A

MINOR PROJECT REPORT

ON

"PDM-Teams"

submitted in the partial fulfillment of the requirement
for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

SUBMITTED BY:

Deepansh Singh (A40319001/20)

Batch: 2019-2023

PROJECT GUIDE:

Dr. Jasvinder Kaur (Head, Department of CSE)

Department of Computer Science & Engineering
Faculty of Engineering & Technology
P.D.M. University, Bahadurgarh

A

MINOR PROJECT REPORT

ON

"PDM-Teams"

submitted in the partial fulfillment of the requirement
for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

SUBMITTED BY:

Deepansh Singh (A40319001/20)

Batch: 2019-2023

PROJECT GUIDE:

Dr. Jasvinder Kaur (Head, Department of CSE)

Department of Computer Science & Engineering
Faculty of Engineering & Technology
P.D.M. University, Bahadurgarh

Certificate of Completion

This is to certify that the Project work entitled **PDM-TEAMS** submitted by **Deepansh Singh (A40319001/20)** in fulfillment for the requirements of the award of **Bachelor of Technology** Degree in Computer Science & Engineering at PDMU, Bahadurgarh, Haryana is an authentic work carried out by his/her under my supervision and guidance. To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree.

Dr. Jasvinder Kaur
Head, Department of CSE
Faculty of Engineering & Technology

DECLARATION BY THE CANDIDATE

I hereby declare that the work presented in this report entitled **"PDM-Teams"**, in fulfillment of the requirement for the award of the degree Bachelor of Technology in Computer Science & Engineering, submitted in CSE Department, PDMU, Bahadurgarh, Haryana is an authentic record of my own work carried out during my degree under the guidance of **Dr. Jasvinder Kaur** (Head, Department of CSE)

The work reported in this has not been submitted by me for award of any other degree or diploma.

Date: 19-04-2022
Place: PDM University, Bahadurgarh

Deepansh Singh (A40319001/20)

B.Tech CSE 6th Sem

ACKNOWLEDGEMENT

I express my sincere gratitude to **Dr. Jasvinder Kaur (Head, Department of CSE)** her valuable guidance and timely suggestions during the entire duration of my dissertation work, without which this work would not have been possible. I would also like to convey my deep regards to all other faculty members who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish this work. Finally, I would also like to thank my friends for their advice and pointing out my mistakes.

Deepansh Singh (A40319001/20)

B.Tech CSE 6th Sem

ABSTRACT

This project is an example of server to server communication which is basically based on public video conferencing and chatting. The app allows multiple users to chat and video conference together. The messages will be updated without refreshing the page. For simplicity we will be avoiding the authentication part. The user should enter user name to chat and paste the link in the browser to connect.

Most of the tools designed are made using HTML, CSS, JAVASCRIPT and NODE.JS. The main impact of software is the interaction with the user of the program via peripheral devices like keyboard, mouse, etc.

TABLE OF CONTENTS

1. Certificate i
2. Declaration..... ii
3. Acknowledgement… iii
4. Abstract….....iv

CHAPTER 1

1. Introduction

Purpose of the Project 01
Technical Details… 01

CHAPTER 2

2. Implementation/Technology Environment

Tools and Technology.....02
HTML 02
CSS.....02
JavaScript… 02
Node.js…..... 03
Introduction to Web Socket… 03
How does Web Socket work?..... 03
Introduction to Socket.IO… 04
With Node HTTP Server..... 05
Express.js….....06
Firing Event…..... 07
Listening to Events 07
WebRTC… 08

Technology Environment… 09
Introduction to Visual Studio… 09

Node.js…..... 13

Gitbash..... 14

CHAPTER 3

3. Coding

Total Files in the Project… 15

assets..... .16

3.1.2 js..... .18

Stream.js ..... .25

App.js 26

index.html 27

CHAPTER 4

4. Testing and Result

Running the project..... .30

Basic view of the project..... 30

A new user joins a room..... 31

Chat using Web Socket ..... .32

Get universal access using ngrok 33

CHAPTER 5

Advantage and Disadvantages…..... 34

Conclusion 55

CHAPTER 6

6. Bibliography… 36

CHAPTER 7

7. Objective and Future..... 37

CHAPTER 8

Methodology… 38

REFERENCES 40

LIST OF FIGURES

Figure No.	Title	Page No.
1.	Web Socket	03
2.	Client request	04
2.	Server response	04
3.	Node HTTP Server	05
4.	Express.js	06
5.	Listening to Events	07
6.	WebRTC	08
7.	Getting start with visual studio	11
8.	Menus in visual studio	11
9.	Toolbar	12
10.	Node.js setup	14
11.	Git setup	14
12.	Total file in the project	15
13.	Public Folder	15
14.	Basic view of project	30
15.	Basic view of project	31
16.	A new user join room	31
17.	A new user join room	32
18.	Chat using Web Socket	32
19.	Get universal access using ngrok	33
20.	Get universal access using ngrok	33
21.	Methodology	38
22.	project timeline	39

CHAPTER 1

1. INTRODUCTION

Our project is an example of server-to-server communication which is basically based on public video conferencing and chatting. The app allows multiple users to chat and video conference together. The messages will be updated without refreshing the page. For simplicity we will be avoiding the authentication part. The user should enter username to chat and paste the link in the browser to connect.

PURPOSE OF THE PROJECT

■ Our university website doesn't have a real time video conferencing app.
■ So it is very difficult for the student to attend online class so I decided to make an online conferencing portal so that I can take my all classes without facing any type of lagging.
■ We faced network problem in the period of covid-19 which tends to cancel the required class.
■ As we are providing the chatting feature to the project, so it is very easy to the teachers to answer the query of students by the text. Text sharing is very important to communicate to other students so that everyone should connect to the portal very easily.

TECHNICAL DETAILS

SOFTWARE REQUIREMENTS:

- HTML (Hyper Text Markup Language), CSS (Cascading Style Sheet) (Front End)
- JAVASCRIPT, NODE JS, EXPRESS, WEB SOCKETS.IO, WebRTC (Back End)
- Microsoft Windows, Linux, Mac (All OF THEM)
- Internet Explorer (5.0 and above) or Mozilla Firefox (6.0 and above)
- VSCODE text editor, A Bash terminal / command prompt.

HARDWARE REQUIREMENTS:

- Intel core 2 (minimum spec)
- Ram: - 2 GB
- Computer Memory: - 128 GB
- Intel HD graphics (minimum version 15.40.46.5144)

CHAPTER 2
IMPLEMENTATION / TECHNOLOGY ENVIRONMENT

Tools and Technology used

FRONTEND

- We are using HTML, CSS, JAVASCRIPT for the frontend part

BACKEND

- We are using NODE.JS for the backend logics.
- We are using WEB SOCKETS for the real time communication.
- And we are using EXPRESS.JS for the server of the site.
- WebRTC to allow media devices (camera and microphone) to stream audio and video between connected devices.

HTML

HTML (Hypertext Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content.

CSS

Cascading Style Sheets (CSS) is style sheet language used to describe the presentation of a document written in HTML or XML .CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

JavaScript

JavaScript is a programming language that allows you to implement complex things on web pages. Every time a web page does more than just sit there and display static information for you to look at—displaying timely content updates, interactive maps, scrolling video jukeboxes, or more.

Node.js

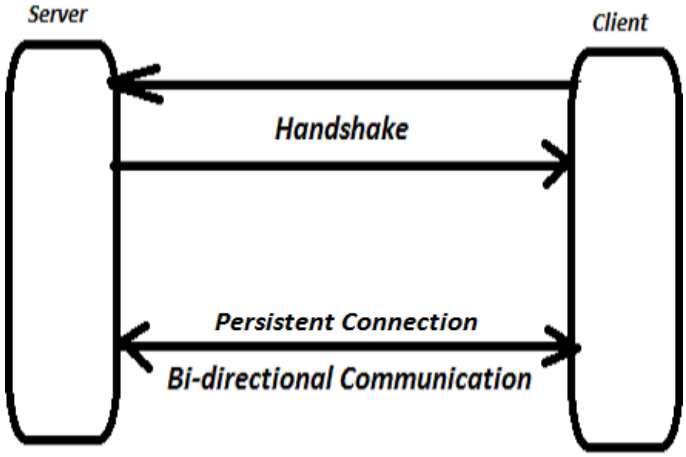
Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside the browser. The most important advantage of using Node is that we can use JavaScript as both a front-end and back-end language.
As we know, JavaScript was used primarily for client-side scripting, in which scripts were embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser.
Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting — running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

Introduction to Web Socket

Web Sockets are very beautiful tools that allow us to establish a real-time communication in modern web applications. In fact, this mechanism is so powerful, and it's used to build different kind of apps like real-time chat or notification system etc.

1. Protocol providing **full-duplex communication** channel over single **TCP-connection.**
2. Web was built around the idea – **'Client requests, Server fulfills'.**
3. AJAX got people started to look for ***bidirectional*** connections.
4. Other strategies like **long polling** had the problem of **carry overhead**, leading to increase in latency.

How does Web Sockets Work?



WebSockets

FIG 1

– Client Request

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Origin: http://example.com
```

– Server Response

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
```

FIG 2

1. Establish connection through 'Web Socket Handshake'

2. After handshake, initial HTTP connection is replaced by Web Socket connection (uses same underlying TCP/IP).

3. Transfer data without incurring any overhead associated with requests.

Introduction to Socket.IO

Socket.IO is a library that enables real-time, bidirectional, and event-based communication between the browser and the server. It consists of:

- a Node.js server: Source | API
- a JavaScript client library for the browser (which can be also run from Node.js): Source | API

1. **JavaScript library** for **real-time** web applications.

2. Has two parts:

– **Client-side** – Runs on Browser
– **Server-side** – Runs on Server – Node.js

3. Primarily uses **WebSocket**, but can **fallback** to other methods like **AJAX-long** polling, **JSONP** polling.

4. In addition to one-to-one communication as in WebSocket, it enables **broadcasting** to multiple nodes.

With Node HTTP Server

```
// app.js
// Requiring Module Dependencies

var app = require('http').createServer(serveFile),
io = require('socket.io')(app),
fs = require('fs');

app.listen(3000, function () {
  console.log('Server up and listening to port 3000');
});

// Handler to serve static files
function serveFile(req, res) {
        fs.readFile(__dirname + '/index.html', function(err, data) {
                if(err) {
                        res.writeHead(500);
                        return res.end('Error loading index.html');
                }
                res.writeHead(200);
                return res.end(data);
        });
}

// Socket.IO
io.on('connection', function (socket) {
        // Event 1
        socket.emit('event_1', {hello: 'world!'});
        // Event 2
        socket.on('event_2', function(data) {
                console.log(data);
        });
});
```

```
// index.html

<html>
<head>
        <title>Socket.io Demo</title>
        <script src="/socket.io/socket.io.js"></script>
        <script>
                var socket = io('http://localhost');
                socket.on('event_1', function (data) {
                        console.log(data);
                        socket.emit('event_2', { my: 'data' });
                });
        </script>
</head>
<body>
        <h1>Socket.IO Demo</h1>
</body>
</html>
```

FIG 3

Express.js

Express.js, or simply Express, is a web application framework for Node.js. Express provides a robust set of features for web and mobile applications. Express provides a thin layer of fundamental web application features, without obscuring Node.js features.
There are many frameworks that can be added as modules to our Node application. These will be explained further on as needed.

```
// Requiring Module Dependencies

var app = require('express')(),
server = app.listen(3000, function () {
  console.log('Server up and listening to port 3000');
}),
io = require('socket.io').listen(server);

app.get('/', function(req, res) {
        res.sendfile(__dirname + '/index.html');
});

// Socket.IO
io.on('connection', function (socket) {
        // Event 1
        socket.emit('event_1', {hello: 'world!'});
        // Event 2
        socket.on('event_2', function(data) {
                console.log(data);
        });
});
```

```
// index.html

<html>
<head>
        <title>Socket.io Demo</title>
        <script src="/socket.io/socket.io.js"></script>
        <script>
                var socket = io('http://localhost');
                socket.on('event_1', function (data) {
                        console.log(data);
                        socket.emit('event_2', { my: 'data' });
                });
        </script>
</head>
<body>
        <h1>Socket.IO Demo</h1>
</body>
</html>
```

FIG 4

Firing Event

- Individual Recipient **- EMIT**
  - Current connected socket

    SYNTAX: socket.emit('eventName', "Event Data");
  - Specific Socket

    SYNTAX: io.sockets.socket(socketId).emit('eventName', "Event Data");
- Multiple Recipients **– BROADCAST**
  - All connected nodes **except** the current one

    SYNTAX: socket.broadcast.emit('eventName', "Event Data");
  - All connected nodes

    SYNTAX: io.sockets.emit('eventName', "Event Data");
- Specific Channel **– TO/IN**
  - To all connected nodes in a channel **except** current

    SYNTAX: socket.broadcast.to(channelName).emit('eventName', "Event Data");
  - To all connected nodes in a channel

    SYNTAX: io.sockets.in(channelName).emit('eventName', "Event Data");

Listening to Events

Listening to events is easier as compared to firing events

Syntax:

```
socket.on('eventName', handler);
```

Example:

```
// Register a handler to listen to 'event_1'
socket.on('event_1', function (data) {
        // Respond by sending message with time stamp to all nodes
        io.sockets.emit('pushMessage', {
                message: data.message,
                time: new Date()
        });
});
```

FIG 5

2.12.1 WebRTC

WebRTC (Web Real Time Communication) is an open-source project that enables peer-to-peer communication between browsers. In other words, WebRTC allows you to exchange any kind of media through the web (such as video, audio, and data) without any required plugin or framework.

Direct communication between browsers improves performance and reduces latency times since clients don't need to keep sending and receiving messages through a server. For instance, we could use WebSocket's to connect two clients, but a server would have to route their messages as in the next diagram:

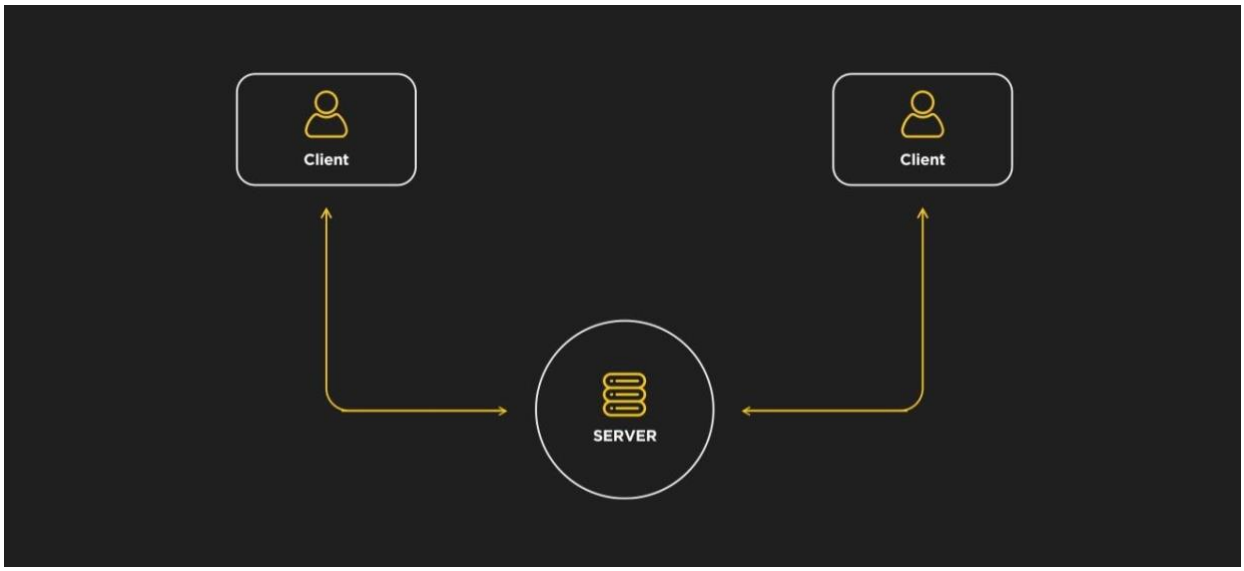


FIG 6

WebRTC JavaScript API

WebRTC is a complex topic where many technologies are involved. However, establishing connections, communication and transmitting data are implemented through a set of JS APIs. The primary APIs include:

- **RTCPeerConnection** – creates and navigates peer-to-peer connections,
- **RTCSessionDescription** – describes one end of a connection (or a potential connection) and how it's configured,
- **navigator.getUserMedia** – captures audio and video.

- **MediaStream (aka getUserMedia):** this interface represents a device's media stream that can include audio and video tracks. The MediaDevies.getUserMedia() method retrieves a MediaStream (for instance, it can be used to access a phone's camera).

- **RTCPeerConnection**: it allows the communication between peers. Streams accessed by MediaDevices.getUserMedia() are added to this component, which also handles the SDP offer and answer messages exchanged between peers and ICE candidates.

RTCDataChannel: it enables real time communication of arbitrary data. It's often compared to WebSockets although it connects browsers to exchange data directly. As explained previously, direct communication between browsers improves performance so this API can be used for some interesting applications like gaming or encrypted file sharing.

TECHNOLOGY ENVIRONMENT

Introduction to Visual Studio

Visual Studio is an **Integrated Development Environment (IDE)** developed by Microsoft to develop GUI (Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB (Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

Evolution of Visual Studio: The first version of VS (Visual Studio) was released in 1997, named as Visual Studio 97 having version number 5.0. The latest version of Visual Studio is 15.0 which was released on March 7, 2017. It is also termed as Visual Studio 2017. The supported *.Net Framework Versions* in latest Visual Studio is 3.5 to 4.7. Java was supported in old versions of Visual Studio but in the latest version doesn't provide any support for Java language.

Visual Studio Editions

There are 3 editions of Microsoft Visual Studio as follows:

1. **Community:** It is a **free** version which is announced in 2014. *All other editions are paid*. This contains the features like Professional edition. Using this edition, any

individual developer can develop their own free or paid apps like .*Net applications*, Web applications and many more. In an enterprise organization, this edition has some limitations. For example, if your organization have more than 250 PCs and having annual revenue greater than $1 Million (US Dollars) then you are not permitted to use this edition. In a non-enterprise organization, up to five users can use this edition. Its main purpose is to provide the Ecosystem (Access to thousands of extensions) and Languages (You can code in C#, VB, F#, C++, HTML, JavaScript, Python, etc.) support.

2. **Professional:** It is the commercial edition of Visual Studio. It comes in Visual Studio 2010 and later versions. It provides the support for XML and XSLT editing and includes the tool like Server Explorer and integration with Microsoft SQL Server. Microsoft provides a free trial of this edition and after the trial period, the user must pay to continue using it. Its main purpose is to provide Flexibility (Professional developer tools for building any application type), Productivity (Powerful features such as CodeLens improve your team's productivity), Collaboration (Agile project planning tools, charts, etc.) and Subscriber benefits like Microsoft software, plus Azure, Pluralsight, etc.
3. **Enterprise:** It is an integrated, end to end solution for teams of any size with the demanding quality and scale needs. Microsoft provides a 90-days free trial of this edition and after the trial period, the user must pay to continue using it. The main benefit of this edition is that it is highly scalable and deliver high-quality software.

Getting Started with Visual Studio 2017

- First, you must download and install the Visual Studio. For that, you can refer to **Downloading and Installing Visual Studio 2017**. Do not forget to select the .NET core workload during the installation of VS 2017. If you forget then you must **modify** the installation.
- You can see several tool windows when you will open the Visual Studio and start writing your first program as follows:

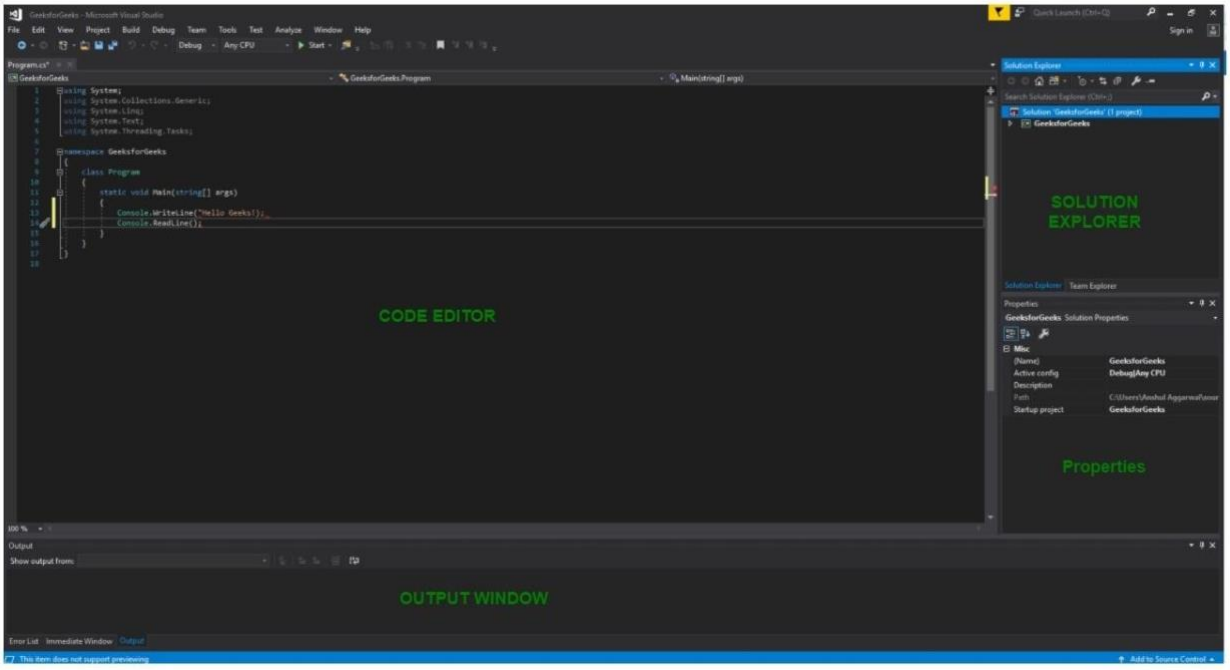


FIG 7

1. **Code Editor:** Where the user will write code.
2. **Output Window:** Here the Visual Studio shows the outputs, compiler warnings, error messages and debugging information.
3. **Solution Explorer:** It shows the files on which the user is currently working.
4. **Properties:** It will give additional information and context about the selected parts of the current project.

- A user can also add windows as per requirement by choosing them from **View** menu. In Visual Studio the tool windows are customizable as a user can add more windows, remove the existing open one or can move windows around to best suit.

- **Various Menus in Visual Studio:** A user can find a lot of menus on the top screen of Visual Studio as shown below

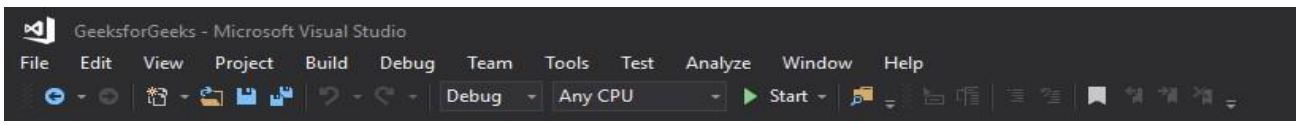


FIG 8

1. Create, Open and save projects commands are contained by **File** menu.
2. Searching, Modifying, Refactoring code commands are contained by the **Edit** menu.
3. **View** Menu is used to open the additional tool windows in Visual Studio.
4. **Project** menu is used to add some files and dependencies in the project.
5. To change the settings, add functionality to Visual Studio via extensions, and access various Visual Studio tools can be used by using **Tools** menu.

- The below menu is known as the **toolbar** which provide the quick access to the most frequently used commands. You can add and remove the commands by going to **View → Customize**

FIG 9

Note:

- Support for different programming languages in Visual Studio is added by using a special **VSPackage** which is known as *Language Service*.
- When you will install the Visual Studio then the functionality which is coded as VSPackage will be available as Service.
- Visual Studio IDE provides the three different types of services known as **SVsSolution**, **SVsUIShell**, and **SVsShell**.
- SVsSolution service is used to provide the functionality to enumerate solutions and projects in Visual Studio.
- SVsUIShell service is used to provide User Interface functionality like toolbars, tabs etc.
- SvsShell service is used to deal with the registration of VSPackages.

Nodejs

Introduction to Nodejs -

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the "hello world" example, many connections can be handled concurrently. Upon each connection, the call-back is fired, but if there is no work to be done, Node.js will sleep.

This contrasts with today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

Node.js is similar in design to, and influenced by, systems like Ruby's Event Machine and Python's Twisted. Node.js takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library. In other systems, there is always a blocking call to start the event-loop. Typically, behavior is defined through call-backs at the beginning of a script, and at the end a server is started through a blocking call like EventMachine::run(). In Node.js, there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no more call-backs to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.

HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js well suited for the foundation of a web library or framework.

Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our child_process.fork() API, and are designed to be easy to communicate with. Built upon that same interface is the cluster module, which allows you to share sockets between processes to enable load balancing over your cores.

The API reference documentation provides detailed information about a function or object in Node.js. This documentation indicates what arguments a method accepts, the return value of that method, and what errors may be related to that method. It also indicates which methods are available for different versions of Node.js.

FIG 10

GITBASH

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

See git tutorial to get started, then see git every day for a useful minimum set of commands. The Git User's Manual has a more in-depth introduction.

After you mastered the basic concepts, you can come back to this page to learn what commands Git offers. You can learn more about individual Git commands with "git help command". Git cli manual page gives you an overview of the command-line command syntax.

FIG 11

CHAPTER 3

SOURCE CODE

GITHUB link for the project: -
https://github.com/deepansharya1111/virtual-conferencing-app-for-college

TOTAL FILES IN THE PROJECT

There are 2 folders and one server file and some utility files in this project.

1. **assets**
2. **js**
3. **app.js**
4. **index.html**

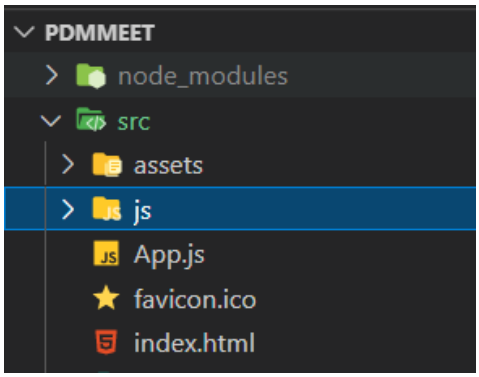


FIG 12

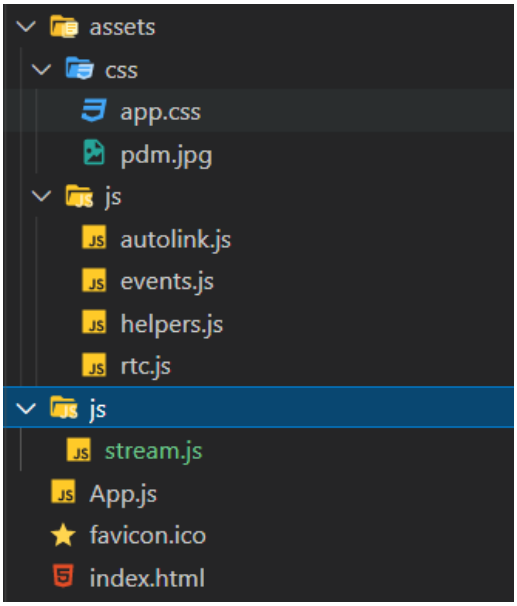


FIG 13

1. **Assets:** - There are 2 folders i.e., CSS and js.

1.1 CSS: - our CSS code.

```
body {
/* font-family: 'Roboto', sans-serif; */
/* font-size: 16px; */
background-image: url("pdm.jpg") ; background-size: cover;
/* margin: 20px; */
}

.chat-col{
    right: -100vw;
    bottom: 0;
    top: 40.5px;
    z-index: 1000;
    position: fixed;
    color: #fff;
    padding-right: 5px;
    padding-left: 5px;
    padding-bottom: 40px;
    padding-top: 15px;
    min-height: 100vh;
}

.chat-box{
    bottom: 30px;
    right: 0;
    position: absolute;
    border: 0;
    border-top: 1px groove white;
    border-left: 1px groove white;
    font-size: small;
}

.remote-video-controls{
    position:absolute;
    bottom: 0;
    background-color:rgba(0, 0, 0, 0.5);
    z-index:300000;
    padding: 10px;
    width: 100%;
    text-align: center;
    visibility: hidden;
}
```

```
.remote-video:hover + .remote-video-controls,
.remote-video-controls:hover{
    visibility: visible;
}


.mirror-mode{
    -ms-transform: scaleX(-1);
    -moz-transform: scaleX(-1);
    -webkit-transform: scaleX(-1);
    transform: scaleX(-1);
}


.sender-info{
    font-size: smaller;
    margin-top: 5px;
    align-self: flex-end;
}

.record-option{
    height: 200px;
    border-radius: 10%;
    border: 1px solid #17a2b8;
    cursor: pointer;
    padding: 10px;
    vertical-align: middle;
}

    .chat-opened::-webkit-scrollbar {
        display: none;
    }
    .local-video{
        width:40vw;
    }
}


@media (min-width:768px){
    .card{
        width: 50%;
        z-index: 1000;
    }
}
```

1.2. **js :-** there are 4 js files is there

1.2.1. **autolink.js**: -

```
( function () {
  var autoLink,
  slice = [].slice;

  autoLink = function () {
    var callback, k, linkAttributes, option, options, pattern, v;
    options = 1 <= arguments.length ? slice.call( arguments, 0 ) : [];
    pattern = /(^|[\s\n]|<[A-Za-z]*\/?>)((?:https?|ftp):\/\/[\-A-Z0-
9+\u0026\u2019@#\/%?=()~_|!:,.;]*[\-A-Z0-9+\u0026@#\/%=~()_|])/gi;
    if ( !( options.length > 0 ) ) {
      return this.replace( pattern, "$1<a href='$2'>$2</a>" );
    }
    option = options[0];
    callback = option["callback"];
    linkAttributes = ( ( function () {
    var results;
      results = [];
      for ( k in option ) {
        v = option[k];
        if ( k !== 'callback' ) {
          results.push( " " + k + "='" + v + "'" );
        }
      }
      return results;
    } )() ).join( " " );
    return this.replace( pattern, function ( match, space, url ) {
      var link;
      link = ( typeof callback === "function" ? callback( url ) : void 0 ) || ( "<a href='" +
url + "'" + linkAttributes + ">" + url + "</a>" );
      return "" + space + link;
    } );
  };

  String.prototype['autoLink'] = autoLink;
} ).call( this );
```

1.2.2. events.js: -

```
import helpers from './helpers.js';

window.addEventListener( 'load', () => {
//When the chat icon is clicked
  document.querySelector( '#toggle-chat-pane' ).addEventListener( 'click', ( e ) => {
    let chatElem = document.querySelector( '#chat-pane' );
    let mainSecElem = document.querySelector( '#main-section' );

    if ( chatElem.classList.contains( 'chat-opened' ) ) {
      chatElem.setAttribute( 'hidden', true );
      mainSecElem.classList.remove( 'col-md-9' );
    }

    else {
      chatElem.attributes.removeNamedItem( 'hidden' );
      chatElem.classList.add( 'chat-opened' );
    }

//remove the 'New' badge on chat icon (if any) once chat is opened.
     setTimeout( () => {
    if ( document.querySelector( '#chat-pane' ).classList.contains( 'chat-opened' ) ) {
      helpers.toggleChatNotificationBadge();
    }
    }, 300 );
  } );


//When the video frame is clicked. This will enable picture-in-picture
  document.getElementById( 'local' ).addEventListener( 'click', () => {
    if ( !document.pictureInPictureElement ) {
      document.getElementById( 'local' ).requestPictureInPicture()
        .catch( error => {
          // Video failed to enter Picture-in-Picture mode.
          console.error( error );
        } );
    }

    else {
      document.exitPictureInPicture()
        .catch( error => {
// Video failed to leave Picture-in-Picture mode.
       console.error( error );
        } );
    }
  } );
```

```
//When the 'Create room" is button is clicked
                document.getElementById( 'create-room' ).addEventListener( 'click', ( e ) => {
                    e.preventDefault();

                    let roomName = document.querySelector( '#room-name' ).value;
                    let yourName = document.querySelector( '#your-name' ).value;

                    if ( roomName && yourName ) {
                        //remove error message, if any
                        document.querySelector( '#err-msg' ).innerHTML = "";

                        //save the user's name in sessionStorage
                        sessionStorage.setItem( 'username', yourName );

//create room link
                        let roomLink = `${ location.origin }?room=${ roomName.trim().replace( ' ', '_' )
            }_${ helpers.generateRandomString() }`;

//show message with link to room
                        document.querySelector( '#room-created' ).innerHTML = `Room successfully
            created. Click <a href='${ roomLink }'>here</a> to enter room.
                            Share the room link with your partners.`;

//When the 'Enter room' button is clicked.
                document.getElementById( 'enter-room' ).addEventListener( 'click', ( e ) => {
                    e.preventDefault();

                    let name = document.querySelector( '#username' ).value;

                    if ( name ) {
                        //remove error message, if any
                        document.querySelector( '#err-msg-username' ).innerHTML = "";

                        //save the user's name in sessionStorage
                        sessionStorage.setItem( 'username', name );

                        //reload room
                        location.reload();
                    }

                    else {
                        document.querySelector( '#err-msg-username' ).innerHTML = "Please input
            your name";
                    }
                } );
```

1.2.3. helpers.js :-

```
export default {
    generateRandomString() {
        const crypto = window.crypto || window.msCrypto;
        let array = new Uint32Array(1);

        return crypto.getRandomValues(array);
    },

    getUserAudio() {
        if ( this.userMediaAvailable() ) {
            return navigator.mediaDevices.getUserMedia( {
                audio: {
                    echoCancellation: true,
                    noiseSuppression: true
                }
            } );
        }

        else {
            throw new Error( 'User media not available' );
        }
    },



    shareScreen() {
        if ( this.userMediaAvailable() ) {
            return navigator.mediaDevices.getDisplayMedia( {
                video: {
                    cursor: "always"
                },
                audio: {
                    echoCancellation: true,
                    noiseSuppression: true,
                    sampleRate: 44100
                }
            } );
        }

        else {
            throw new Error( 'User media not available' );
        }
    },
```

```
    getIceServer() {
        return {
            iceServers: [
                {
                    urls: ["stun:eu-turn4.xirsys.com"]
                },
                {
                    username:
"ml0jh0qMKZKd9P_9C0UIBY2G0nSQMCFBUXGlk6IXDJf8G2uiCymg9WwbEJTM
wVeiAAAAAF2_hNSaW5vbGVl",
                    credential: "4dd454a6-feee-11e9-b185-6adcafebbb45",
                    urls: [
                        "turn:eu-turn4.xirsys.com:80?transport=udp",
                        "turn:eu-turn4.xirsys.com:3478?transport=tcp"
                    ]
                }
            ]
        };
    },


    saveRecordedStream( stream, user ) {
        let blob = new Blob( stream, { type: 'video/webm' } );

        let file = new File( [blob], `${ user }-${ moment().unix() }-record.webm` );

        saveAs( file );
    },

    adjustVideoElemSize() {
        let elem = document.getElementsByClassName( 'card' );
        let totalRemoteVideosDesktop = elem.length;
        let newWidth = totalRemoteVideosDesktop <= 2 ? '50%' : (
            totalRemoteVideosDesktop == 3 ? '33.33%' : (
                totalRemoteVideosDesktop <= 8 ? '25%' : (
                    totalRemoteVideosDesktop <= 15 ? '20%' : (
                        totalRemoteVideosDesktop <= 18 ? '16%' : (
                            totalRemoteVideosDesktop <= 23 ? '15%' : (
                                totalRemoteVideosDesktop <= 32 ? '12%' : '10%'
                            )
                        )
                    )
                )
            )
        );
```

1.2.4 rtc.js: -

```
import h from './helpers.js';


//Get user video by default
getAndSetUserStream();


socket.on( 'connect', () => {
  //set socketId
  socketId = socket.io.engine.id;


  socket.emit( 'subscribe', {
    room: room,
    socketId: socketId
  } );


  socket.on( 'new user', ( data ) => {
    socket.emit( 'newUserStart', { to: data.socketId, sender: socketId } );
    pc.push( data.socketId );
    init( true, data.socketId );
  } );


  socket.on( 'newUserStart', ( data ) => {
    pc.push( data.sender );
    init( false, data.sender );
  } );

function sendMsg( msg ) {
  let data = {
    room: room,
    msg: msg,
    sender: username
  };


//send ice candidate to partnerNames
   pc[partnerName].onicecandidate = ( { candidate } ) => {
    socket.emit( 'ice candidates', { candidate: candidate, to: partnerName, sender:
socketId } );
  };
```

```
//When the video icon is clicked
                document.getElementById( 'toggle-video' ).addEventListener( 'click', ( e ) => {
                    e.preventDefault();

                    let elem = document.getElementById( 'toggle-video' );

                    if ( myStream.getVideoTracks()[0].enabled ) {
                        elem.setAttribute( 'title', 'Show Video' );
                        myStream.getVideoTracks()[0].enabled = false;
                    }

                    else {
                        elem.setAttribute( 'title', 'Hide Video' );
                        myStream.getVideoTracks()[0].enabled = true;
                    }

                    broadcastNewTracks( myStream, 'video' );
                } );


//When the mute icon is clicked
                document.getElementById( 'toggle-mute' ).addEventListener( 'click', ( e ) => {
                    e.preventDefault();

                    let elem = document.getElementById( 'toggle-mute' );

                    if ( myStream.getAudioTracks()[0].enabled ) {
                        e.target.classList.remove( 'fa-microphone-alt' );
                        elem.setAttribute( 'title', 'Unmute' );

                        myStream.getAudioTracks()[0].enabled = false;
                    }

                    else {
                        elem.setAttribute( 'title', 'Mute' );
                } );


//When user clicks the 'Share screen' button
                document.getElementById( 'share-screen' ).addEventListener( 'click', ( e ) => {
                    e.preventDefault();

                    if ( screen && screen.getVideoTracks().length &&
            screen.getVideoTracks()[0].readyState != 'ended' ) {
                        stopSharingScreen();
                        }
```

```
            else {
                shareScreen();
            }
        } );


//When record button is clicked
        document.getElementById( 'record' ).addEventListener( 'click', ( e ) => {
            /**
             * Ask user what they want to record.
             * Get the stream based on selection and start recording
             */
            if ( !mediaRecorder || mediaRecorder.state == 'inactive' ) {
                h.toggleModal( 'recording-options-modal', true );
            }

            else if ( mediaRecorder.state == 'paused' ) {
                mediaRecorder.resume();
            }

            else if ( mediaRecorder.state == 'recording' ) {
                mediaRecorder.stop();
            }
        } );
```

2. js (stream.js): -

```
const stream = ( socket ) => {
    socket.on( 'subscribe', ( data ) => {
        //subscribe/join a room
        socket.join( data.room );
        socket.join( data.socketId );

        //Inform other members in the room of new user's arrival
        if ( socket.adapter.rooms[data.room].length > 1 ) {
            socket.to( data.room ).emit( 'new user', { socketId: data.socketId } );
        }
    } );
    socket.on( 'newUserStart', ( data ) => {
        socket.to( data.to ).emit( 'newUserStart', { sender: data.sender } );
    } );


    socket.on( 'sdp', ( data ) => {
        socket.to( data.to ).emit( 'sdp', { description: data.description, sender: data.sender }
);
```

```
    } );


    socket.on( 'ice candidates', ( data ) => {
        socket.to( data.to ).emit( 'ice candidates', { candidate: data.candidate, sender:
data.sender } );
    } );


    socket.on( 'chat', ( data ) => {
        socket.to( data.room ).emit( 'chat', { sender: data.sender, msg: data.msg } );
    } );
};

module.exports = stream;
```

3. App.js: -

```
let express = require( 'express' );
let app = express();
let server = require( 'http' ).Server( app );
let io = require( 'socket.io' )( server );
let stream = require( './ws/stream' );
let path = require( 'path' );
let favicon = require( 'serve-favicon' );

app.use( favicon( path.join( __dirname, 'favicon.ico' ) ) );
app.use( '/assets', express.static( path.join( __dirname, 'assets' ) ) );
app.get( '/', ( req, res ) => {
    res.sendFile( __dirname + '/index.html' );
} );

io.of( '/stream' ).on( 'connection', stream );

server.listen( process.env.PORT || 3000 );
```

4. Index.html :-

```
<!DOCTYPE html>
<html>
   <head>
      <title>PDM-TEAMS(A Platform For Pdmites)</title>

<link rel="apple-touch-icon" type="image/png" sizes="180x180" href="assets/favicon/apple-touch-icon.png">
<link rel="icon" type="image/png" sizes="32x32" href="assets/favicon/favicon-32x32.png">
<meta name="msapplication-TileColor" content="#da532c">
<meta name="theme-color" content="#ffffff">

      <meta content="width=device-width, initial-scale=1" name="viewport" />

      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
      <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
      <link rel='stylesheet' href='assets/css/app.css' type="text/css">

      <script src='/socket.io/socket.io.js'></script>
      <script type="module" src='assets/js/rtc.js'></script>
      <script type="module" src='assets/js/events.js'></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/webrtc-adapter/7.3.0/adapter.min.js"
integrity="sha256-2qQheewaqnZlXJ3RJRghVUwD/3fD9HNqxh4C+zvgmF4="
crossorigin="anonymous"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.24.0/moment.min.js"></script>
      <script
src='https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/1.3.8/FileSaver.min.js'></script>
      <script src='https://cdn.rawgit.com/yahoo/xss-filters/master/dist/xss-filters.js'></script>
      <script src='assets/js/autolink.js'></script>
   </head>

   <body>
      <div class="custom-modal" id='recording-options-modal'>
         <div class="custom-modal-content">
            <div class="row text-center">
               <div class="col-md-6 mb-2">
                  <span class="record-option" id='record-video'>Record video</span>
               </div>
               <div class="col-md-6 mb-2">
```

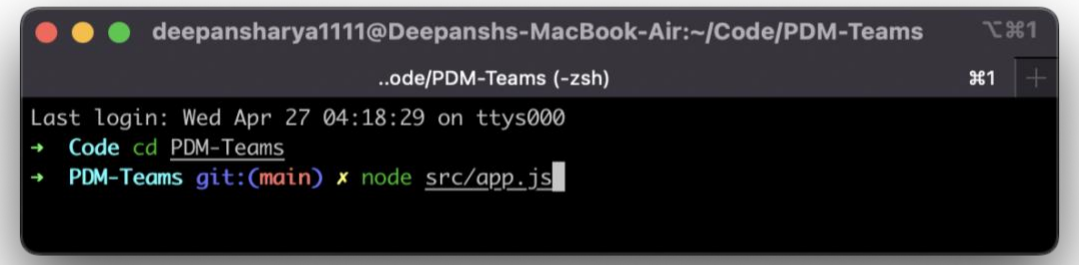
```
                <span class="record-option" id='record-screen'>Record screen</span>
              </div>
            </div>

            <div class="row mt-3">
              <div class="col-md-12 text-center">
                <button class="btn btn-outline-danger" id='closeModal'>Close</button>
              </div>
            </div>
          </div>
        </div>



        <nav class="navbar fixed-top bg-info rounded-0 d-print-none">
          <!-- <img src="logo.png" alt=""> -->
          <h2><div class="text-white" align="center">Pdm-Teams</div></h2>
          <h4 align="right"><div class="text-white">Made with ♥ by Deepansh Singh</div></h4>
         <!-- <h4><img src="./logo.png" alt=""></h4> -->
          <div class="pull-right room-comm" hidden>
            <button class="btn btn-sm rounded-0 btn-no-effect" id='toggle-video' title="Hide
Video">
              <i class="fa fa-video text-white"></i>
            </button>

            <button class="btn btn-sm rounded-0 btn-no-effect" id='toggle-mute' title="Mute">
              <i class="fa fa-microphone-alt text-white"></i>
            </button>

            <button class="btn btn-sm rounded-0 btn-no-effect" id='share-screen' title="Share
screen">
              <i class="fa fa-desktop text-white"></i>
            </button>

            <button class="btn btn-sm rounded-0 btn-no-effect" id='record' title="Record">
              <i class="fa fa-dot-circle text-white"></i>
            </button>

            <button class="btn btn-sm text-white pull-right btn-no-effect" id='toggle-chat-pane'>
              <i class="fa fa-comment"></i> <span class="badge badge-danger very-small font-
weight-lighter" id='new-chat-notification' hidden>New</span>
            </button>

            <button class="btn btn-sm rounded-0 btn-no-effect text-white">
              <a href="/" class="text-white text-decoration-none"><i class="fa fa-sign-out-alt text-
white" title="Leave"></i></a>
            </button>
```

```
        </div>
      </nav>
      <!-- <h1><img src="./logo.png" alt=""></h1> --> -->
      <div class="container-fluid" id='room-create' hidden>

        <div class="row">
          <div class="col-12 h2 mt-5 text-center" style="color: black">Create Room</div>
        </div>

        <div class="row mt-2">
          <div class="col-12 text-center">
            <span class="form-text small text-danger" id='err-msg'></span>
          </div>

          <div class="col-12 col-md-4 offset-md-4 mb-3">
            <label for="room-name" style="color: white">Enter Room Name</label>
            <input type="text" id='room-name' class="form-control rounded-0"
placeholder="Room Name">
          </div>

          <div class="col-12 col-md-4 offset-md-4 mb-3">
            <label for="your-name" style="color: white">Enter Your Name</label>
            <input type="text" id='your-name' class="form-control rounded-0" placeholder="Your
Name">
          </div>

          <div class="col-12 col-md-4 offset-md-4 mb-3">
            <button id='create-room' class="btn btn-block rounded-0 btn-info">Create
Room</button>
          </div>

          <div class="col-12 col-md-4 offset-md-4 mb-3" id='room-created'></div>
        </div>
      </div>
    </body>
</html>
```

CHAPTER 4

TESTING & RESULT

RUNNING THE PROJECT



Now type **" localhost:3000 "** in the address bar of your browser to see the node.js project app.

BASIC VIEW OF THE PROJECT

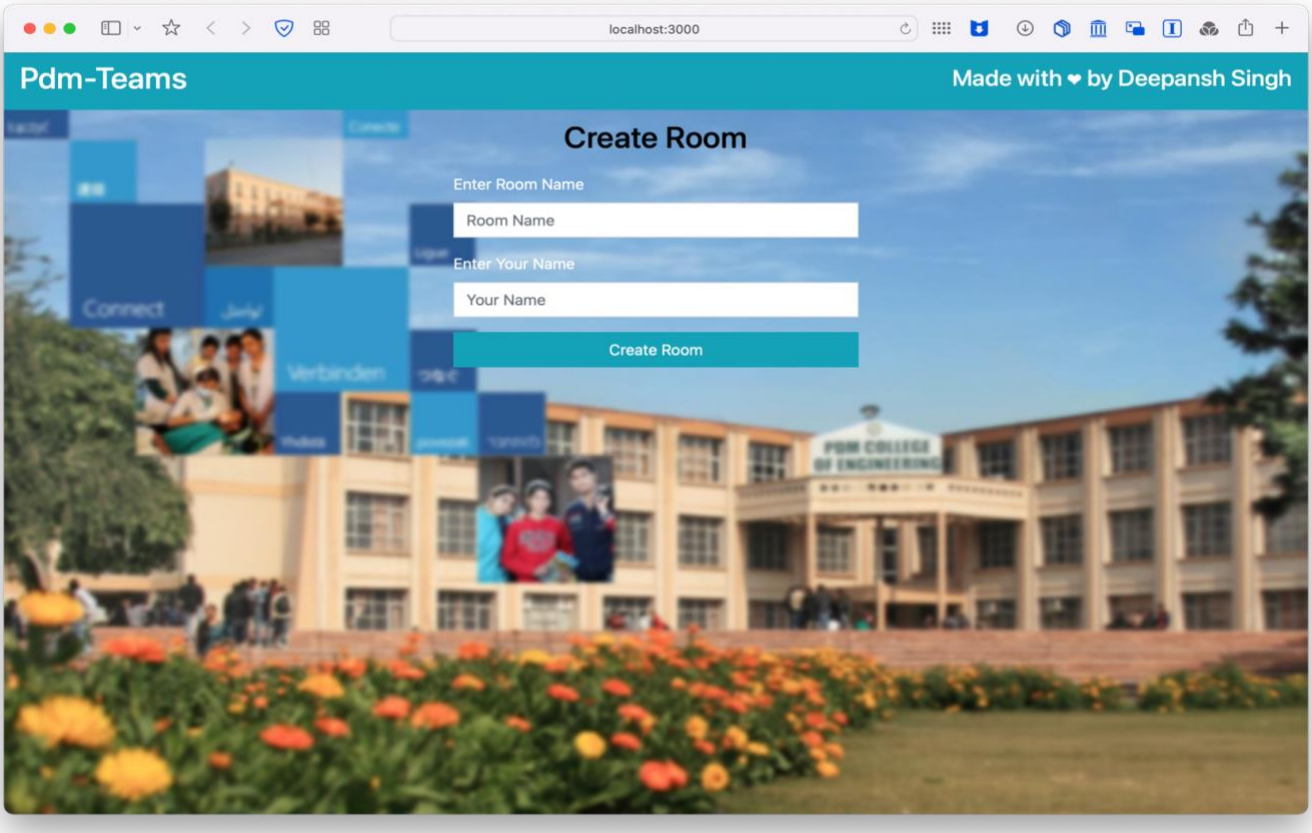


FIG 14

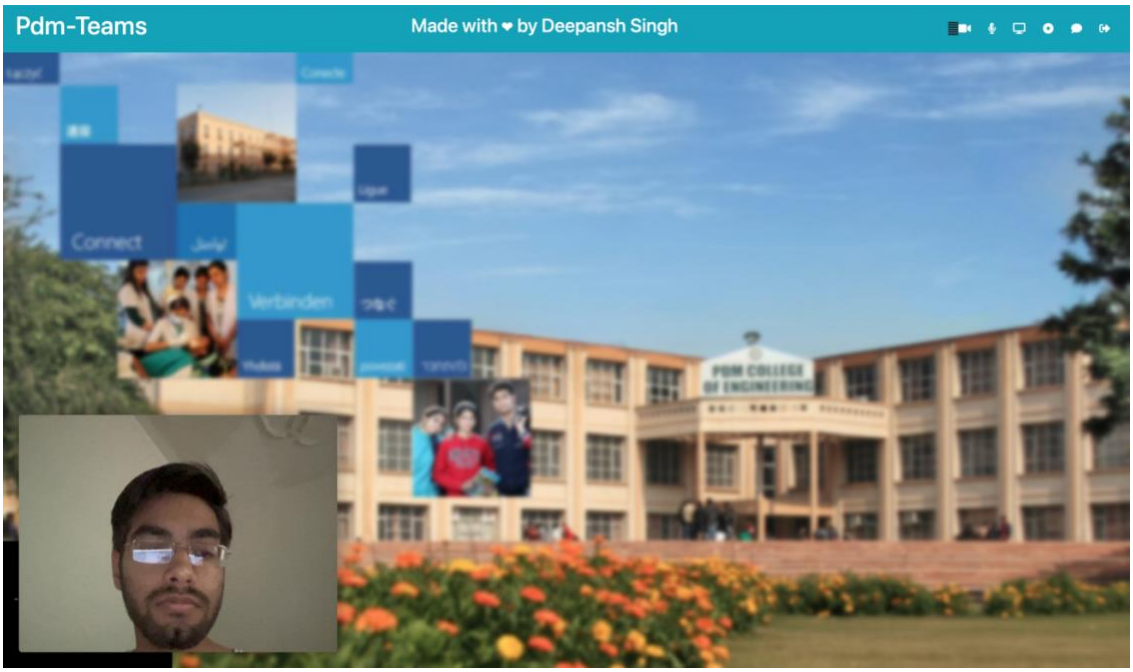


FIG 15

A NEW USER JOINS THE ROOM: -

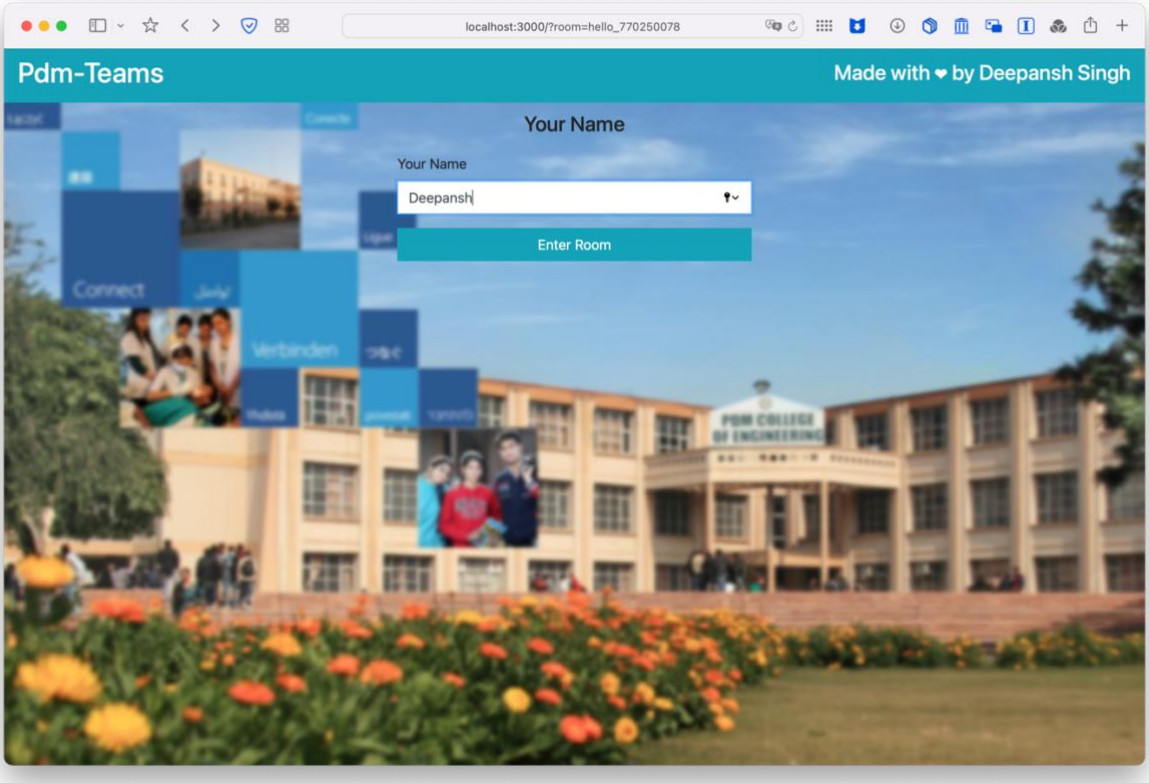


FIG 16

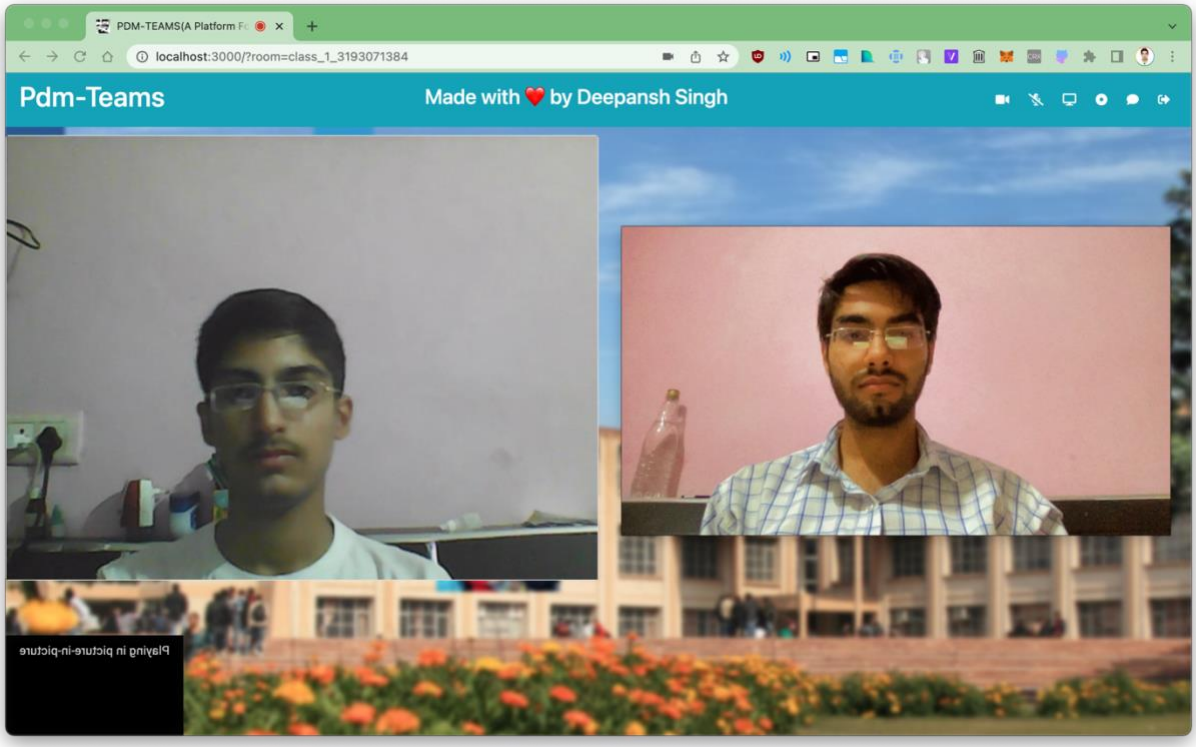


FIG 17

CHAT USING WEB SOCKETS: -

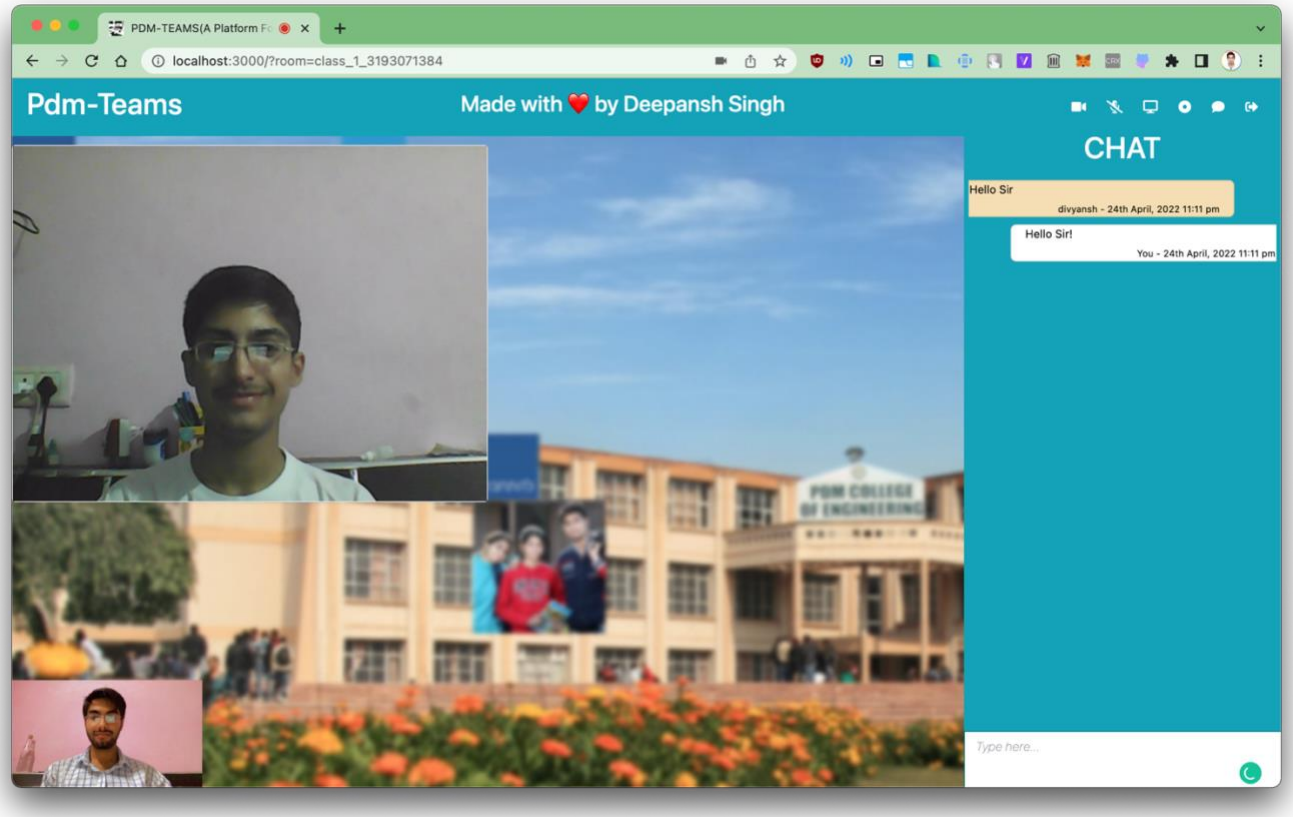


FIG 18

GET UNIVERSAL ACCESS USING NGROK: -

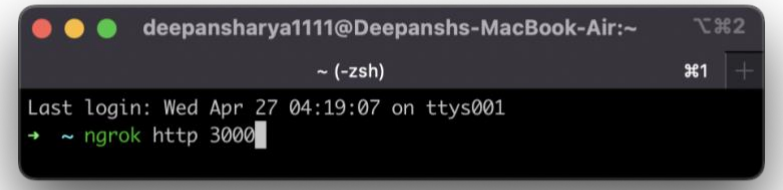


FIG 19

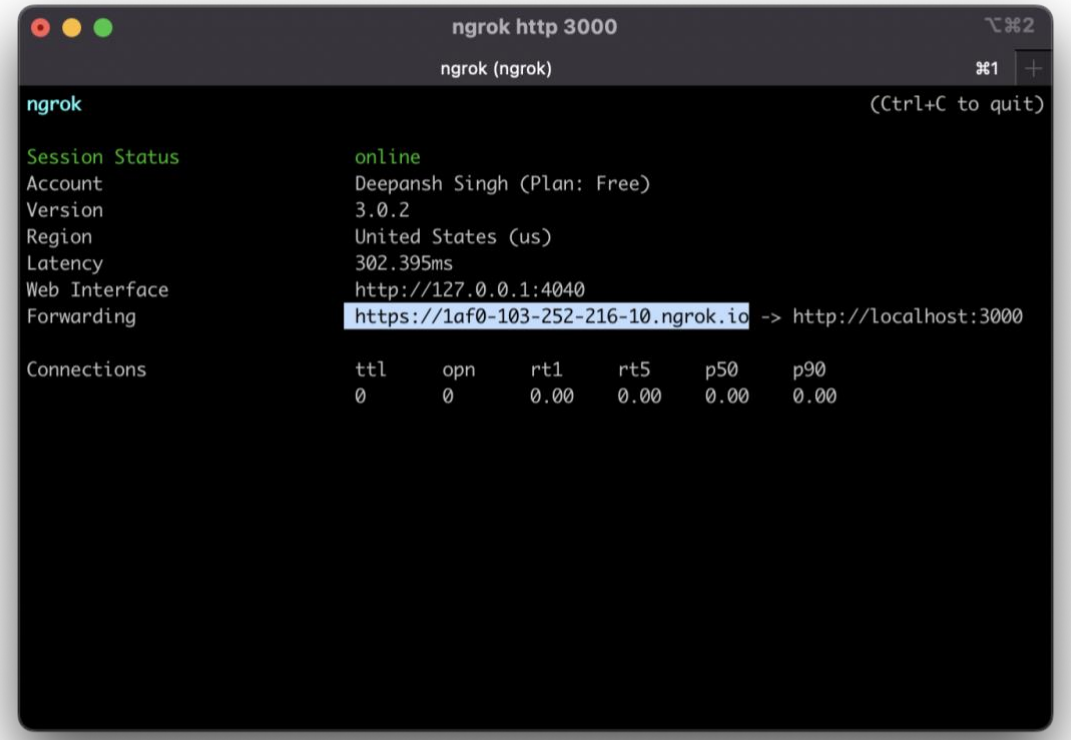


FIG 20

RESULT: - Successfully compiled the projects with ability to use it anywhere on web using Ngrok's localhost port forwarding. This can now be deployed and run on the local organization's server or on a cloud platform like Google Cloud Firebase, Azure, Heroku, Xirsys, etc.

CHAPTER 5

ADVANTAGES AND DISADVANTAGES

- **ADVANTAGES**

1. **Open Source**

   Node.js is open source, so it's free to use and no need to pay for license. There are also many open-source modules supported by Node.js.

2. **JavaScript as Programming Language**

   It uses JavaScript as a programming language for both front-end and back-end which increase programmer productivity and code reusability.

3. **Scalable**

   You can scale your Node.js application by using two ways – Horizontal Scaling and Vertical Scaling, which helps you to improve your application performance.

   - In Horizontal scaling you can add more nodes to your existing system.
   - In Vertical scaling you can add more resources to a single node.

4. **Better Performance**

   It provides better performance, since Node.js I/O operations are non-blocking. Also, it uses V8 JavaScript engine to execute JavaScript code. V8 engine compiles the JS code directly into machine code which make it fast.

5. **Caching Support**

   Node.js supports caching of modules. Hence, when a Node.js modules is requested first time, it is cached into the application memory. So next calls for loading the same module may not cause the module code to be executed again.

6. **Lightweight and Extensible**

   Node.js is based on JavaScript which can be executed on client side as well as server side. Also, it supports exchange of data using JSON which is easily consumed by JavaScript. This makes it light weight as compared to other frameworks.
   Node.js is open source. Hence you can extend it as per your need.

7. **REST API Support**

   Using Node.js you can also develop RESTful services API easily.

8. **Unit Testing**
   It supports unit testing out of box. You can use any JS unit testing frameworks like Jasmin to test your Node.js code.

9. **Server Development**
   Node.js has some built-in API which help you to create different types of Server like HTTP Server, DNS Server, TCP Server etc.

10. **Community Support**
    Node.js has a wide community of developers around the world. They are active in development of new modules or packages to support different types of applications development.

- **DISADVANTAGES**

1. It doesn't support multi-threaded programming.

2. It doesn't support very high computationally intensive tasks. When it executes long running task, it will queue all the incoming requests to wait for execution, since it follows JavaScript event loop which is single threaded.

3. Node good for executing synchronous and CPU intensive tasks.

CONCLUSION

We have now successfully built a PDM-Teams video conferencing application and have a fair understanding of how messages audio and video are exchanged from one user to another.

The current app is deployed on Heroku. If you are interested in checking it out, go to https://pdm-teams.herokuapp.com/

Pre-Created room
https://pdm-teams.herokuapp.com/?room=1

CHAPTER 6

BILIOGRAPHY

1. https://css-tricks.com/
2. https://nodejs.org/en/
3. https://developer.mozilla.org/en-US/
4. https://www.npmjs.com/package/websocket
5. https://expressjs.com/
6. https://medium.com/better-programming/building-a-chat-application-from-scratch-with-room-functionality-df3d1e4ef662
7. https://www.dotnettricks.com/learn/nodejs/advantages-and-limitations-of-nodejs
8. https://www.bacancytechnology.com/blog/why-nodejs-for-real-time-application-development.
9. https://www.slideshare.net/.
10. https://www.npmjs.com/package/mongodb.
11. https://webrtc.org/getting-started/overview
12. https://www.falcon.io/

CHAPTER 7

OBJECTIVE& SCOPE

- Our university website doesn't have a real time chat feature.
- So, it is very difficult for the user to get instant answers of their query.
- We also face the same problem in the period of covid-19.
- So, we are decided to make our minor project to fix this issue.
- As we are providing the room feature to the project, so it is very easy to answer the query of outside user as well as university user.

FUTURE SCOPE

- Set up a login system and can have a friends list.
- Add About and Profile sections.
- Try to implement features like WhatsApp.

CHAPTER 8
METHODOLOGY

Agile is a set of techniques to manage software development projects. It consists in: • Being able to respond to changes and new requirements quickly. • Teamwork, even with the client. • Building operating software over extensive documentation. • Individuals and their interaction over tools. We believed it was a perfect fit for our project since we did not know most requirements beforehand. By using the Agile, we were able to focus only on the features which had the most priority at the time.

⇨ **Agile project management** is an iterative approach to **project management** which allows you to break large projects down into more manageable tasks tackled in short iterations or sprints. This enables your team to adapt to change quickly and deliver work fast.

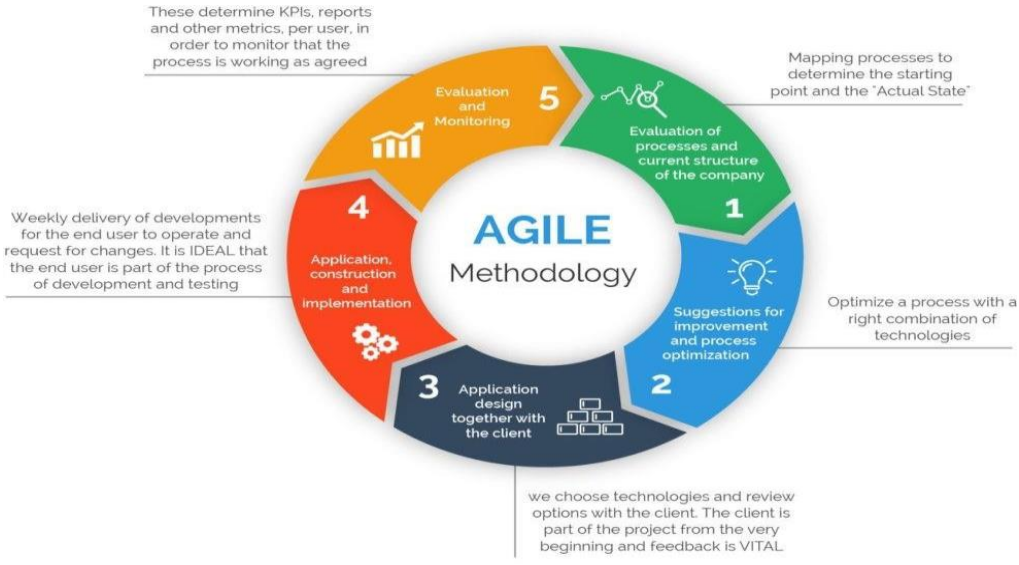


FIG 21

Project timeline

Fig 22

REFERENCES

1. https://css-tricks.com/
2. https://nodejs.org/en/
3. https://developer.mozilla.org/en-US/
4. https://www.npmjs.com/package/websocket
5. https://expressjs.com/
6. https://webrtc.org/getting-started/overview
7. https://www.falcon.io/