LAMPIRAN

1. App.js

```
import * as React from 'react';
import { Button, View, Text }
from 'react-native';
import { NavigationContainer }
from '@react-navigation/native';
import {
createNativeStackNavigator }
from '@react-navigation/native-
stack';

import MainMenu from
"./components/MainMenu/Main
Menu.component";
// import MainScreen from
"./components/MainScreen/Main
Screen.component";
import firepadRef, { db,
userName } from
"./server/firebase";
import "./App.css";
// import { useEffect } from
"react";
import {
  setMainStream,
  addParticipant,
  setUser,
  removeParticipant,
  updateParticipant,
} from "./store/actioncreator";
import { connect } from "react-
redux";

function App(props) {
  const getUserStream = async ()
=> {
    const localStream = await
navigator.mediaDevices.getUser
Media({
      audio: true,
      video: true,
    });

    return localStream;
  };

  useEffect(async () => {
    const stream = await
getUserStream();

stream.getVideoTracks()[0].enab
led = false;
    props.setMainStream(stream);

    connectedRef.on("value",
(snap) => {
      if (snap.val()) {
        const defaultPreference = {
          audio: true,
          video: false,
          screen: false,
        };
        const userStatusRef =
participantRef.push({
          userName,
          preferences:
defaultPreference,
        });
        props.setUser({
          [userStatusRef.key]: {
name: userName,
...defaultPreference },
        });

userStatusRef.onDisconnect().re
move();
      }
    });
  }, []);

  const connectedRef =
db.database().ref(".info/connecte
d");
  const participantRef =
firepadRef.child("participants");

  const isUserSet = !!props.user;
  const isStreamSet =
!!props.stream;

  useEffect(() => {
```

```
   if (isStreamSet && isUserSet)
{

participantRef.on("child_added",
(snap) => {
       const
preferenceUpdateEvent =
participantRef
         .child(snap.key)
         .child("preferences");

preferenceUpdateEvent.on("child
_changed", (preferenceSnap) =>
{
         props.updateParticipant({
           [snap.key]: {
           [preferenceSnap.key]:
preferenceSnap.val(),
           },
         });
       });
       const { userName: name,
preferences = {} } = snap.val();
       props.addParticipant({
         [snap.key]: {
           name,
           ...preferences,
         },
       });
     });

participantRef.on("child_remove
d", (snap) => {

props.removeParticipant(snap.ke
y);
     });
   }
  }, [isStreamSet, isUserSet]);

  return (
    <div className="App">
      <MainMenu />
      {/* <MainScreen /> */}
    </div>
  );
}

const mapStateToProps = (state)
=> {
  return {
    stream: state.mainStream,
    user: state.currentUser,
  };
};

const mapDispatchToProps =
(dispatch) => {
  return {
    setMainStream: (stream) =>
dispatch(setMainStream(stream))
,
    addParticipant: (user) =>
dispatch(addParticipant(user)),
    setUser: (user) =>
dispatch(setUser(user)),
    removeParticipant: (userId) =>
dispatch(removeParticipant(userI
d)),
    updateParticipant: (user) =>
dispatch(updateParticipant(user))
,
  };
};

function HomeScreen({
navigation }) {
  return (
    <View style={{ flex: 1,
alignItems: 'center',
justifyContent: 'center' }}>
      <Text>Home Screen</Text>
      <Button
        title="Go to Details"
        onPress={() =>
navigation.navigate('Details')}
      />
    </View>
  );
}
```

```
export default
connect(mapStateToProps,
mapDispatchToProps)(HomeScr
een);
```

2. firebase.js

```
import firebase from "firebase";

var firebaseConfig = {
  // apiKey: "", // Add API Key
  // databaseURL:"" // Add
databaseURL
  apiKey:
"AIzaSyC36L67iXobi1gEUo8tP
PpmRSYIJ35uKbE",
  databaseURL: "https://web-
meet-f4b8b-default-rtdb.asia-
southeast1.firebasedatabase.app",
};
// Initialize Firebase
firebase.initializeApp(firebaseCo
nfig);

export const db = firebase;

var firepadRef =
firebase.database().ref();

export const userName = "TES";

export default firepadRef;
```

3. peerConnection.js

```
import firepadRef from
"./firebase";
import { store } from "../index";

const participantRef =
firepadRef.child("participants");

export const updatePreference =
(userId, preference) => {
  const currentParticipantRef =
participantRef
    .child(userId)
    .child("preferences");
  setTimeout(() => {
currentParticipantRef.update(pref
erence);
  });
};

export const createOffer = async
(peerConnection, receiverId,
createdID) => {
  const currentParticipantRef =
participantRef.child(receiverId);
peerConnection.onicecandidate
= (event) => {
    event.candidate &&
    currentParticipantRef
        .child("offerCandidates")
        .push({
...event.candidate.toJSON(),
userId: createdID });
  };

  const offerDescription = await
peerConnection.createOffer();
await
peerConnection.setLocalDescript
ion(offerDescription);

  const offer = {
    sdp: offerDescription.sdp,
    type: offerDescription.type,
    userId: createdID,
  };

  await
currentParticipantRef.child("offe
rs").push().set({ offer });
};

export const initializeListensers =
async (userId) => {
  const currentUserRef =
participantRef.child(userId);


currentUserRef.child("offers").on
```

```
("child_added", async (snapshot)
=> {
    const data = snapshot.val();
    if (data?.offer) {
      const pc =

store.getState().participants[data.
offer.userId].peerConnection;
      await
pc.setRemoteDescription(new
RTCSessionDescription(data.offe
r));
      await
createAnswer(data.offer.userId,
userId);
    }
  });


currentUserRef.child("offerCandi
dates").on("child_added",
(snapshot) => {
    const data = snapshot.val();
    if (data.userId) {
      const pc =
store.getState().participants[data.
userId].peerConnection;
      pc.addIceCandidate(new
RTCIceCandidate(data));
    }
  });


currentUserRef.child("answers").
on("child_added", (snapshot) =>
{
    const data = snapshot.val();
    if (data?.answer) {
      const pc =

store.getState().participants[data.
 answer.userId].peerConnection;
     const answerDescription =
new
RTCSessionDescription(data.ans
wer);

pc.setRemoteDescription(answer
Description);
    }
  });


currentUserRef.child("answerCa
ndidates").on("child_added",
(snapshot) => {
    const data = snapshot.val();
    if (data.userId) {
      const pc =
store.getState().participants[data.
userId].peerConnection;
      pc.addIceCandidate(new
RTCIceCandidate(data));
    }
  });
};

const createAnswer = async
(otherUserId, userId) => {
const pc =
store.getState().participants[other
UserId].peerConnection;
  const participantRef1 =
participantRef.child(otherUserId)
;
  pc.onicecandidate = (event) =>
{
    event.candidate &&
      participantRef1
        .child("answerCandidates")
        .push({
...event.candidate.toJSON(),
userId: userId });
  };

  const answerDescription =
await pc.createAnswer();
await
pc.setLocalDescription(answerD
escription);

  const answer = {
```

```
    type: answerDescription.type,
    sdp: answerDescription.sdp,
    userId: userId,
  };

  await
participantRef1.child("answers").
push().set({ answer });
};
```

4. MainScreen.component.js

```
import React, { useRef, useEffect
} from "react";
import MeetingFooter from
"../MeetingFooter/MeetingFooter
.component";
import Participants from
"../Participants/Participants.comp
onent";
import "./MainScreen.css";
import { connect } from "react-
redux";
import { setMainStream,
updateUser } from
"../../store/actioncreator";

const MainScreen = (props) => {
const participantRef =
useRef(props.participants);

  const onMicClick =
(micEnabled) => {
    if (props.stream) {

props.stream.getAudioTracks()[0
].enabled = micEnabled;
      props.updateUser({ audio:
micEnabled });
    }
  };
  const onVideoClick =
(videoEnabled) => {
    if (props.stream) {

props.stream.getVideoTracks()[0
].enabled = videoEnabled;
      props.updateUser({ video:
videoEnabled });
    }
  };

  useEffect(() => {
    participantRef.current =
props.participants;
  }, [props.participants]);

  const updateStream = (stream)
=> {
    for (let key in
participantRef.current) {
      const sender =
participantRef.current[key];
      if (sender.currentUser)
continue;
      const peerConnection =
sender.peerConnection
        .getSenders()
        .find((s) => (s.track ?
s.track.kind === "video" : false));

peerConnection.replaceTrack(str
eam.getVideoTracks()[0]);
    }
    props.setMainStream(stream);
  };

  const onScreenShareEnd =
async () => {
    const localStream = await
navigator.mediaDevices.getUser
Media({
      audio: true,
      video: true,
    });


localStream.getVideoTracks()[0].
enabled = Object.values(
      props.currentUser
    )[0].video;

    updateStream(localStream);
```

```
    props.updateUser({ screen:
false });
  };

  const onScreenClick = async ()
=> {
    let mediaStream;
    if (navigator.getDisplayMedia)
{
      mediaStream = await
navigator.getDisplayMedia({
video: true });
    } else if
(navigator.mediaDevices.getDisp
layMedia) {
      mediaStream = await
navigator.mediaDevices.getDispl
ayMedia({
        video: true,
      });
    } else {
      mediaStream = await
navigator.mediaDevices.getUser
Media({
        video: { mediaSource:
"screen" },
      });
    }


mediaStream.getVideoTracks()[0
].onended = onScreenShareEnd;

    updateStream(mediaStream);

    props.updateUser({ screen:
true });
  };
  return (
    <div className="wrapper">
      <div className="main-
screen">
        <Participants />
      </div>
      <div className="footer">
        <MeetingFooter

onScreenClick={onScreenClick}

onMicClick={onMicClick}

onVideoClick={onVideoClick}
        />
      </div>
    </div>
  );
};

const mapStateToProps = (state)
=> {
  return {
    stream: state.mainStream,
    participants: state.participants,
    currentUser: state.currentUser,
  };
};

const mapDispatchToProps =
(dispatch) => {
  return {
    setMainStream: (stream) =>
dispatch(setMainStream(stream))
,
    updateUser: (user) =>
dispatch(updateUser(user)),
  };
};

export default
connect(mapStateToProps,
mapDispatchToProps)(MainScre
en);
```

5. MainScreen.css

```
.main-screen{
    width: 100%;
    height: 90vh;
    background: #3c4043;
}
```

```
.wrapper{
   width: 100%;
}

.footer{
   height: 10vh;
}
```

6. MeetingFooter.component.js

```
import React, { useEffect,
useState } from "react";
import { FontAwesomeIcon }
from "@fortawesome/react-
fontawesome";
import {
 faMicrophone,
 faVideo,
 faDesktop,
 faVideoSlash,
 faMicrophoneSlash,
} from "@fortawesome/free-
solid-svg-icons";
import ReactTooltip from "react-
tooltip";
import "./MeetingFooter.css";
const MeetingFooter = (props)
=> {
 const [streamState,
setStreamState] = useState({
   mic: true,
   video: false,
   screen: false,
 });
 const micClick = () => {
   setStreamState((currentState)
=> {
     return {
       ...currentState,
       mic: !currentState.mic,
     };
   });
 };

 const onVideoClick = () => {
   setStreamState((currentState)
=> {
     return {
       ...currentState,
       video: !currentState.video,
     };
   });
 };

 const onScreenClick = () => {

props.onScreenClick(setScreenSt
ate);
 };

 const setScreenState =
(isEnabled) => {
   setStreamState((currentState)
=> {
     return {
       ...currentState,
       screen: isEnabled,
     };
   });
 };
 useEffect(() => {

props.onMicClick(streamState.m
ic);
 }, [streamState.mic]);
 useEffect(() => {

props.onVideoClick(streamState.
video);
 }, [streamState.video]);
 return (
   <div style={{background:
"#3c4043", display: "flex",
justifyContent: "center", height:
"100%"}}>
     <div className="meeting-
footer">
       <div
         className={"meeting-
icons " + (!streamState.mic ?
"active" : "")}
```

```
        data-tip={streamState.mic
? "Mute Audio" : "Unmute
Audio"}
        onClick={micClick}
       >
        <FontAwesomeIcon
         icon={!streamState.mic ?
faMicrophoneSlash :
faMicrophone}
         title="Mute"
        />
       </div>
       <div
        className={"meeting-
icons " + (!streamState.video ?
"active" : "")}
        data-
tip={streamState.video ? "Hide
Video" : "Show Video"}
        onClick={onVideoClick}
       >
        <FontAwesomeIcon
icon={!streamState.video ?
faVideoSlash : faVideo} />
       </div>
       <div
        className="meeting-
icons"
        data-tip="Share Screen"
        onClick={onScreenClick}

disabled={streamState.screen}
       >
        <FontAwesomeIcon
icon={faDesktop} />
       </div>
       <ReactTooltip />
     </div>
   </div>
  );
};

export default MeetingFooter;
```

7. MeetingFooter.js

```
.meeting-footer{
   background: #202124;
   height: 100%;
   display: flex;
   justify-content: center;
   align-items: center;
   border-radius: 15px;
   margin-top: -10px;
}

.meeting-icons{
   color: white;
   border-radius: 50%;
   background:#3c4043;
   border-radius: 50%;
   width: 40px;
   height: 40px;
   margin: 10px;
   display: flex;
   align-items: center;
   justify-content: center;
   cursor: pointer;
}

.active {
  background-color:  #1d32ee;
}

div[disabled]
{
  pointer-events: none;
  opacity: 0.3;
}
```

8. Participants.component.js

```
import React, { useEffect, useRef
} from "react";
import "./Participants.css";
import { connect } from "react-
redux";
import { Participant } from
"./Participant/Participant.compon
ent";

const Participants = (props) => {
  const videoRef = useRef(null);
```

```
  let participantKey =
Object.keys(props.participants);
useEffect(() => {
    if (videoRef.current) {
      videoRef.current.srcObject =
props.stream;
      videoRef.current.muted =
true;
    }
  }, [props.currentUser,
props.stream]);

  const currentUser =
props.currentUser
    ?
Object.values(props.currentUser)
[0]
    : null;

  let gridCol =
    participantKey.length === 1 ?
1 : participantKey.length <= 4 ?
2 : 4;
  const gridColSize =
participantKey.length <= 4 ? 1 :
2;
  let gridRowSize =
    participantKey.length <= 4
      ? participantKey.length
      :
Math.ceil(participantKey.length /
2);

  const screenPresenter =
participantKey.find((element) =>
{
    const currentParticipant =
props.participants[element];
    return
currentParticipant.screen;
  });

  if (screenPresenter) {
    gridCol = 1;
    gridRowSize = 2;
  }

  const participants =
participantKey.map((element,
index) => {
    const currentParticipant =
props.participants[element];
    const isCurrentUser =
currentParticipant.currentUser;
    if (isCurrentUser) {
      return null;
    }
    const pc =
currentParticipant.peerConnectio
n;
    const remoteStream = new
MediaStream();
    let curentIndex = index;
    if (pc) {
      pc.ontrack = (event) => {

event.streams[0].getTracks().forE
ach((track) => {

remoteStream.addTrack(track);
        });
        const videElement =
document.getElementById(

`participantVideo${curentIndex}
`
        );
        if (videElement)
videElement.srcObject =
remoteStream;
      };
    }

    return (
      <Participant
        key={curentIndex}

currentParticipant={currentPartic
ipant}
        curentIndex={curentIndex}
        hideVideo={screenPresenter
&& screenPresenter !==
element}
```

```
        showAvatar={
          !currentParticipant.video
&&
          !currentParticipant.screen
&&
          currentParticipant.name
        }
      />
    );
  });
  return (
    <div
      style={{
        "--grid-size": gridCol,
        "--grid-col-size":
gridColSize,
        "--grid-row-size":
gridRowSize,
      }}
      className={`participants`}
    >
      {participants}
      <Participant

currentParticipant={currentUser}

curentIndex={participantKey.len
gth}
        hideVideo={screenPresenter
&& !currentUser.screen}
        videoRef={videoRef}
        showAvatar={currentUser
&& !currentUser.video &&
!currentUser.screen}
        currentUser={true}
      />
    </div>
  );
};

const mapStateToProps = (state)
=> {
  return {
    participants: state.participants,
    currentUser: state.currentUser,
    stream: state.mainStream,
  };
};

export default
connect(mapStateToProps)(Parti
cipants);
```

9. Participants.css

```
.participants {
  display: grid;
  grid-template-columns:
repeat(var(--grid-size), 1fr);
grid-gap: 20px;
  height: 85vh;
  padding: 10px;
  }

.participants img{
    border-radius: 50%;
    height: 120px;
    width: 120px;
  }

  @media (max-width: 800px) {
    .participants {
      grid-template-columns:
repeat(var(--grid-col-size), 1fr);
      grid-template-rows:
repeat(var(--grid-row-size), 1fr);
    }
  }
```

10. Participant.component.js

```
import React from "react";
import Card from
"../../Shared/Card/Card.compone
nt";
import { faMicrophoneSlash }
from "@fortawesome/free-solid-
svg-icons";
import { FontAwesomeIcon }
from "@fortawesome/react-
fontawesome";
import "./Participant.css";

export const Participant = (props)
=> {
```

```
  const {
    curentIndex,
    currentParticipant,
    hideVideo,
    videoRef,
    showAvatar,
    currentUser,
  } = props;
  if (!currentParticipant) return
<></>;
  return (
    <div className={`participant
${hideVideo ? "hide" : ""}`}>
      <Card>
        <video
          ref={videoRef}
          className="video"

id={`participantVideo${curentIn
dex}`}
          autoPlay
          playsInline
        ></video>
        {!currentParticipant.audio
&& (
          <FontAwesomeIcon
            className="muted"

icon={faMicrophoneSlash}
            title="Muted"
          />
        )}
        {showAvatar && (
          <div
            style={{ background:
currentParticipant.avatarColor }}
            className="avatar"
          >

{currentParticipant.name[0]}
          </div>
        )}
        <div className="name">
          {currentParticipant.name}
          {currentUser ? "(You)" :
""}
        </div>
      </Card>
    </div>
  );
};
```

11. Participant.css

```
.video{
  background-color: #353b48;
  height: 100%;
  width: 100%;
  position: relative;
  object-fit: cover;
  border-radius: 5px;
}

.participant {
  background-color: #353b48;
  height: 100%;
  width: 100%;
  position: relative;
  border-radius: 5px;
  min-height: 200px
}

.muted {
  color: #fff;
  right: 10px;
  position: absolute;
  top: 10px;
  background: #353b48;
  border-radius: 50%;
  height: 22px;
  width: 22px!important;
}

.hide {
  display: none;
}

.avatar {
  width: 200px;
  height: 200px;
  border-radius: 50%;
```

```
    font-size: 150px;
    font-family: roboto, arial;
    color: white;
    text-align: center;
    background: orange;
    position: absolute;
    display: flex;
    justify-content: center;
    align-items: center;
    text-transform: uppercase;
  }

  .name {
    color: white;
    bottom: 10px;
    position: absolute;
    left: 20px;
  }

  @media (max-width: 800px) {
    .participant{
      min-height: 0px;
    }
    .avatar {
      width: 80px;
      height: 80px;
      font-size: 50px;
    }
  }
```

12. Card.component.js

```
import React from "react";
import "./Card.css";

const Card = (props) => {
  return <div
className="card">{props.childr
en}</div>;
};

export default Card;
```

13. Card.css

```
.card{
    background: #3c4043;
    border-radius: 10px;
    display: flex;
        justify-content: center;
        align-items: center;
        height: 100%;
      }
```

13. Card.css

```
.card{

background: #3c4043; border-radius:
10px; display: flex;
justify-content: center; align-items:
center; height: 100%;
}
```