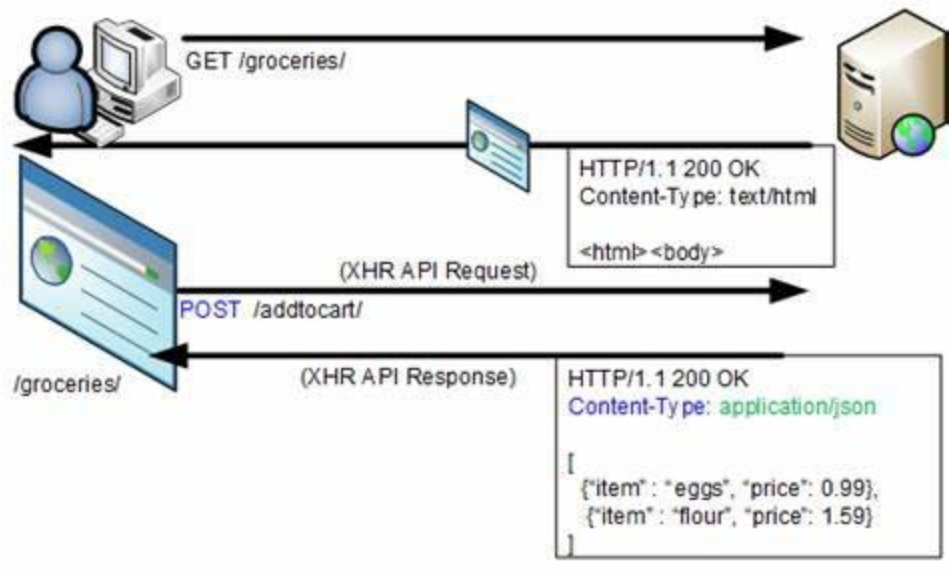
2.4 Web Socket connection type

WebSocket is a communication protocol that enables real-time, bidirectional communication between a client and a server over a single, long-lived connection. Unlike HTTP (Hypertext Transfer Protocol), which is a request-response protocol, WebSocket is a full-duplex protocol, which means that data can be sent and received by both the client and the server at any time, without needing to wait for a request or response.

WebSocket connections are initiated with an HTTP or HTTPS request, using a special HTTP header called the Upgrade header, which requests that the connection be upgraded to a WebSocket connection. If the server supports WebSocket connections, it will respond with an HTTP response that includes a WebSocket-specific header, and the connection will be upgraded to a WebSocket connection.

WebSocket connections can be used for a variety of real-time applications, such as chat applications, online games, and real-time data streaming. WebSocket connections have several advantages over other types of communication protocols, including:

- Low latency: WebSocket connections can be used to deliver real-time data with minimal latency, as data can be sent and received without waiting for a request or response.
- Full-duplex communication: WebSocket connections allow for bidirectional communication between the client and server, making it possible to push data to the client in real-time.
- Reduced overhead: WebSocket connections have less overhead than traditional HTTP connections, as they don't require headers to be sent with each message.



2.5 XMLHttpRequest (XHR)

XMLHttpRequest (XHR) is an API provided by web browsers that enables client-side scripts to make HTTP requests to a web server. It allows JavaScript code to send HTTP requests and receive responses from a server without reloading the entire page.

XHR was originally designed for exchanging XML data between client and server, but it can also be used with other data formats such as JSON or plain text. XHR requests can be sent asynchronously or synchronously. Asynchronous requests are more commonly used as they do not block the UI of the web page and allow other scripts to continue executing while the request is being processed.

XHR requests consist of several parts, including:

- A method: typically GET or POST, but can also be PUT, DELETE, etc.
- A URL: the location of the resource being requested.
- Optional request headers: additional information about the request, such as the content type or authorization credentials.
- An optional request body: data to be sent to the server with the request.

Once an XHR request is sent, the server processes the request and sends a response back to the client. The XHR object can then access the response data and use it as needed. XHR responses typically consist of a status code, headers, and a response body containing the data sent back by the server.

XHR has been widely used for many years and is still used today, but it has some limitations. For example, it is subject to the same-origin policy, which means that requests can only be made to the same domain as the web page making the request, unless the server explicitly allows cross-origin requests. Additionally, XHR can be difficult to work with for more complex scenarios, such as real-time updates or long-lived connections.

3.1 Design and Architecture

Real-Time Communication architecture

1. WebRTC:
- WebRTC is a free, open-source project that enables real-time communication (RTC) between browsers and mobile applications.
- WebRTC can be used for audio, video, and data sharing, and it is well-suited for use in online meeting platforms.

2. Signaling Server:
- A signaling server can be used to coordinate communications between clients using WebRTC.
- The signaling server acts as an intermediary between clients, allowing them to exchange messages and negotiate the parameters for establishing a WebRTC connection.
- The signaling server can be implemented using a number of technologies, such as Node.js, Java, or Python.

3. NAT Traversal:
- Network Address Translation (NAT) can cause issues with establishing a WebRTC connection, particularly if clients are located behind firewalls or routers.
- A STUN (Session Traversal Utilities for NAT) or TURN (Traversal Using Relay NAT) server can be used to help clients establish a WebRTC connection even if they are behind a NAT.

4. Media Servers:
- Media servers can be used to handle the media streams for online meetings.
- Media servers can perform functions such as transcoding, recording, and streaming.
- Media servers can be implemented using a number of technologies, such as Kurento, Janus, or Jitsi.

3.2 User Interface

1. Responsive Design:
- The user interface should be designed to be responsive and adaptable to a variety of devices, including desktops, laptops, tablets, and smartphones.
- The layout and navigation should be optimized for each device type, ensuring a consistent user experience across devices.
2. Intuitive Navigation:
- The user interface should be easy to navigate and intuitive, with clear and logical menus, buttons, and links.
- Navigation should be optimized for efficiency, allowing users to quickly access the features and information they need.
3. User-Centered Design:
- The user interface should be designed with the needs and preferences of the users in mind.
- User research and testing can be used to identify user needs, preferences, and pain points, which can then be used to inform the design.
4. Branding:
- The user interface should reflect the branding of the online meeting platform.
- This can include the use of colors, logos, and other visual elements that are consistent with the platform's branding.
5. Accessibility:
- The user interface should be designed to be accessible to users with disabilities, such as visual impairments or motor impairments.
- This can include features such as screen readers, text-to-speech, and keyboard navigation.
6. Consistency:
- The user interface should be consistent throughout the platform, with a common design language and visual style.
- This can help users understand the interface more quickly and reduce confusion

4.1 Implementation
Front End Development

HTML (Hypertext Markup Language) is a markup language used to create and structure content for the web. HTML documents consist of a series of HTML tags and elements that define the structure and content of a webpage. Here are some key concepts and elements of HTML:

1. Tags:

- HTML tags are used to define the structure and content of a webpage.
- Tags are enclosed in angle brackets (<>) and consist of a tag name and any associated attributes.

2. Elements:

- HTML elements are comprised of one or more tags and define the structure and content of a webpage.
- Elements may be nested within each other to create a hierarchical structure.

3. Attributes:

- HTML attributes provide additional information about a tag or element.

- Attributes are specified within the opening tag of an element and consist of a name and a value.

4. Head:

- The <head> element is used to define metadata for a webpage, such as the title, description, and keywords.
- This information is not visible to the user but is used by search engines and other programs to index and classify the webpage.

5. Body:

- The <body> element is used to define the visible content of a webpage.
- This includes text, images, videos, and other multimedia elements.

6. Links:

- HTML links allow users to navigate between webpages and to other resources on the web.
- Links are created using the <a> (anchor) tag and can be styled with CSS to create different visual effects.

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of HTML (Hypertext Markup Language) and XML (Extensible Markup Language) documents on the web. Here are some key concepts and elements of CSS:

1. Selectors:

- CSS selectors are used to target HTML elements and apply styles to them.
- Selectors can target elements based on their tag name, class, ID, and other attributes.

2. Properties:

- CSS properties are used to define the style of HTML elements.
- Properties can be used to control things like font size, color, spacing, borders, and background images.

3. Values:

- CSS values are used to define the specific settings for a given property.
- Values can be numeric (e.g. font size), color codes (e.g. #000000 for black), or keywords (e.g. bold for font-weight).

4. Classes:

- CSS classes are used to apply styles to groups of HTML elements.

- Classes are defined using the . (dot) notation and can be applied to HTML elements using the class attribute.

5. IDs:

- CSS IDs are used to apply styles to specific HTML elements.
- IDs are defined using the # (hash) notation and can be applied to HTML elements using the id attribute.

6. Inheritance:

- CSS styles can be inherited from parent elements to child elements.
- This allows for consistent styling across multiple HTML elements.

Bootstrap

Bootstrap is a free and open-source front-end web development framework that is used to design responsive and mobile-first web pages. It includes a set of CSS styles and JavaScript plugins that make it easy to create consistent, professional-looking layouts and UI components.Here are some key features of Bootstrap:

1. Responsive design:

- Bootstrap is designed to be responsive, meaning that it automatically adjusts the layout of a webpage to fit different screen sizes and devices, such as mobile phones, tablets, and desktop computers.

2. Grid system:

- Bootstrap's grid system is a powerful tool for creating responsive layouts.
- It allows you to easily create rows and columns that resize and stack based on the screen size.

3. UI components:

- Bootstrap includes a wide variety of pre-built UI components, such as navigation bars, dropdowns, forms, modals, and carousels.
- These components can be easily customized and styled to fit the needs of your project.

4. Cross-browser compatibility:

- Bootstrap is designed to work across all major web browsers, including Chrome, Firefox, Safari, and Internet Explorer.

5. Easy to customize:

- Bootstrap is highly customizable, allowing you to change the colors, fonts, and other aspects of the design to match your brand or project.

6. Large community:

- Bootstrap has a large and active community of developers, which means that there are a lot of resources available, including documentation, tutorials, and plugins.

5.1 Backened Development Technology

Node.js is a server-side, open-source, cross-platform JavaScript runtime environment. It allows developers to run JavaScript on the server-side, which was traditionally done using languages such as PHP, Python, or Ruby. Node.js was developed by Ryan Dahl in 2009 and has since become very popular among developers.

One of the key benefits of Node.js is its event-driven, non-blocking I/O model, which allows it to handle large volumes of data and requests efficiently. This makes it an excellent choice for building scalable, real-time applications such as chat apps, video conferencing, and online games.

Node.js is built on top of Google's V8 JavaScript engine and provides developers with a rich set of built-in libraries and modules that can be used to develop server-side applications. These libraries include HTTP, HTTPS, File System, and many others.

Some other features of Node.js include:

- Asynchronous programming: Node.js supports asynchronous programming using callbacks and promises, which allows developers to write non-blocking code that can handle multiple requests at the same time.
- NPM: Node Package Manager (NPM) is a package manager for Node.js that allows developers to easily install and manage dependencies for their applications.
- Cross-platform: Node.js can run on multiple platforms, including Windows, Mac, and Linux.
- Open-source: Node.js is an open-source project with a large community of developers contributing to its development.

Node.js has become a popular choice for building server-side applications due to its efficiency, scalability, and versatility.

Socket.IO is a JavaScript library that enables real-time, bi-directional communication between web clients (such as browsers) and servers. It uses websockets, which provide a persistent connection between a client and server, allowing for real-time data transfer.
Socket.IO also supports other transport mechanisms, such as polling and long-polling, for browsers that do not support websockets or when websockets are blocked by firewalls or proxies.
Socket.IO consists of two parts: a client-side library that runs in the browser, and a server-side library for Node.js. The two libraries work together to establish and maintain a real-time connection between the client and server.
Socket.IO provides a number of features, including:

- Event-based communication: Clients and servers can send and receive custom events, allowing for flexible and efficient communication.
- Room management: Clients can be grouped into "rooms" to enable selective broadcast of messages to specific clients or groups of clients.
- Error handling: Socket.IO provides mechanisms for error handling, including server-side logging and client-side error events.
- Reconnection: If a connection is lost, Socket.IO will automatically attempt to reconnect and resume communication.
- Binary data transfer: Socket.IO supports the transfer of binary data, in addition to text-based data.

Socket.IO is widely used for real-time web applications, such as chat applications, online gaming, and collaborative editing tools.

Express.js is a popular, minimalist web framework for Node.js that is widely used for building web applications, APIs, and server-side web applications. It provides a set of features and tools for building robust, scalable, and maintainable web applications.

Express.js is built on top of Node.js and uses its core features such as the HTTP module for handling requests and responses, and the event-driven programming model for handling asynchronous code. Express.js provides an additional layer of abstraction on top of these core features, making it easier to build web applications.

Some of the features of Express.js include:

1. Routing: Express.js provides a simple and flexible mechanism for defining routes for handling HTTP requests.
2. Middleware: Express.js middleware functions are functions that are executed in the order they are defined and can modify the request and response objects, add functionality to the application, and handle errors.
3. Templating: Express.js provides support for various template engines like Pug, EJS, and Handlebars, making it easy to render dynamic views for your web application.
4. Error handling: Express.js provides a robust mechanism for handling errors, including default error handlers and custom error handlers.
5. Static file serving: Express.js makes it easy to serve static files like images, CSS, and JavaScript files.

Express.js is highly extensible and can be easily integrated with other Node.js modules and libraries. It is also easy to learn, making it a great choice for developers who are new to Node.js and web development.

WebRTC (Web Real-Time Communication) is a free, open-source project that provides real-time communication capabilities to web browsers and mobile applications through simple APIs. It enables peer-to-peer communication between browsers without requiring any plugins or downloads, making it a powerful tool for building real-time applications like video conferencing, online gaming, and file sharing.

Some of the features of WebRTC include:

1. Audio and video communication: WebRTC provides APIs for real-time audio and video communication between web browsers.
2. Data channel: In addition to audio and video communication, WebRTC also provides a data channel for sending arbitrary data between browsers.
3. NAT and firewall traversal: WebRTC includes mechanisms for NAT and firewall traversal, allowing it to work in a variety of network environments.
4. Encryption: WebRTC includes support for encryption, ensuring that all communication is secure and private.

WebRTC is supported by all major web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, making it a powerful tool for building cross-platform applications. It is built on top of JavaScript and provides simple, easy-to-use APIs for developers to work with.

Overall, WebRTC is a powerful and flexible tool for building real-time communication applications on the web. It has become increasingly popular in recent years as more and more developers look to build real-time applications on the web.

Heroku provides a fully-managed platform that abstracts away the underlying infrastructure and allows developers to focus on their application code. It utilizes a container-based architecture to isolate applications and provide better scalability, security, and fault tolerance.

One of the key features of Heroku is its ease of deployment. Developers can easily deploy their applications using Git or a deployment pipeline, and Heroku takes care of the rest, including managing the server infrastructure, load balancing, and scaling.

Heroku also provides a wide range of add-ons and services that can be easily integrated into applications, such as databases, caching, logging, monitoring, and more. These add-ons can be easily added to an application with just a few clicks and can help extend its functionality.

Another benefit of Heroku is its pricing model, which is based on the resources used by an application, such as the number of dynos (containers) and add-ons. This allows developers to scale their applications up or down as needed, and only pay for the resources they use.

Heroku also has a strong community of developers who contribute to open-source libraries, tools, and documentation. This community is supported by Heroku's Dev Center, which provides a wealth of resources for developers, including guides, tutorials, and best practices.

In summary, Heroku is a cloud platform that offers a fully-managed infrastructure for deploying, managing, and scaling web applications. It provides ease of deployment, a wide range of add-ons and services, a flexible pricing model, and a supportive community of developers.

6.1 Importance of Testing

1. Software testing is really required to point out the defects and errors that were made during the development phases.Example: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.

2. It's essential since it makes sure that the customer finds the organization reliable and their satisfaction in the application is maintained. If the customer does not find the testing organization reliable or is not satisfied with the quality of the deliverable, then they may switch to a competitor organization.
Sometimes contracts may also include monetary penalties with respect to the timeline and quality of the product. In such cases, if proper software testing may also prevent monetary losses.

3. It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence. (Know more about Software Quality) As explained in the previous point, delivering good quality product on time builds the customers confidence in the team and the organization.

4. Testing is necessary in order to provide the facilities to the customers like the delivery of the high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results. High quality product typically has fewer defects and requires lesser maintenance effort, which in turn means reduced costs.

5. Testing is required for an effective performance of software application or product.

6. It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development. Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application. If defects related to requirements or design are detected late in the life cyle, it can be very expensive to fix them since this might require redesign, re-implementation and retesting of the application.

7. It's required to stay in the business.Users are not inclined to use software that has bugs. They may not adopt a software if they are not happy with the stability of the application. In case of a product organization or startup which has only one product, poor quality of software may result in lack of adoption of the product and this may result in losses which the business may not recover from.

6.2 Types of Testing

6.3 Functional Testing:

Functional Testing is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black Box Testing technique is used in which the internal

logic of the system being tested is not known to the tester.Functional testing is normally performed during the levels of System Testing and Acceptance Testing.
Typically, functional testing involves the following steps:

- Identify functions that the software is expected to perform.
- Create input data based on the function's specifications.
- Determine the output based on the function's specifications.
- Execute the test case.
- Compare the actual and expected outputs.

Functional testing is more effective when the test conditions are created directly from
user/business requirements. When test conditions are created from the system documentation (system requirements/ design documents), the defects in that documentation will not be detected through testing and this may be the cause of end-users' wrath when they finally use the software.

6.4. Non-Functional Testing

NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. An excellent example of a non-functional test would be to check how many people can simultaneously login into a software. Non-functional testing is equally important as functional testing and affects client satisfaction. The Non Functional requirements were also not given proper attention in the earlier test cycles. However, this has changed now. Non-functional tests are now most important as they consider all the application performance and security issues these days. This testing has a greater impact on applications when it comes to the performance of the application under high user traffic. This testing ensures that your application is stable and is
able to handle load under extreme conditions.

6.5 Regression Testing

REGRESSION TESTING is a type of software testing that intends to ensure

that changes (enhancements or defect fixes) to the software have not adversely affected it. The likelihood of any code change impacting functionalities that are not directly associated with the code is always there and it is essential that regression testing is conducted to make sure that fixing one thing has not broken another thing.
During regression testing, new test cases are not created but previously created test cases are re-executed. Regression [noun] literally means the act of going back to a previous place or state; return or reversion. In an ideal case, a full regression test is desirable but oftentimes there are time/resource constraints. In such cases, it is essential to do an impact analysis of the changes to identify areas of the software that have the highest probability of being affected by the change and that have the highest impact to users in case of malfunction and focus testing around those areas. Due to the scale and importance of regression testing, more and more companies and projects are adopting regression test automation tools.

6.6 Performance Testing

Performance testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload.
Performance testing can involve quantitative tests done in a lab, or occur in the production environment in limited scenarios. Typical parameters include processing speed, data transfer rate, network bandwidth and throughput, workload efficiency and reliability. For example, an organization can measure the response time of a program when a user requests an action or the number of millions of instructions per second (MIPS) at which a mainframe functions.

7.1 Source Code

Landing page

Action.html

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Create Or Join Meeting</title>
        <link rel="stylesheet" href="public/Assets/css/bootstrap.min.css">
        <!-- <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons"> -->
        <link rel="stylesheet" href="public/Assets/css/style.css">
        <style>

        </style>

    </head>
    <body style="padding-top: 3.5rem;">
        <nav class="navbar navbar-expand-md fixed-top">
            <img src="public/assets/images/google-meet-icon.png" class="logo"
alt="">
            <a href="#" class="navbar-brand text-dark">WebMeet</a>
            <div class="collapse navbar-collapse">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item">
                        <!-- <a href="#" class="nav-link">At a glance</a>
                    </li>
                    <li class="nav-item">
                        <a href="#" class="nav-link">How it works</a>
                    </li>
                    <li class="nav-item">
                        <a href="#" class="nav-link">Plan and Price</a>
                    </li> -->
                </ul>
                <ul class="navbar-nav mr-0">
                    <li class="nav-item sign-in display-center">
                        <a href="#" class="nav-link">Sign in</a>
                    </li>


                    <li class="nav-item">
```

```
                        <button class="btn btn-lg btn-info text-light font-
weight-bold new-meeting">Start a meeting</button>
                    </li>
                    <li class="nav-item">
                        <button class="btn btn-outline-secondary btn-lg text-info
font-weight-bold join-meeting" >Join the meeting</button>

                    </li>
                </ul>
            </div>
        </nav>
        <main>
            <div class="jumbotron h-100 d-flex">
                <div class="container w-50">
                    <!-- <h1 style="font-size: 3rem;"> Premium video meeting. Now
it is available for free to everyone. </h1>
                    <p style="font-size: 20px;">
                        We're redesigning the WebMeet service for secure business
meetings and making it free for everyone to use.
                    </p> -->
                    <ul class="display-center justify-content-start" >
                        <li style="padding: 0;">
                            <button class="btn btn-lg text-light font-weight-bold
display-center new-meeting" style="background-color: #3c0bff;">New
Meeting</button>
                        </li>
                        <li class="pl-3">
                            <button class="btn btn-lg btn-outline-secondary text-
dark font-weight-bold display-center" style="background-color: #ffffff;"><input
type="text" placeholder="Enter a code" style="border: none;" class="enter-
code"></button>
                        </li>
                        <li class="text-dark font-weight-bold cursor-pointer pl-2
join-action">Join</li>

                    </ul>
                </div>
                    <!-- <div class="container w-50">
                        <img src="public/Assets/images/google-meet-people.avif"
alt="" class="signin-image">
                    </div> -->
            </div>
            <div class="container w-50">
                <img src="public/Assets/images/google-meet-people.avif" alt=""
class="signin-image">
```

```
            </div>
        </main>
        <!-- <footer class="container"><h6>Learn more about <span class="learn-
more text-info">WebMeet</span> .</h6></footer> -->
        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
        <script>
        $(function(){
            $(document).on("click", ".join-meeting", function(){
                $(".enter-code").focus();
            })
            $(document).on("click", ".join-action", function(){
                var join_value = $('.enter-code').val();
                var meetingUrl = window.location.origin+"?meetingID="+join_value;
                window.location.replace(meetingUrl);
            })
            $(document).on("click", ".new-meeting", function(){
            var eight_d_value = Math.floor(Math.random()*100000000);
                var meetingUrl =
window.location.origin+"?meetingID="+eight_d_value;
                window.location.replace(meetingUrl);
            })

        })</script>
    </body>
    </html>
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Google Meet</title>
    <link rel="stylesheet" href="public/Assets/css/bootstrap.min.css">
    <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
    <link rel="stylesheet" href="public/Assets/css/style.css">
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.4/socket.io.js"></scrip
t>
    <script src="public/Assets/js/jquery-3.4.1.min.js"></script>
    <script src="public/Assets/js/app.js"></script>
    <script>
        $(function(){
            const urlParams = new URLSearchParams(window.location.search);
            var meeting_id = urlParams.get('meetingID');
            var user_id = window.prompt('Enter your userid');

            if(!user_id || !meeting_id){
                alert('User id or meeting id missing');
                window.location.href = '/action.html';
                return;
            }
            $("#meetingContainer").show();

            MyApp._init(user_id,meeting_id);
        })
    </script>
</head>
<body>
    <main class=" d-flex flex-column home-wrap">
        <div class="g-top text-light">
            <div class="top-remote-video-show-wrap d-flex">
                <div id="meetingContainer"  style="display: none;flex-basis:
75%;">
                <div class="call-wrap" style="background-color: black;">
                <div class="video-wrap" id="divUsers" style="display:flex; flex-
wrap:wrap">
                    <div id="me" class="userbox display-center flex-column">
                        <h2 class="display-center" style="font-size: 14px;"></h2>
```

```
                        <!-- ..................HandRaise ..............-->
                        <div class="display-center" style="position: relative;">
                            <img class="handRaise"
src="/public/Assets/images/handRaise.png" style="position: absolute;height:
30px;top:8%; left: 3%;display: none;"/>

                             <!-- ..................HandRaise ..............--
>
                            <video autoplay muted id="locaVideoPlayer"></video>
                        </div>
                    </div>
                    <div id="otherTemplate" class="userbox display-center flex-
column" style="display:none">
                        <h2 class="display-center" style="font-size: 14px;"></h2>
                        <!-- ..................HandRaise ..............--
>
                        <div class="display-center" style="position: relative;">
                            <img src="/public/Assets/images/handRaise.png"
style="position: absolute;height: 30px;top:8%; left: 3%;display: none;"/>
                            <!-- ..................HandRaise
..............-->
                            <video autoplay muted></video>
                            <audio autoplay controls
style="display:none"></audio>
                        </div>
                    </div>
                </div>
                </div>
            </div>
            <div class="g-right-details-wrap bg-light text-secondary h-100"
style="flex-basis: 25%;  z-index: 1; display: none;">
                <div class="meeting-heading-wrap d-flex justify-content-between
align-items-center pr-3 pl-3" style="height: 10vh;">
                    <div class="meeting-heading font-weight-bold ">Meeing
Details</div>
                    <div class="meeting-heading-cross display-center cursor-
pointer">
                        <span class="material-icons">clear</span>
                    </div>
                </div>
                <div class="people-chat-wrap d-flex justify-content-between
align-items-center ml-3 mr-3 pr-3 pl-3" style="height: 10vh;font-size: 14px;">
                    <div class="people-heading display-center cursor-pointer">
                        <div class="people-headin-icon display-center mr-1">
                            <span class="material-icons">people</span>
```

```
                        </div>
                        <div class="people-headin-text display-center">
                            Participant  (<span class="participant-
count">1</span>)
                        </div>
                    </div>
                    <div class="chat-heading d-flex just-content-round align-
items-center cursor-pointer">
                        <div class="chat-heading-icon display-center mr-1">
                            <span class="material-icons">
                                message
                            </span>
                        </div>
                        <div class="chat-heading-text">
                            Chat
                        </div>
                    </div>
                </div>
                <div class="in-call-chat-wrap mr-3 ml-3 pl-3 pr-3" style="font-
size: 14px; height: 69vh; overflow-y: scroll;">
                    <div class="in-call-wrap-up" style="display: none
!important;">
                        <div class="in-call-wrap d-flex justify-content-between
align-items-center mb-3">
                            <div class="participant-img-name-wrap display-center
cursor-pointer">
                                <div class="participant-img">
                                    <img src="public/Assets/images/other.jpg"
alt="" class="border border-secondary" style="height: 40px;width: 40px;border-
radius: 50%;">
                                </div>
                                <div class="participant-name ml-2">You</div>
                            </div>
                            <div class="participant-action-wrap display-center">
                                <div class="participant-action-dot display-center
mr-2 cursor-pointer">
                                    <span class="material-icons">
                                        more_vert
                                    </span>
                                </div>
                                <div class="participant-action-pin display-center
mr-2 cursor-pointer">
                                    <span class="material-icons">
                                        push_pin
                                    </span>
```

```
                                </div>
                            </div>
                        </div>
                        <div class="chat-show-wrap text-secondary  flex-column
justify-content-between h-100" style="font-size: 14px; display: flex;">
                            <div class="chat-message-show" id="messages"></div>
                            <div class="chat-message-sent d-flex justify-content-
between align-items-center" style="margin-bottom:35px">
                                <div class="chat-message-sent-input" style="width:
85%;">
                                    <input type="text" name="" class="chat-message-
sent-input-field w-100" id="msgbox" placeholder="Send a message to everyone"
style="border-bottom: 1px solid teal; border: none;">
                                </div>
                                <div class="chat-message-sent-action display-center"
id="btnsend" style="color: teal; cursor:pointer;">
                                    <span class="material-icons">send</span>
                                </div>
                        </div>
                        </div>
                    </div>

                </div>


                </div>
                <div class="g-top-left bg-light text-secondary w-25 d-flex align-
items-center justify-content-between pl-2 pr-2">
                    <div class="top-left-participant-wrap pt-2 cursor-pointer">
                        <div class="top-left-participant-icon">
                            <span class="material-icons">people</span>
                        </div>
                        <div class="top-left-participant-count participant-
count">1</div>
                    </div>
                    <div class="top-left-chat-wrap pt-2 cursor-pointer">
                        <span class="material-icons">message</span>
                    </div>
                    <div class="top-left-time-wrap"></div>
                </div>
            </div>
            <div class="g-bottom bg-light m-0 d-flex justify-content-between align-
items-center">
                <div class="bottom-left d-flex" style="height:10vh">
```

```
                <div class="g-details border border-success mb-2" style="display:
none;min-height: 19.5vh;">
                    <div class="g-details-heading d-flex justify-content-between
align-items-center border-bottom pb-1">
                        <div class="g-details-heading-detail d-flex align-items-
center cursor-pointer">
                            <span class="material-icons">error</span
style="margin-top:-5px">Details<span></span>
                        </div>
                        <div class="g-details-heading-attachment d-flex align-
items-center cursor-pointer">
                            <span class="material-icons">attachment</span
style="margin-top:-5px">Attachment<span></span>
                        </div>
                    </div>
                    <div class="g-details-heading-show-wrap">
                        <div class="g-details-heading-show">
                            <div style="font-weight: 600;color:gray">Joining
Info</div>
                            <div class="meeting_url" style="padding: 5px 0;"
data-toggle="tooltip" data-placement="top"></div>
                            <div style="cursor: pointer;">
                                <span class="material-icons" style="font-
size: 14px;">content_copy</span>
                                <span class="copy_info font-weight-bold">Copy
Joining Info <span style="display: none;background-color: aquamarine; border-
radius: 5px;" class="link-conf font-weight-bold p-1">Link Copied</span></span>
                            </div>
                        </div>
                        <div class="g-details-heading-show-attachment"
style="display: none;position: relative;">
                            <div class="show-attach-file"></div>
                            <div class="upload-attach-file">
                                <form enctype="multipart/form-data"
ref="uploadForm" class="display-center" id="uploadForm" style="justify-content:
space-between;">
                                    <div class="custom-file" style="flex-
basis:79%">
                                        <input type="file" class="custom-
file-input" id="customFile" name="imagefile">
                                        <label for="customFile"
class="custom-file-label">Choose File</label>
                                    </div>
                                    <div class="share-button-wrap">
```

```
                                        <button class="btn btn-primary btn-sm
share-attach" style="flex-basis:19%;padding: 6px 20px;">Share</button>
                                    </div>
                                </form>
                            </div>


                        </div>
                    </div>
                </div>
                <div class="display-center cursor-pointer meeting-details-
button">
                    Meeting Details<span class="material-
icons">keyboard_arrow_down</span>
                </div>
            </div>
            <div class="bottom-middle d-flex just-content-center align-items-
center" style="height: 10vh;">
                <div class="mic-toggle-wrap action-icon-style display-center mr-2
cursor-pointer" id="miceMuteUnmute">
                    <span class="material-icons" style="width:
100%;">mic_off</span>
                </div>
                <div class="end-call-wrap action-icon-style display-center mr-2
cursor-pointer">
                    <span class="material-icons text-danger">call</span>
                </div>
                <div class="video-toggle-wrap action-icon-style display-center
cursor-pointer" id="videoCamOnOff"><span class="material-icons" style="width:
100%;">videocam_off</span></div>
                <!-- .....................HandRaise .................-->
                <div class="handRaiseAction display-center cursor-pointer"
id="handRaiseAction" style="margin-left: 5px;"><Img
src="/public/Assets/images/handRaiseBlack.png" style="height: 30px;width: 30px;"
/></div>
                <!-- .....................HandRaise .................-->
            </div>
            <div class="bottom-right d-flex just-content-center align-items-
center mr-3" id="screenShare-wrap" style="height: 10vh;">
                <div class="present-now-wrap d-flex just-content-center flex-
column align-items-center mr-5 cursor-pointer" id="ScreenShareOnOf">
                    <span class="material-icons">present_to_all</span>
                    <div>Present Now</div>
                </div>
```

```
                <div class="option-wrap cursor-pointer display-center"
style="height: 10vh; position:relative;">
                    <div class="recording-show">
                        <button class="btn btn-dark text-danger start-
record">Start Recording</button>
                    </div>
                    <div class="option-icon">
                        <span class="material-icons">more_vert</span>
                    </div>
                </div>
            </div>
        </div>
        <div class="top-box-show" style="display: none;">

        </div>
    </main>
</body>
</html>
```

Server.js

```
// Import required modules
const express = require("express");
const path = require("path");
const fs = require("fs");
const fileUpload = require("express-fileupload");
const io = require("socket.io");

// Create Express app
const app = express();

// Start the server and listen on port 3000
const server = app.listen(3000, function () {
  console.log("Listening on port 3000");
});

// Configure socket.io to use the server
const socketIo = io(server, {
  allowEIO3: true, // enable compatibility with Socket.IO v2.x clients
});

// Serve static files from the root directory
app.use(express.static(path.join(__dirname, "")));

// Initialize an array to store user connections
const userConnections = [];

// Handle socket.io connections
socketIo.on("connection", (socket) => {
  console.log("socket id is ", socket.id);

  // Handle userconnect event
  socket.on("userconnect", (data) => {
    console.log("userconnent", data.displayName, data.meetingid);

    // Find other users in the same meeting
    const other_users = userConnections.filter(
      (p) => p.meeting_id == data.meetingid
    );

    // Add user to the userConnections array
    userConnections.push({
      connectionId: socket.id,
      user_id: data.displayName,
      meeting_id: data.meetingid,
```

```
    });

    // Inform other users in the same meeting about the new user
    const userCount = userConnections.length;
    console.log(userCount);
    other_users.forEach((v) => {
      socket.to(v.connectionId).emit("inform_others_about_me", {
        other_user_id: data.displayName,
        connId: socket.id,
        userNumber: userCount,
      });
    });

    // Inform the new user about other users in the same meeting
    socket.emit("inform_me_about_other_user", other_users);
  });

  // Handle SDPProcess event
  socket.on("SDPProcess", (data) => {
    socket.to(data.to_connid).emit("SDPProcess", {
      message: data.message,
      from_connid: socket.id,
    });
  });

  // Handle sendMessage event
  socket.on("sendMessage", (msg) => {
    console.log(msg);
    const mUser = userConnections.find((p) => p.connectionId == socket.id);
    if (mUser) {
      const meetingid = mUser.meeting_id;
      const from = mUser.user_id;
      const list = userConnections.filter((p) => p.meeting_id == meetingid);
      list.forEach((v) => {
        socket.to(v.connectionId).emit("showChatMessage", {
          from: from,
          message: msg,
        });
      });
    }
  });

  // Handle fileTransferToOther event
  socket.on("fileTransferToOther", (msg) => {
    console.log(msg);
```

```
  const mUser = userConnections.find((p) => p.connectionId == socket.id);
  if (mUser) {
    const meetingid = mUser.meeting_id;
    const from = mUser.user_id;
    const list = userConnections.filter((p) => p.meeting_id == meetingid);
    list.forEach((v) => {
      socket.to(v.connectionId).emit("showFileMessage", {
        username: msg.username,
        meetingid: msg.meetingid,
        filePath: msg.filePath,
        fileName: msg.fileName,
      });
    });
  }
});

// Handle disconnect event

socket.on("disconnect", function () {
  console.log("Disconnected");
  var disUser = userConnections.find((p) => p.connectionId == socket.id);
  if (disUser) {
    var meetingid = disUser.meeting_id;
    userConnections = userConnections.filter(
      (p) => p.connectionId != socket.id
    );
    var list = userConnections.filter((p) => p.meeting_id == meetingid);
    list.forEach((v) => {
      var userNumberAfUserLeave = userConnections.length;
      socket.to(v.connectionId).emit("inform_other_about_disconnected_user", {
        connId: socket.id,
        uNumber: userNumberAfUserLeave,
      });
    });
  }
});

// <!-- .....................HandRaise .................-->
socket.on("sendHandRaise", function (data) {
  var senderID = userConnections.find((p) => p.connectionId == socket.id);
  console.log("senderID :", senderID.meeting_id);
  if (senderID.meeting_id) {
    var meetingid = senderID.meeting_id;
    // userConnections = userConnections.filter(
    //   (p) => p.connectionId != socket.id
```

```
      // );
      var list = userConnections.filter((p) => p.meeting_id == meetingid);
      list.forEach((v) => {
        var userNumberAfUserLeave = userConnections.length;
        socket.to(v.connectionId).emit("HandRaise_info_for_others", {
          connId: socket.id,
          handRaise: data,
        });
      });
    }
  });
  // <!-- .....................HandRaise .................-->
});

app.use(fileUpload());
app.post("/attachimg", function (req, res) {
  var data = req.body;
  var imageFile = req.files.zipfile;
  console.log(imageFile);
  var dir = "public/attachment/" + data.meeting_id + "/";
  if (!fs.existsSync(dir)) {
    fs.mkdirSync(dir);
  }

  imageFile.mv(
    "public/attachment/" + data.meeting_id + "/" + imageFile.name,
    function (error) {
      if (error) {
        console.log("couldn't upload the image file , error: ", error);
      } else {
        console.log("Image file successfully uploaded");
      }
    }
  );
});
```

App.js file

```
var AppProcess = (function () {
  var peers_connection_ids = [];
  var peers_connection = [];
  var remote_vid_stream = [];
  var remote_aud_stream = [];
  var local_div;
  var serverProcess;
  var audio;
  var isAudioMute = true;
  var rtp_aud_senders = [];
  var video_states = {
    None: 0,
    Camera: 1,
    ScreenShare: 2,
  };
  var video_st = video_states.None;
  var videoCamTrack;
  var rtp_vid_senders = [];
  async function _init(SDP_function, my_connid) {
    serverProcess = SDP_function;
    my_connection_id = my_connid;
    eventProcess();
    local_div = document.getElementById("locaVideoPlayer");
  }
  function eventProcess() {
    $("#miceMuteUnmute").on("click", async function () {
      if (!audio) {
        await loadAudio();
      }
      if (!audio) {
        alert("Audio permission has not granted");
        return;
      }
      if (isAudioMute) {
        audio.enabled = true;
        $(this).html(
          "<span class='material-icons' style='width:100%;'>mic</span>"
        );
        updateMediaSenders(audio, rtp_aud_senders);
        console.log(rtp_aud_senders);
      } else {
        audio.enabled = false;
        $(this).html(
```

```
          "<span class='material-icons' style='width:100%;'>mic_off</span>"
        );
        removeMediaSenders(rtp_aud_senders);
        audio.stop();
        console.log(rtp_aud_senders);
      }
      isAudioMute = !isAudioMute;
    });
    $("#videoCamOnOff").on("click", async function () {
      if (video_st == video_states.Camera) {
        await videoProcess(video_states.None);
      } else {
        await videoProcess(video_states.Camera);
      }
    });

    $("#ScreenShareOnOf").on("click", async function () {
      if (video_st == video_states.ScreenShare) {
        await videoProcess(video_states.None);
      } else {
        await videoProcess(video_states.ScreenShare);
      }
    });
  }
  async function loadAudio() {
    try {
      var astream = await navigator.mediaDevices.getUserMedia({
        video: false,
        audio: true,
      });
      audio = astream.getAudioTracks()[0];
      audio.enabled = false;
    } catch (e) {
      console.log(e);
    }
  }

  function connection_status(connection) {
    if (
      connection &&
      (connection.connectionState == "new" ||
        connection.connectionState == "connecting" ||
        connection.connectionState == "connected")
    ) {
      return true;
```

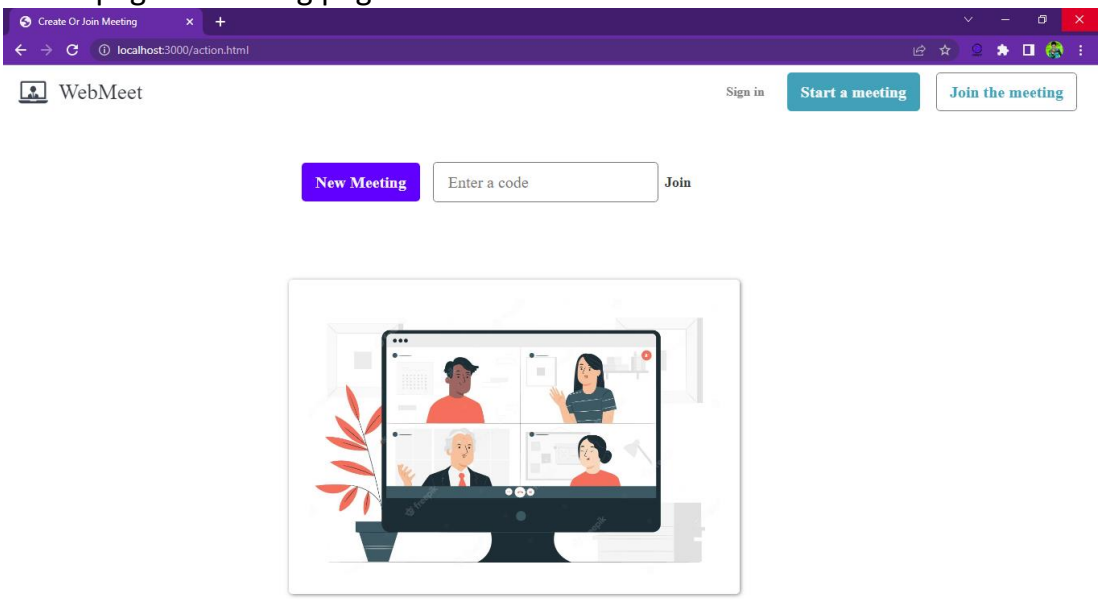
```
    } else {
      return false;
    }
  }
  async function updateMediaSenders(track, rtp_senders) {
    for (var con_id in peers_connection_ids) {
      if (connection_status(peers_connection[con_id])) {
        if (rtp_senders[con_id] && rtp_senders[con_id].track) {
          rtp_senders[con_id].replaceTrack(track);
        } else {
          rtp_senders[con_id] = peers_connection[con_id].addTrack(track);
        }
      }
    }
  }
  function removeMediaSenders(rtp_senders) {
    console.log("rtp_senders :", rtp_senders);
    for (var con_id in peers_connection_ids) {
      if (rtp_senders[con_id] && connection_status(peers_connection[con_id])) {
        peers_connection[con_id].removeTrack(rtp_senders[con_id]);
        rtp_senders[con_id] = null;
      }
    }
  }
  function removeVideoStream(rtp_vid_senders) {
    if (videoCamTrack) {
      videoCamTrack.stop();
      videoCamTrack = null;
      local_div.srcObject = null;
      removeMediaSenders(rtp_vid_senders);
    }
  }
  async function videoProcess(newVideoState) {
    if (newVideoState == video_states.None) {
      $("#videoCamOnOff").html(
        "<span class='material-icons' style='width:100%;'>videocam_off</span>"
      );
      $("#ScreenShareOnOf").html(
        '<span class="material-icons">present_to_all</span><div>Present
Now</div>'
      );
      video_st = newVideoState;

      removeVideoStream(rtp_vid_senders);
      console.log("rtp_vid_senders", rtp_vid_senders);
```

```
      // Video_switch_off
      serverProcess(
        JSON.stringify({
          Video_switch_off: "Video_switch_off",
        }),
        rtp_vid_senders
      );
      // Video_switch_off
      return;
    }
    if (newVideoState == video_states.Camera) {
      $("#videoCamOnOff").html(
        "<span class='material-icons' style='width:100%;'>videocam_on</span>"
      );
    }
    try {
      var vstream = null;
      if (newVideoState == video_states.Camera) {
        vstream = await navigator.mediaDevices.getUserMedia({
          video: {
            width: 1920,
            height: 1080,
          },
          audio: false,
        });
      } else if (newVideoState == video_states.ScreenShare) {
        vstream = await navigator.mediaDevices.getDisplayMedia({
          video: {
            width: 1920,
            height: 1080,
          },
          audio: false,
        });
        vstream.oninactive = (e) => {
          removeVideoStream(rtp_vid_senders);
          $("#ScreenShareOnOf").html(
            '<span class="material-icons ">present_to_all</span><div >Present
Now</div>'
          );
        };
      }
      if (vstream && vstream.getVideoTracks().length > 0) {
        videoCamTrack = vstream.getVideoTracks()[0];
        if (videoCamTrack) {
          local_div.srcObject = new MediaStream([videoCamTrack]);
```

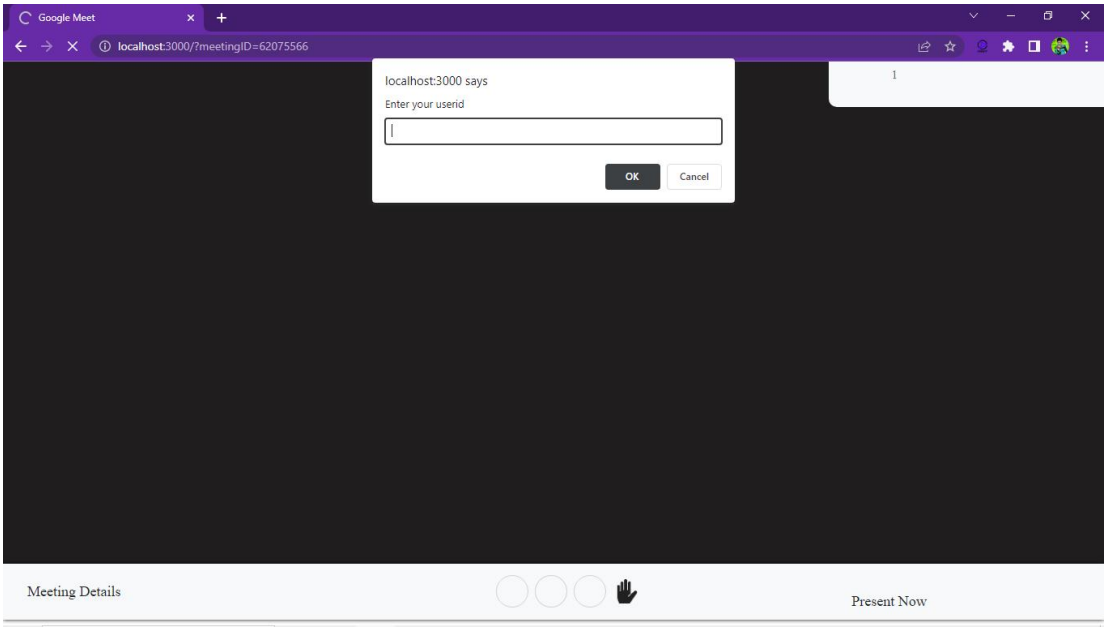
```
          updateMediaSenders(videoCamTrack, rtp_vid_senders);
        }
      }
    } catch (e) {
      console.log(e);
      return;
    }
    video_st = newVideoState;
    if (newVideoState == video_states.Camera) {
      $("#videoCamOnOff").html(
        '<span class="material-icons" style="width: 100%;">videocam</span>'
      );
      $("#ScreenShareOnOf").html(
        '<span class="material-icons ">present_to_all</span><div >Present
Now</div>'
      );
    } else if (newVideoState == video_states.ScreenShare) {
      $("#videoCamOnOff").html(
        '<span class="material-icons" style="width: 100%;">videocam_off</span>'
      );
      $("#ScreenShareOnOf").html(
        '<span class="material-icons text-success">present_to_all</span><div
class="text-success">Stop Present Now</div>'
      );
    }
  }

  async function setConnection(connid) {
    var connection = new RTCPeerConnection(iceConfiguration);

    connection.onnegotiationneeded = async function (event) {
      await setOffer(connid);
    };
    connection.onicecandidate = function (event) {
      if (event.candidate) {
        serverProcess(
          JSON.stringify({ icecandidate: event.candidate }),
          connid
        );
      }
```

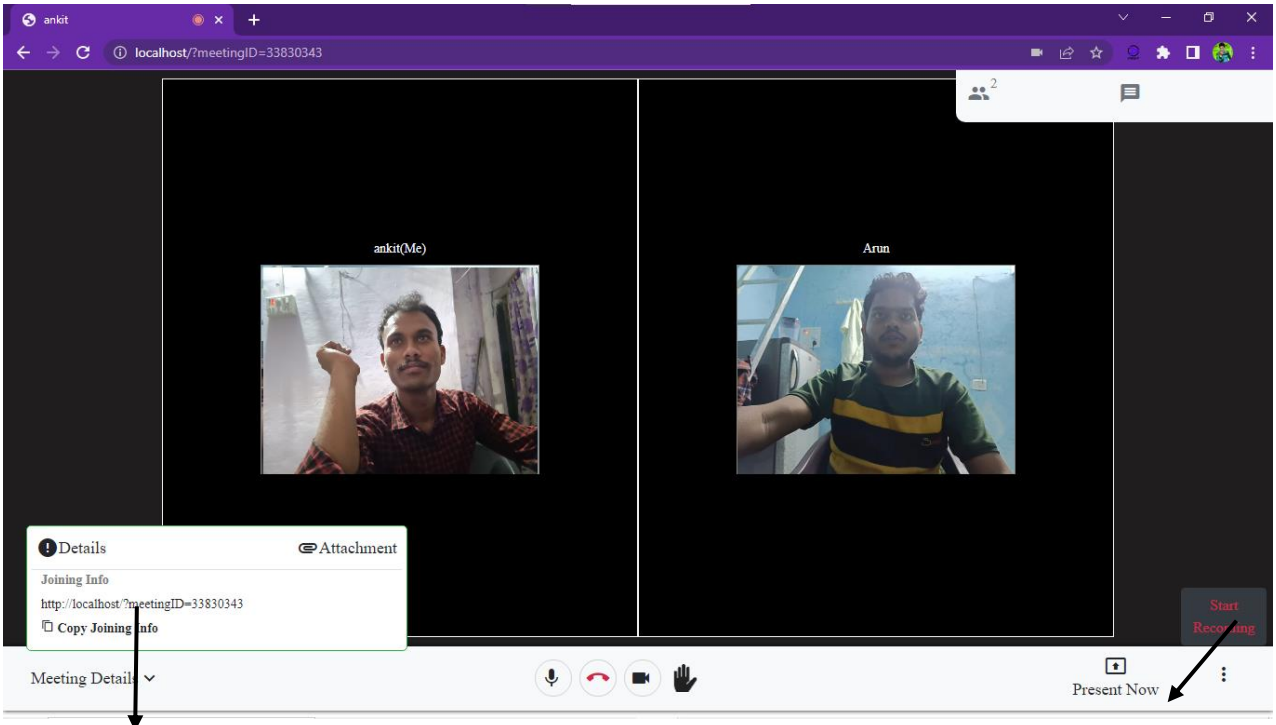
8. OUTPUT And PREVIEW

Home page or landing page screen shot



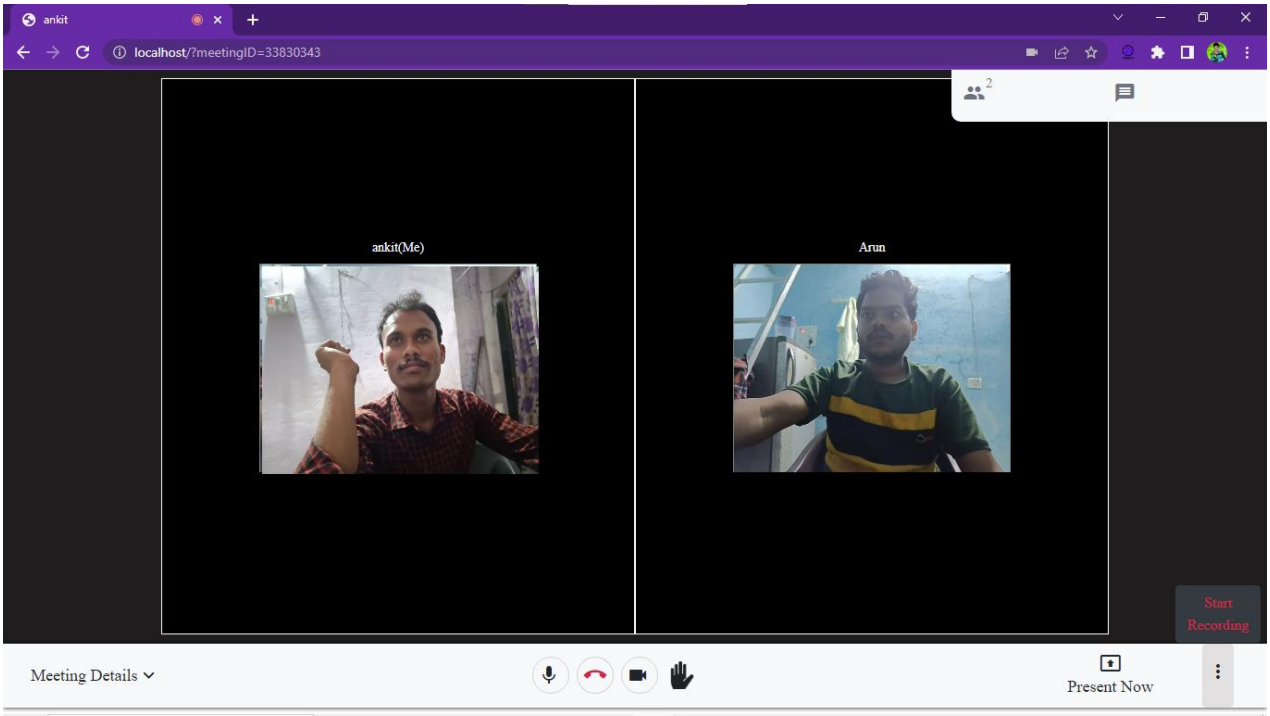
User ID request



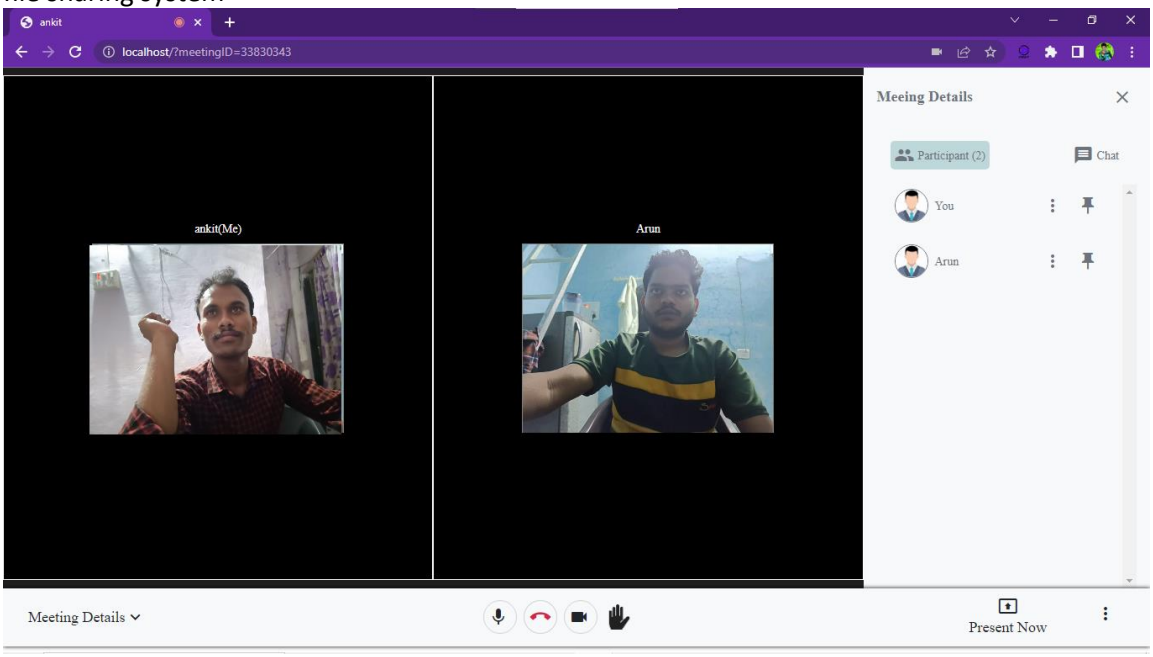


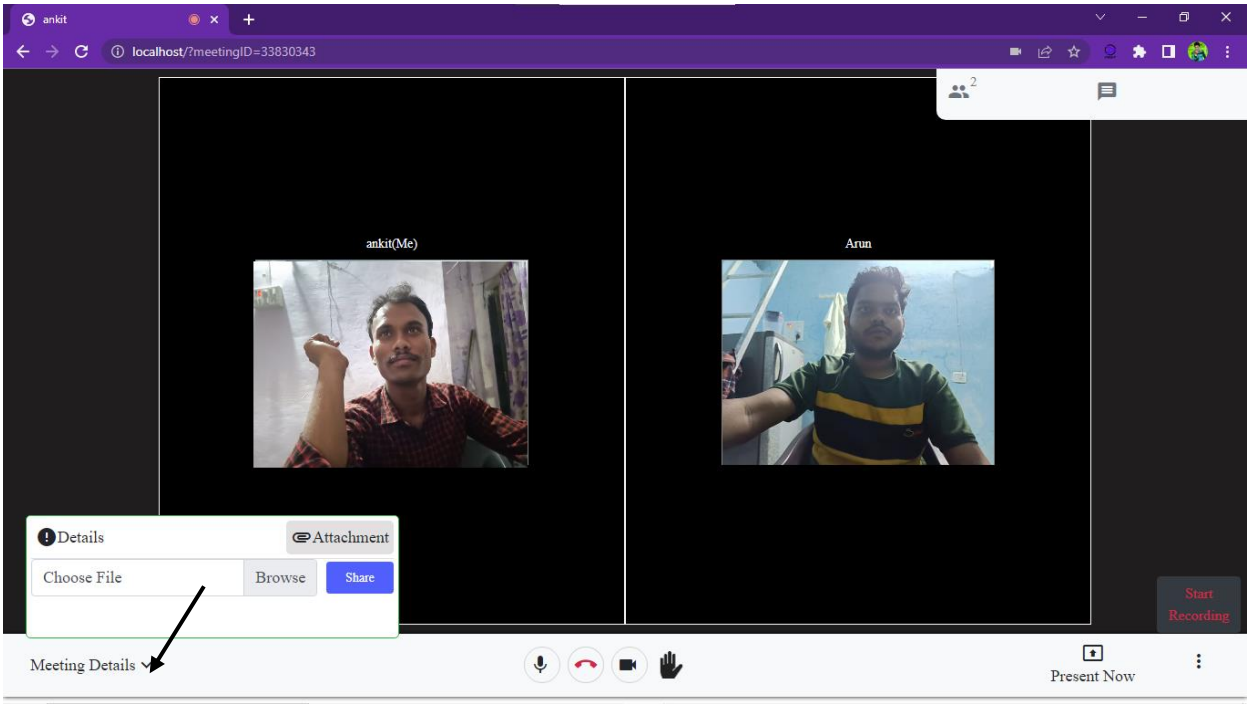
Meeting link

recording button



file sharing system





File sharing feature

9.1 Conclusion

In conclusion, an online meeting platform is an essential tool for modern-day businesses and individuals to connect and collaborate remotely. It provides a cost-effective and time-efficient way to communicate with others, share information, and conduct meetings from any location with an internet connection.

The benefits of using an online meeting platform include improved productivity, reduced travel costs, increased collaboration and engagement, and better work-life balance. With the advancement of technology and the need for remote work, the demand for online meeting platforms is only going to increase in the future.

To ensure the success of an online meeting platform, it is important to choose the right platform based on your needs and requirements, provide proper training and support to users, and implement security measures to protect sensitive information.

Overall, an online meeting platform can revolutionize the way we work and communicate in the digital age, and its importance will only continue to grow as we become increasingly connected and reliant on technology.

9.2 References

https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

https://socket.io/

https://developer.mozilla.org/en-US/docs/Web/CSS

https://developer.mozilla.org/en-US/docs/Web/JavaScript

https://developer.mozilla.org/en-US/docs/Web/HTML

https://getbootstrap.com/docs/4.1/getting-started/introduction/

https://api.jquery.com/

https://webrtc.org/getting-started/overview