Hello everyone.My name is Divya,I am a second year undergraduate pursuing B.Tech, in ECE, from Igdtuw. For Engage 2021, the challenge given to us this year is to build a Microsoft Teams clone.

Microsoft Teams is a vast and versatile application used for group chatting and organizing workflow of teams. The challenge appeared to be intimidating in the start as I had no experience before in web development, but I tried my best to build the clone with the help of my mentors and my team-mates.

During the Engage mentorship program, Microsoft mentors introduced us to the agile methodology and encouraged us to incorporate it into our development phase. I tried to follow the agile methodology by dividing my development phase into three sprints starting from June 14 which was the date of commencement of the challenge for this year,i.e,2021.

A big thanks to Microsoft for making it possible for us to work on this amazing project, and giving us the opportunity to learn new things, explore new domains, and hone our skills and special thanks to my mentors who always stood beside to help and each and every step of the program. I am honored to be Microsoft Engage 2021 Mentee.

* **FIRST SPRINT:-**

During the first sprint, I took time to research various libraries and SDKs available to build video call applications. I came across webRTC, Jitsi, Twilio, Azure communication services, and many other platforms, but I decided to stick with webRTC and familiarized myself with it. I also designed the wireframe for my video call app and decided what features I wanted to have, and also decided the color scheme to be similar to Microsoft Teams.

In the first sprint,I focussed more on the mandatory feature given to us by Microsoft,i.e,built a peer to peer connection between 2 nodes.

* **SECOND SPRINT:-**

In the second sprint, I tried to implement features like Leave meeting,mic on/off,video on/off,Record screen and audio,screen sharing. I took the help of various tutorials and documentation available online. It was my first time building a web application using React JS and Node JS, so bugs were no strangers to me and frequently gave me a hard time, and

* **THIRD SPRINT:-**

The third sprint surprised us with the chat feature that we had to adapt to our video call app. My in-depth research about video call platforms and frequent YouTube tutorial visits made it clear to me that I had to build a chat application using Nodejs and socket io and link it to my previous video call app.I took on the challenge to debug my bugs which are part of the project. I also tweaked the UI to make it more eye-pleasing.

* **FOURTH SPRINT:-**

And I finally made it to the last week of the program.In the last week,I took time to refine my UI and my features.Made a report as to why I chose the particular Tech stack to build the clone.

**TECH-STACK USED**

1. **WebRTC**

I used WebRTC to build the video call applications because it has the most impressive feature to stream real-time video across browsers without requiring plugins or downloads.WebRTC requires no downloads or plugins to work. It operates automatically in Chrome, Firefox, and Opera, which frees consumers from worrying about the universe of technological widgets that sometimes seem to create more problems than they solve. Some of the APIs provided

1. **WebRTC JavaScript API**

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.

1. **NODE.JS**

To make a remote connection between two or more devices we need a server. In this case, we need a server that handles real-time communication. As we know, Node.js is built for real-time scalable applications. To develop two-way connection apps with free data exchange, we would probably use WebSockets that allows opening a communication session between a client and a server. Requests from the client are processed as a loop, more precisely – the event loop, which makes Node.js a good option because it takes a “non-blocking” approach to serve requests and thus, achieves low latency and high throughput along the way.

Also used:-

* express library to serve static files like our HTML file which stands for our UI,
* socket.io library to establish a connection between two devices with WebSockets,
* WebRTC to allow media devices (camera and microphone) to stream audio and video between connected devices.